An investigation into recommendation algorithms with application to dynamic environments

Florian Geier

15 September, 2010

Department of Computing
Faculty of Mathematics, Computing and Technology
The Open University

Walton Hall, Milton Keynes, MK7 6AA
United Kingdom

http://computing.open.ac.uk
An investigation into recommendation algorithms with application to dynamic environments

A dissertation submitted in partial fulfilment of the requirements for the Open University’s Master of Science Degree in Software Development

Florian Geier

4 March 2011

Word Count: 15220
Preface

This dissertation came into being during my work for the Open University’s Master of Science Degree in Software Development. It is based on a project named regionrex.com that I developed in my spare time and that I have grown really fond of, not least because I learned a lot on the way.

At this point, I want to acknowledge everyone who contributed to this project. I would like to thank my tutor, George Haywood, for his helpful support and for all the time he invested in answering my endless questions. Most of all, I am grateful to my family, who stood behind me and encouraged me at all times.
# Table of Contents

Preface ............................................................................................................................................................. i

Table of Contents ............................................................................................................................................. ii

List of Figures .................................................................................................................................................... iv

List of Tables ..................................................................................................................................................... vi

Abstract ............................................................................................................................................................ vii

Chapter 1 Problem Overview ................................................................................................... 1

1.1 Introduction ............................................................................................................................................... 1

1.2 History of recommender systems ........................................................................................................ 1

1.3 The problem domain ............................................................................................................................. 3

1.4 Aims of the research ................................................................................................................................ 5

1.5 Contribution to knowledge .................................................................................................................... 8

1.6 Summary.................................................................................................................................................. 10

Chapter 2 Fundamentals of recommender systems ................................................................ 11

2.1 Classification of recommendation techniques ...................................................................................... 11

2.2 Similarity metrics .................................................................................................................................... 13

2.3 Alternative recommendation algorithms ............................................................................................... 15

2.4 Advantages and disadvantages of individual algorithms ....................................................................... 16

2.5 Hybrid systems ....................................................................................................................................... 17

2.6 Recommender systems and location based services ............................................................................ 19

2.7 Summary.................................................................................................................................................. 20

Chapter 3 Research Methodology .......................................................................................... 21

3.1 Evaluation metrics ................................................................................................................................... 21

3.2 Evaluation strategy .................................................................................................................................. 23

3.3 Setup of the experiments ......................................................................................................................... 24

3.4 Database preparation ............................................................................................................................... 26

3.5 Algorithms under investigation ............................................................................................................. 29
3.6 The hybrid recommendation engine prototype ............................................................... 31
3.7 Possible alternative research methods ............................................................................ 32
3.8 Summary ......................................................................................................................... 32

Chapter 4 Evaluation of recommender systems ..................................................................... 33
4.1 Boundary conditions to the experimental setup .............................................................. 33
4.2 Evaluation results ........................................................................................................... 35
4.3 Data smoothing ............................................................................................................... 39
4.4 Analysis of the observed results ..................................................................................... 42
4.5 Summary ......................................................................................................................... 43

Chapter 5 The hybrid recommendation engine ...................................................................... 44
5.1 The rationale for the hybrid algorithms .......................................................................... 44
5.2 The switching hybrid prototype ...................................................................................... 44
5.3 The weighted hybrid prototype ....................................................................................... 46
5.4 Analysis of the observed results ..................................................................................... 47
5.5 Summary ......................................................................................................................... 52

Chapter 6 Discussion .............................................................................................................. 53
6.1 Analysis of the results with respect to the research aims ............................................... 53
6.2 The current research in a wider context .......................................................................... 53
6.3 Summary ......................................................................................................................... 54

Chapter 7 Conclusions ............................................................................................................ 55
7.1 Project Review ................................................................................................................ 55
7.2 Evaluation of the scientific method ................................................................................ 56
7.3 Future Work ..................................................................................................................... 57

References ............................................................................................................................ 59
Index ..................................................................................................................................... 64

Appendix A: Extended Abstract .......................................................................................... 65
Appendix B: Complete result plots ....................................................................................... 71
Appendix C: Derivation of the polynomial curve fitting function ............................................. 74
List of Figures

Figure 1: Web user interface of regionrex.com ............................................................................. 4
Figure 2: A diagram of users, reviews, locations and their geographical positions .................. 6
Figure 3: A graphical representation of users, items and ratings ................................................ 13
Figure 4: A graphical representation of the Slope One example ............................................... 15
Figure 5: Experimental setup ....................................................................................................... 26
Figure 6: A scenario for two randomly picked datasets ................................................................ 28
Figure 7: The general research method of the project ................................................................. 31
Figure 8: F1 scores for factor ‘number of recommendation candidate locations’ ...................... 35
Figure 9: F1 scores for factor 'number of locations in dataset' ................................................... 36
Figure 10: F1 scores for factor ‘number of reviews’ .................................................................... 37
Figure 11: F1 scores for factor ‘number of users’ ....................................................................... 37
Figure 12: F1 scores for factor ‘sparsity’ .................................................................................... 38
Figure 13: F1 scores for factor ‘average number of reviews per location’ ................................. 38
Figure 14: Curve fitting functions (F1) for factor ‘number of recommendation candidate
locations’ ................................................................................................................................. 40
Figure 15: Curve fitting functions (F1) for factor ‘number of locations in dataset’ .................. 40
Figure 16: Curve fitting functions (F1) for factor ‘number of reviews’ .................................... 41
Figure 17: Curve fitting functions (F1) for factor ‘number of users’ ........................................ 41
Figure 18: Curve fitting functions (F1) for factor ‘sparsity’ ........................................................ 42
Figure 19: Curve fitting functions (F1) for factor ‘average number of reviews per location’ .... 42
Figure 20: The principle trend observed in the experiments with conventional algorithms ...... 44
Figure 21: The functional principle of the switching hybrid recommender ............................... 45
Figure 22: The functional principle of the weighted hybrid recommender ............................... 46
Figure 23: F1 scores for factor ‘number of recommendation candidate locations’ including the
hybrid algorithms .................................................................................................................... 48
Figure 24: F1 scores for factor ‘number of locations in dataset’ including the hybrid algorithms .................................................................49

Figure 25: F1 scores for factor ‘number of reviews’ including the hybrid algorithms .................49

Figure 26: F1 scores for factor ‘number of users’ including the hybrid algorithms......................50

Figure 27: F1 scores for factor ‘sparsity’ including the hybrid algorithms.................................50

Figure 28: F1 scores for factor ‘average number of reviews per location’ including the hybrid algorithms ........................................................................................................51

Figure 29: The principle trend observed in the experiments with the hybrid algorithms.........52
List of Tables

Table 1: The recommendation algorithms under investigation ................................................... 30
Table 2: The runtime factors under investigation ........................................................................ 34
Abstract

Today, recommender systems are widely used in various domains. There are a lot of methods to generate recommendations and numerous parameters to adjust these methods. Each one of them has its individual strengths and weaknesses in certain situations. Based on the real-world use case of an existing location-based service application, the research at hand proves that by employing a custom-made hybrid recommender system, it is possible to exploit these strengths and, at the same time, to limit the weaknesses.

The project analyses some of the most popular recommendation algorithms with respect to their predictive accuracy on datasets with different characteristics. For this purpose, a suitable evaluation method was designed and implemented in the form of an experimental setup and protocol. Six runtime factors characterizing each dataset are identified and investigated. The results show that these runtime factors have a direct influence on the quality of the recommendations and that they affect different recommendation algorithms in different, sometimes oppositional ways. These results are used to explore whether the predictive accuracy of a superordinate hybrid recommender system in a dynamic environment can be improved, when compared with each single subordinate recommendation algorithm. This is achieved by dynamically selecting or weighting the results of the respective sub-algorithms in consideration of the current situation of use. For this purpose, two different hybrid recommendation algorithms were developed and analysed in direct comparison with the conventional algorithms. It is demonstrated that one of them achieves the most accurate results over the whole range of runtime factor values under investigation, thus effectively eluding the limitations of the specific sub-algorithms it utilizes. The second hybrid algorithm achieves relatively good results, but falls short of the author’s expectations to outperform all other methods including the first hybrid.
Chapter 1 Problem Overview

1.1 Introduction

The world in which we live is characterized by information. According to Google CEO Eric Schmidt, “every two days now we create as much information as we did from the dawn of civilization up until 2003 […] something like five exabytes of data” (Schmidt, 2010). If anything, this can be called ‘information overload’, a term that was coined by the American writer Alvin Toffler back in 1990, the same year that the idea of the ‘world wide web’ was proposed by Sir Tim Berners-Lee (Berners-Lee and Cailliau, 1990).

Today, the world wide web provides easy access to this enormous amount of data, including much more information on almost any given topic than a human user can process in reasonable time. Finding solutions to the information overload problem has become an extremely active field of research within the last two decades. Recommender systems promise one such solution by selecting pieces of information or items that the user is likely to find most interesting and useful.

The internet grows rapidly, and so do the potential fields of application for recommender systems. The present work investigates how some of the most commonly used algorithms for recommendation generation behave under different circumstances, and how recommender systems can be optimised for applications where these circumstances are frequently changing.

1.2 History of recommender systems

The first recommendation algorithms were introduced in the mid-1990s when researchers started to make explicit use of ratings provided by users (Hill et al., 1995; Resnick et al., 1994; Shardanand and Maes, 1995). These algorithms attempt to predict how beneficial the user would find a particular item, in other words how he would rate this item, based on the ratings of other users which are considered to be similar to the active user (i.e. the ‘neighbourhood’ of the active user) – a technique now referred to as collaborative filtering. Collaborative filtering algorithms have the advantage that they do not depend on knowledge about the nature of the items or the domain of the application (Ramezani et al., 2008). They try to compile a list of recommended items, but based only on the knowledge about relationships between users and items. In other words, they completely ignore the content of
the items (Montaner et al., 2003). This means that they work in the same way, no matter whether the items they deal with are movies, news articles or even people using a dating website.

Within the last two decades, many new algorithms and methods for recommendation generation have been developed, and each one of them has its individual strengths and weaknesses (Burke, 2002). Burke classifies modern recommendation techniques as ‘collaborative’, ‘content-based’, ‘demographic’, ‘utility-based’, ‘knowledge-based’ and ‘hybrid’ (ibid.).

The idea behind collaborative recommendation techniques was explained above. Collaborative filtering has become one of the most popular methods in recommender systems (Symeonidis et al., 2008). Content-based methods are defined by the features of the recommended items. They determine which kinds of items the user has preferred in the past and then recommend items that are similar to those; an approach that is necessarily domain-specific. Demographic recommendation techniques are based on demographic information about the users (e.g. data in user-profiles like sex, age, family status, etc.). Utility based systems use a utility function provided by the user, for example in the form of an interactive questionnaire (Burke, 2002). Knowledge-based recommenders utilize functional knowledge about items. This means that they reason about how a particular item meets a particular user’s needs. In the simplest case, the knowledge could be provided directly by the user in the form of a query, for example in the input field of a search engine (ibid.). The last category, hybrid recommenders, refers to systems that combine two or more algorithms or techniques in order to improve the performance of the recommender system as a whole.

Recommendation engines have been applied to many different domains, from e-commerce (e.g. by companies like Amazon, Netflix and Ebay) to internet services like Google news and Last.fm. For each of these domains, the characteristics of the data that can be used for recommendation generation as well as the requirements for the recommender system itself are different. As Herlocker et al. (2004) point out, the effectiveness of an algorithm depends very much on the application and on the characteristics of the underlying dataset.

Typically, recommendation engines operate on very large datasets, containing for example movies, songs or products in an online shop. These datasets often range into hundreds of thousands or even millions of items, thus scalability is a critical requirement for the recommendation algorithms. Methods that produce good results on small datasets are often not efficient enough for large datasets (Vozalis and Margaritis, 2003a). Another challenge that
recommender systems have to face is the sparsity of the dataset. This means that the percentage of items rated by a given user is typically very small on large datasets, which poses a problem to the process of neighbourhood formation (ibid.). Many solutions to these problems have been suggested, including hybrid recommendation algorithms like the ‘content-boosting’-approach of Melville et al. (2002) and the application of so-called ‘filterbots’ (which will be explained in some detail in section 3.4; see also Sarwar et al., 1998).

1.3 The problem domain

Besides recommender systems, a second aspect related to the present work has to do with location-based services (LBS). The international Open Geospatial Consortium (OGC, 2005) defines LBS as “wireless-IP services that use geographic information to serve a mobile user”. Recently, this domain gained more and more importance because mobile devices are now much more capable than they were a few years ago and mobile internet access has finally become widely available and affordable. However, up until now, there is not much research into how recommender systems can be designed and tuned to optimise their suitability for LBS. These applications usually have requirements and datasets that differ significantly from the traditional areas of application for recommender systems. For example, the relevant portion of the dataset is likely to be much smaller than the whole dataset, because only data in a given geographical area might be of interest. In some cases, this fact makes it possible to apply recommendation techniques that would not be efficient enough for other domains. Current solutions to problems like scalability or dataset sparsity usually focus on traditional applications and very large datasets like the MovieLens- (GroupLens, 2006), Netflix- (Netflix, 2009) or the Jester Joke Rating-dataset (Goldberg, 2003).

Another important point to consider is the fact that in most applications, the situation in which recommendation generation takes place is rather static – the system computes predictions for every request based on the whole item- and user-space, or at least a very large part of it (e.g. all products that are on stock in an online shop). In contrast to that, in an LBS application the situation can change from request to request, depending on the geographical area for which recommendations are requested. This affects the respective sets of items that have to be taken into account. For this reason, LBS applications are a perfect example for the kind of dynamic environments investigated in the course of the present work. This issue will be explained in some more detail in section 1.4.
Regionrex.com, the application that provides the problem domain for the present dissertation project is a LBS that generates recommendations for places of interest like restaurants, bars or clubs based on the geographical position of the user. It is a map-based application that currently exists in a version that can be used from any web browser as well as a native version running on smartphones. The system determines the user’s current geographical position by means of the mobile phone’s ‘Global Positioning System’ (GPS) module (GPS, 2010) or by using the HTML5 ‘Geolocation API’ of the browser (W3C, 2010). As an alternative, the user can manually enter an address or just select his current position on the map. The user interface of the system provides control elements that allow the user to enter the maximum geographical distance that he is willing to travel to a location. The system will only recommend locations within that distance (depicted by the blue circle in Figure 1). The following section will show that this fact makes it a perfect use case for the present research.

![Figure 1: Web user interface of regionrex.com](image)

It can be concluded that location based services and applications such as regionrex.com open up new challenges and opportunities for recommender systems, and that the optimised
application of traditional techniques and algorithms in this domain forms a field of research that still contains many open problems and requires further investigation and improvement.

1.4 Aims of the research

LBS systems set the stage for the potentially successful application of several well known recommendation techniques and algorithms. In the context of this project, regionrex.com can be regarded as one example of such systems. As explained above, this recommendation engine only considers locations within a certain geographical distance of the current user as candidates for recommendation. This means that it uses the location as a so-called ‘hard criterion’ to narrow the set of items before recommendation generation takes place. The exact definition of the term ‘hard criterion’ in this context will be explained in section 2.6, but for now it is sufficient to know that only a small subset of all the locations will be taken into account. The characteristics of this subset are unknown at design time of the system, but rather depend on the temporary context of use. This fact makes the system a good case of application for the present research.

The following scenario illustrates the functional principle of regionrex.com:

Alice is a regular user of regionrex.com. She has created a user profile that expresses which music she likes and which kinds of food she prefers. Her profile also contains personal information like her age and sex. Alice has already entered many reviews for the restaurants and bars in her hometown into the system. The locations she has reviewed have also been reviewed by others, so there is a neighbourhood of users that potentially have the same taste as Alice.

She logs into regionrex.com. The system recognizes that she is in her hometown and retrieves a list of locations in her geographical area. In order to recommend some of these items, the recommendation engine can use the results of every recommender module that is implemented, or just pick the one that is known to produce the best results under these circumstances. The system computes a list of recommendations and presents it. Alice selects a bar a few blocks from her house that she doesn’t yet know. She goes out and has a wonderful evening.

A week later, she is on a business trip to a foreign city. She is looking for a restaurant there and logs into regionrex.com. This time, the system cannot form a neighbourhood of similar
users based on past reviews, because Alice has never rated a location in this city, thus there are no reviews that “overlap” with the reviews of other users. However, the system can still form a user neighbourhood based on Alice’s user profile and implicit knowledge about her taste derived from her reviews for locations in her hometown.

Bob is a new user. He has not created a user profile, nor has he written any reviews. In this case, the system has to rely on rules like “men of Bob’s age tend to like this location”, or just recommend the places that get high ratings overall.

This scenario shows that the situation from the point of view of the system can be a different one for each request. It depends on the geographical position of the user and the maximum allowed distance from this position to a potentially recommendable location (i.e. the current value of the “maximal travel distance” option \(d\), explained below). The temporary sub-dataset that the recommendation engine can use for recommendation generation might be very different, according to whether the user is located in a big city with a lot of restaurants and bars, or in a small village. The following example shows that there are various factors which characterize this sub-dataset:

\[ P_{u1} \] is the geographical position of the “current user” \(u1\) (i.e. the user for whom recommendations are generated), and \(d\) is the maximum distance he is willing to travel to a location. \(R_{u1}\) shows the locations for which \(u1\) has entered reviews: \(R_{u1} = \{l4, l7\}\). Likewise, \(R_{u2}\) shows the locations

\[ R_{u2} \]

\[ R_{u3} \]

\[ P_{u1} \]

\[ d \]

\[ d \]
for which the user $u_2$ has entered reviews ($R_{u_2} = \{l_4, l_6, l_7, l_8\}$) and $R_{u_3}$ shows the same for user $u_3$ ($R_{u_3} = \{l_3, l_4, l_5, l_6, l_9\}$).

The locations $\{l_2, l_3, l_4, l_5, l_6\}$ could be candidates for recommendation, since all the other locations are too far away from the current user. However, the recommendation engine cannot evaluate location $l_2$, because none of the users has expressed an opinion about it.

The engine can for example use the set of corresponding or overlapping reviews between users $u_1$ and $u_2$, $\{l_4, l_7\}$, to determine the similarity between these two users. Likewise, the set of locations $\{l_4, l_6\}$ can be used to determine how similar are the tastes of $u_2$ and $u_3$. This information can then be used to predict how $u_1$ would like the locations in the subset of potentially recommendable locations $I_{rec} = \{l_3, l_5, l_6\}$. Depending on the requirements of the application, location $l_4$ could also be part of this subset. However, because the current user already knows this location and has reported an opinion about it, the location is usually excluded from the recommendations.

The dataset that contains relevant information for recommendation generation in the form of overlapping reviews is $I_{rel} = \{l_4, l_6, l_7\}$. The datasets $I_{rec}$ and $I_{rel}$ have certain characteristics, for example the respective number of items per dataset, which depend on the surrounding circumstances. These characteristics are referred to as ‘runtime factors’, because in a dynamic environment, they will be different for each user and request and can only be determined at runtime.

The hypothesis under investigation in this research project is that these runtime factors, which will be described in more detail in Chapter 3, have a direct influence on the quality of the recommendations that the system can produce. Presumably, each recommendation algorithm is affected in a different way from other recommendation algorithms. If this is the case, then a hybrid system that dynamically selects or combines different algorithms depending on the characteristics of the relevant sub-dataset has the potential to produce better results over the whole range of runtime factor values, than a system using only a single recommendation technique or a static combination of algorithms.

This hypothesis leads to the research question:

“Can the overall predictive accuracy of a recommender system be improved by dynamically selecting or combining different recommendation algorithms depending on the current situation of use?”
In this context, predictive accuracy means “how precise is the system’s predicted rating compared with the user’s actual rating?”. This can be measured in a purely objective and scientific way, which will be explained in detail in section 3.1 of this document. Predictive accuracy can be regarded as an equivalent of recommendation quality. The better the system is at predicting user ratings, the higher will be the quality of the recommendations.

The term ‘current situation of use’ refers to the respective runtime factor values as explained above.

Since the problem domain for the present research is an LBS, the items to recommend are different locations. In this case they can be restaurants, bars, clubs, etc. Consequently, the term ‘location’ denotes a special kind of an ‘item’. In the rest of the document, both terms will be used depending on the context.

**1.5 Contribution to knowledge**

Most of the research on recommender systems focuses on the typical fields of application: areas where huge amounts of relatively static data have to be taken into account. The application of recommendation algorithms to domains with very dynamic content is not fully investigated. The results of the present work ought to be of interest to researchers and practitioners trying to adapt recommendation techniques to these domains where the suitability of a particular algorithm cannot be generally determined in advance. This is probably the case for a high percentage of LBS applications, because the geographical distance between the current user and the potential items of interest is likely to change in a rather unpredictable way - at least from the point of view of the application. Most of the items will usually be too far away to be relevant. The results can also be generalised to other domains where recommendations have to be generated under rapidly changing circumstances. Modern online shopping sites are one example. In this case, the product category of required items will define the relevant subset of items that the application will recommend.

Web-based applications employing recommendation engines often operate in domains where the ability to gather content used to be rather limited. For example, online shops usually had to rely on a single source of information about the products they were selling: content provided by the manufacturer. In addition to that, the possibilities to generate detailed customer profiles used to be somewhat restricted, because privacy was an important concern for many users of these web applications. Recently, the advent of so-called ‘Web 2.0’
applications changed this picture. These applications heavily rely on content provided by the user. Many online shops now offer their customers the ability to rate and describe products. This development contributes to the fact that many recommender systems now have much richer datasets to work with, but it also leads to a situation where effective recommender systems can no longer treat all users in the same way with regard to recommendation generation. Some users will eagerly participate in the process of reviewing and rating products while others will rarely do so. Thus, the system has to decide which algorithm is most suitable based on the available information about the current user and the users in his neighbourhood. This is another example where dynamically selecting or combining algorithms at runtime could be advantageous.

The present research project introduces a new approach to optimising this kind of hybrid recommender systems by analysing the subset of items that is relevant for recommendation generation for each particular request. Consequently, the recommendation engine will be able to adapt to various situations as they occur at runtime of the system. The research will be conducted in the context a specific application (regionrex.com), but it is easily possible to apply the exact same generic technique to other applications and domains. In this aspect, the approach differs from others (e.g. the one followed by van Setten et al. (2004)) which use domain-specific semantics to predefine strategies for selecting or combining different recommendation methods.

To the best of the author’s knowledge, no other research project or productive system takes this particular aspect of recommendation generation into account. Consequently, some novel methods for designing and evaluating such a system had to be developed. The evaluation method employed will be described in more detail in Chapter 3 of this document.

The ‘Web 2.0’ trend is expected to continue and the need for recommender systems that can deal with the resulting requirements and inhomogeneous datasets will grow even further in the future. The limits of what internet applications can do are pushed further and further, and there is a growing number of web software developers wishing to adapt recommender systems to new application domains in this area. Consequently, the present work should be of interest to a broad audience of software engineers and computer scientists.
1.6 Summary

Recommender systems help users in finding items of interest from a potentially overwhelming set of choices. Different techniques to achieve this goal were developed within the last 20 years. However, recently a lot of new potential fields of application for recommender systems appeared, including location-based services and other very dynamic environments. These new use cases and applications such as regionrex.com open up new challenges and opportunities for recommendation engines. The present work introduces an approach for the optimised application of traditional techniques and algorithms in this context.

The rest of the document at hand is organized as follows: Chapter 2 provides an overview of the past research regarding recommender systems. Chapter 3 describes and justifies the research methods that have been used. The results of the experimental evaluation of individual algorithms are presented in Chapter 4. Chapter 5 introduces the hybrid recommendation algorithm prototypes and explains how they work while Chapter 6 provides an in-depth analysis of the results. Chapter 7 summarizes the project, draws conclusions and gives recommendations for further research.
Chapter 2  Fundamentals of recommender systems

Recommender systems are a very active field of research, and today a wide variety of techniques for generating recommendations exists. The following chapter surveys the relevant literature in this field and introduces the basic principles behind some of the most popular recommendation algorithms.

2.1 Classification of recommendation techniques

The main motivation behind the development of recommender systems is to solve the “information overload problem” (Toffler, 1990). It seems that this problem is perceived as a very severe one by many people, because in the last 20 years, a lot of information retrieval, data mining and recommendation techniques and algorithms were developed. In fact, today, there are so many different approaches that one can easily be overwhelmed by all the different options and combinations, when a new recommender system is to be planned and implemented. Classifying these different approaches can help to select the best solution for a given domain and application.

Burke (2002) divides algorithms into ‘collaborative’, ‘content-based’, ‘demographic’, ‘utility-based’, ‘knowledge-based’ and ‘hybrid’ (see section 1.2). Unfortunately, however, this clear distinction is not always possible and not everyone agrees which category an algorithm might fit in. One of the first collaborative filtering systems, Tapestry, which was developed in 1992 (Goldberg et al., 1992) can be considered as an example. The system was supposed to solve the electronic mail overload problem by filtering messages and newsgroup articles, so that only relevant items would be presented to the user. Human participants collaborated in this filtering process by annotating electronic documents, so the authors coined the term ‘collaborative filtering’ (CF) (Deshpande and Karypis, 2004). Tapestry required the users to explicitly provide filtering queries in form of the so-called ‘Tapestry Query Language’ (Goldberg et al., 1992). Consequently, following Burke’s classification, this system should be assigned to the family of knowledge-based or utility-based recommenders rather than CF. Modern CF systems, on the other hand, are usually quite transparent for the user and just utilize the relationships between users and items to determine similarities “behind the scenes”. The first systems that actually used techniques that can be regarded as CF in the sense that Burke (2002) specifies came up two years later in 1994. Among these is the GroupLens system (Resnick et al., 1994) as well as Fab (Balabanović and Shoham, 1997). Although
details about Fab were not published until 1997, the system was already operational for 3 years at that time (ibid.).

CF algorithms can again be classified as model-based or memory-based (Breese et al., 1998). The former generate recommendations based on a model of user ratings that has to be calculated and saved in advance, while the latter determine the nearest neighbours of the current user at runtime. (Symeonidis et al., 2008). This means that they require the whole rating-matrix to be held in the computer system’s random access memory (RAM) (Brozovsky and Petricek, 2007), thus the name ‘memory-based’. In this context, the nearest neighbours are the users that are most similar to the current user by some measure (details about different similarity measures will be explained in section 2.2). The algorithm then uses these users’ ratings to determine the list of recommendable items for the current user. Consequently, these algorithms are also known as ‘nearest-neighbour algorithms’ (ibid.).

Another differentiating factor is whether an algorithm is user-based or item-based. In general, user-based algorithms try to determine the items the active user would probably like, based on information of those items that people with comparable taste preferred in the past. In other words, they calculate similarities between users. In contrast to that, item-based algorithms look at the items the active user reported to like, and then try to find items that are similar to those. Of course, the similarity between items is also calculated based on the ratings given by other users (Sarwar et al., 2001).

There seems to be some disagreement whether item-based algorithms can be regarded as model-based or memory-based. Although Symeonidis et al. (2008) allege that item-based algorithms are always memory-based, other authors classify them as model-based (e.g. Brozovsky and Petricek, 2007). It can be concluded that the distinction cannot be made so easily. It depends on the implementation and context of use. In many applications, the relationships between items do not change as frequently as the features of users and the relationships between them (Sarwar et al., 2001). This fact makes it possible to calculate and store the similarities between items in advance. Some argue that the process of recommendation generation could then be regarded as model-based, because it does not require that all the data about users, items and ratings is present in the computer’s memory. On the other hand, the process of similarity calculation can still be regarded as memory-based. It is just not conducted online, but possibly over night and on a different machine.

As these examples show, classifying the enormous variety of recommendation techniques that exist today is not an easy task. There seems to be an almost unlimited number of options how
recommender systems can be implemented and not everyone agrees on the differentiating factors of algorithms.

### 2.2 Similarity metrics

Many CF algorithms use some sort of ‘similarity metric’ or ‘similarity measure’, based on a logical matrix of users, items and ratings (Resnick et al., 1994). This matrix can be regarded as a kind of user profile, where each cell of the matrix contains historical data representing the evaluation of a single item by a single user (Montaner et al., 2003). Depending on the context, similarity metrics are sometimes also referred to as ‘correlation metrics’ or ‘distance metrics’ (Adomavicius et al., 2005; Segaran, 2007). In general, they are methods to calculate some kind of score that expresses how similar users or items are with respect to each other. These scores can then be used as the foundation of user- or item-based recommendation generation.

The simplest method to determine a similarity score is to calculate the standard deviation (Wikipedia, 2010a) between all the user’s ratings and to normalize it so that a score in a predefined range, for example from 0 to 1, is achieved. In some applications, this very simple method might lead to acceptable results. In others, a more sophisticated metric is needed. A related, but slightly more elaborate method uses the Euclidian distance measure (Segaran, 2007). The idea can be visualised by regarding users or items as points in a multi-dimensional space. For example, consider an application with five users $u_1$...$u_5$ and two items. In this application, users can rate items on a scale from 0 to 5, so the relationship between users, items and ratings can be depicted like this:

![Figure 3: A graphical representation of users, items and ratings](image)

The chart shows a graphical representation of the ratings $r_1$ and $r_2$ that the users $u_1$...$u_5$ gave for the items $i_1$ and $i_2$, in other words, it shows their preferences for those items (Segaran,
The distance between the users on this chart shows how similar their taste is regarding the items $i_1$ and $i_2$. To determine the similarity score between two given users (e.g. $u_3$ and $u_5$), the distance $d$ between them can be calculated (Padilla, 2008) like this:

$$d = \sqrt{(r_1(u5) - r_1(u3))^2 + (r_2(u5) - r_2(u3))^2}$$

For the general case of $n$ rated items per pair of users this can be written as:

$$d = \sqrt{\sum_{i=1}^{n} (r_i(u5) - r_i(u3))^2}$$

A similarity score between 0 and 1 can then be achieved by normalizing the result of this formula by 5, which is the maximal possible distance on the rating scale in this example, and then subtracting it from 1. A higher degree of similarity corresponds to a higher score, and a score of 1 means that two users have identical preferences.

An alternative similarity metric is called the Pearson correlation coefficient (Shardanand and Maes, 1995; Vozalis and Margaritis, 2003a). For two users $u_3$ and $u_5$, it would be calculated like this:

$$\frac{\sum_{i=1}^{n} (r_i(u5) - \bar{r}(u5))(r_i(u3) - \bar{r}(u3))}{\sqrt{\sum_{i=1}^{n} (r_i(u5) - \bar{r}(u5))^2 \sum_{i=1}^{n} (r_i(u3) - \bar{r}(u3))^2}}$$

The resulting score ranges from -1 to 1, where -1 indicates a negative correlation, 0 indicates no correlation and 1 indicates perfect similarity. The Pearson correlation coefficient often leads to better results than the aforementioned metrics in cases where some users tend to always give better or worse ratings than the average (Segaran, 2001).

There are many other similarity metrics that can be used to implement recommender systems, like for example ‘Manhattan distance’, (Adomavicius et al., 2005; Black, 2006), ‘Cosine/Vector similarity’ (Su and Khoshgoftaar, 2009; Vozalis and Margaritis, 2003a) or the ‘Tanimoto coefficient’ also known as ‘extended Jaccard coefficient’ (Strehl and Ghosh, 2000; Wikipedia, 2010b). As Adomavicius et al. (2005) point out, “the choice of a specific distance
metric depends largely on a specific application domain”. A metric that works fine for a given dataset with specific features might not be suitable for a different dataset.

### 2.3 Alternative recommendation algorithms

Not all methods for recommendation generation are based on similarity metrics. One alternative algorithm is called ‘Slope One’ (Lemire and Maclachlan, 2005). This algorithm is based on a technique that Lemire and Maclachlan (2005) call ‘popularity differential’. The algorithm looks at pairs of items and their ratings. It first calculates the average difference between the items’ ratings for each pair on a dataset. This data can then be used to predict ratings for items that have not been rated by a given user. As a simple example, consider an application with a rating scale from 0 to 5 and with three items \( i_1 \ldots i_3 \). The average of all the users’ ratings for \( i_1 \), \( \bar{r}_{i1} \) is 2, \( \bar{r}_{i2} \) for \( i_2 \) as well as \( \bar{r}_{i3} \) for \( i_3 \) is 3. Assume that the system is supposed to predict a rating for item \( i_3 \) for a user \( u \) who has already rated \( i_1 \) with 2 (\( r_{ui1} \)) and \( i_2 \) with 4 (\( r_{ui2} \)). Looking at \( i_1 \), the Slope One algorithm would predict a rating of \( r_{ui3} + (\bar{r}_{i3} - \bar{r}_{i1}) = 2 + 3 - 2 = 3 \), where \( (\bar{r}_{i3} - \bar{r}_{i1}) \) represents the first slope. Looking at \( i_2 \), the algorithm would predict \( r_{ui2} + (\bar{r}_{i3} - \bar{r}_{i2}) = 4 + 3 - 3 = 4 \) with the help of the second slope \( (\bar{r}_{i3} - \bar{r}_{i2}) \). Consequently, the final prediction would be 3.5, which is the average of all the intermediate predictions. Figure 4 visualises this example.

![Figure 4](image)

**Figure 4: A graphical representation of the Slope One example**

Another recommendation method is referred to as ‘clustering’ (Ungar and Foster, 1998). Cluster-based recommenders first use some sort of similarity metric to form groups of likeminded users. In a second processing step, the items that the majority of users in a
particular cluster like best are determined and recommended to other users within the same cluster (Montaner et al., 2003).

2.4 Advantages and disadvantages of individual algorithms

The wide variety of recommendation algorithms implicates that there is always a choice how the goal of recommending the most interesting items can be achieved. Every single algorithm has its individual advantages and disadvantages as compared with other algorithms in regard to a particular use case. For example, collaborative, content-based and demographic recommender systems all suffer, in one way or another, from the so-called ‘cold start problem’ (Burke, 2007). This means that the system is unable to make useful recommendations until the target user has provided sufficient data about what he likes or dislikes. This is easily understandable, given the fact that these systems have to rely on information about the user and his personal preferences.

Besides the cold start problem, Rao and Talwar (2008) list several other individual advantages and disadvantages of recommendation techniques. For example, they argue that CF systems exhibit the so-called ‘Critical Mass problem’: They need a relatively large number of users and ratings before they can produce good results. According to them, the predictive accuracy of CF increases with the available amount of data.

‘Overspecialisation’ on the other hand is a problem of content-based recommender systems (Adomavicius and Tuzhihin, 2005). This means that the system cannot recommend items different from those that the user already knows about. If a user has never expressed an opinion about classical music, an overspecialised music recommender system could never recommend any classical titles to this user. CF systems do not suffer from this problem.

Karypis (2001) compares item-based and user-based algorithms with respect to their individual advantages and disadvantages. He claims that the former produce significantly better results than the latter. On the other hand, Vozalis and Margaritis (2003b) state that they were not able to reproduce these results and that they found that user-based methods produce recommendations that are more accurate. It should be noted that Vozalis and Margaritis conducted their experiments with one of the original GroupLens datasets consisting of 10 000 ratings. Karypis used five different datasets, which also included one of the GroupLens datasets. However, it was not the same dataset as used by Vozalis and Margaritis. This dataset included 100 000 entries and, unlike Vozalis and Margaritis, Karypis ignored the actual
ratings. The experiments that showed dramatically better results for item-based algorithms were conducted with yet another dataset, which consisted of credit card transactions. It can again be concluded that the predictive accuracy of an algorithm significantly depends on the characteristics of the dataset.

In summary, it should be noted that there are some general advantages and disadvantages of different recommendation techniques. Other effects are specific to a given application and dataset and cannot be generalised. Some algorithms are simply more effective in a given situations than others. For example, a method might lead to better results than others when the dataset is very sparse, but might on the other hand suffer from overspecialisation.

### 2.5 Hybrid systems

It is possible to combine different recommendation methods in order to compensate the disadvantages of each individual method, while at the same time utilizing its benefits (Burke, 2007). Balabanović and Shoham (1997) claim that their hybrid recommender system, a combination of CF and content-based algorithms, displays qualities which are superior as compared to a system using only a single recommendation technique. Pazzani (1999) also argues that the strengths of the collaborative and the content-based approach are complementary. Various strategies for bringing these two ideas together have been extensively addressed in the past (e.g. Kim et al., 2006; Basilico and Hofmann, 2004). Typically, such methods are used to improve recommendation quality with respect to sparse datasets or to avoid the cold start problem. They often aim at reducing the sparsity of the whole dataset by adding synthetic ratings that are calculated in an offline processing step before recommendations are generated. Sarwar et al. (1998) and Melville et al. (2002) give other examples. The combinations suggested in the literature also include mixing item-based and user-based methods (Vozalis and Margaritis, 2004) as well as memory-based and model-based algorithms (Gong et al., 2009).

Burke (2002) proposes seven different design strategies for hybrid recommender systems. He defines ‘switching’ hybrids as algorithms that choose between the available recommendation techniques, based on some switching criterion. For example, the system proposed by Tran and Cohen (2000) switches between CF and a knowledge-based recommender depending on the number of users for which the system knows their individual preference profiles and the number of rated items in the database.
On the other hand, in cases where it is feasible to run multiple recommendation engines simultaneously, the results of these engines can be combined in some way. ‘Weighted’ hybrids use the prediction scores of different recommendation methods in order to produce a unified prediction score. The system can thereby assign a different weight to the results of each single algorithm. Burke (2002) claims that Claypool et al. (1999) follow this approach to integrate a content-based and a collaborative recommendation engine into a single system. However, Claypool et al. (1999) themselves argue that their method is not a hybrid approach, because the basis for the particular recommender modules is kept separate. They just use the weighted average of the predictions of both modules.

Another option is to present the recommendations generated by multiple methods together or next to each other. Burke (2002) refers to this approach as ‘mixed’ hybrid. In his movie recommender application “PickAFlick” (Burke, 2000) he uses this technique to produce several sets of recommendations which are then proposed to the user at the same time. In contrast to that, ‘cascade’ hybrids use multiple algorithms in sequence. One advantage of this approach is that the system does not consume so much computation time, because not every algorithm has to analyse the whole dataset.

Burke (2002) also suggests ‘feature augmentation’ and ‘meta-level’ hybrids. These methods use models or input features from one algorithm as input to another algorithm. The last category is referred to as ‘feature combination’ hybrid. This approach treats collaborative information as feature data that can be regarded as an enhancement of the dataset. Content-based algorithms are then used to produce recommendation from this dataset. Such a system does not depend only on relationships between users or items. As a result, it can even produce acceptable results on sparse datasets.

As Rao and Talwar (2008) point out, hybrid recommender systems should be regarded as state-of-the-art; they are going to play an even more important role in the next generation of recommendation engines. The full potential of the huge variety of algorithms that are available today can only be utilized by combining different approaches, thus exploiting the benefits of each single one of them.
2.6 Recommender systems and location based services

Virrantaus et al. (2002) define location-based services (LBS) as “services accessible with a mobile device through the mobile network and utilizing the ability to make use of the location of the terminals”. These kinds of services are rapidly evolving due to the increasing maturity of mobile devices and the availability of third generation mobile data connections (3G). In order to make these services useful, special needs of the mobile user for information and usability have to be met.

LBS applications differ from conventional media because they are aware of the location of the user as part of the context in which they are being used (Steiniger et al., 2006). They can adapt and filter their contents accordingly in order to provide the user with relevant information. In this aspect, they are similar to recommender systems, which also aim at helping the user to find his way quickly and easily through large amounts of data (van Setten et al., 2004). Various approaches to integrating recommendation functionality into LBS applications have been developed in the past. These approaches are differentiated by whether they use the location as a hard or soft criterion in the recommendation process (ibid.). In the former case, the system discards items that do not match the criterion before recommendation generation takes place. In the latter case, the location is treated as a feature of the item that directly influences the predicted relevance. The system developed by van Setten et al. (2004) is an example of the former category, because items that are too far away from the user are filtered out from the dataset that is used by the recommendation engine. The same applies to the method presented by Yu et al. (2009) and also to the regionrex.com-recommender investigated in the present research.

In contrast to that, the approach followed by Brunato and Battiti (2003) falls in the latter category. Their system calculates a virtual ellipsoid from the geographical positions of users showing an interest in a particular set of items. Recommendations are then generated solely based on the geographical distance of the current user from the centre of this ellipsoid.

McCarthy (2002) presents another method which uses the location as a soft criterion. He calculates preferences as a sum of four different criteria scores, each one multiplied with a priority value. The distance between the current user and a location is one of these criteria, thus it contributes to the final score on which any recommendations are based.
2.7 Summary

There is a multitude of ways in which recommendation techniques can be classified. Besides the category of the algorithm (CF, content-based, demographic, etc...), the most relevant distinguishing criteria are whether an algorithm is model-based or memory-based and whether it works with users or items. Most recommendation algorithms use some kind of similarity metric, but there are also alternative methods that do not need a notion of similarity (e.g. the Slope One algorithm). Altogether, there are many different methods to build recommender systems and each strategy has its own individual advantages and disadvantages. Hybrid systems often achieve better results than systems using only a single recommendation technique, because they can compensate the shortcomings of individual algorithms.

In the context of location-based services, recommender systems can be differentiated on whether they use the location as a hard criterion to filter items before the recommendations are generated or whether they use it as a soft criterion that has an influence on the recommendation algorithm itself.
Chapter 3  Research Methodology

Because the present research is concerned with the predictive accuracy of various recommendation algorithms in different situations, empirical analysis in the form of controlled experiments was chosen as the predominant research method. In the context of this project, several recommendation algorithms as well as an environment for experimental setups to evaluate these algorithms against sub-datasets with varying characteristics were implemented. This will be explained in detail in the following chapter.

3.1 Evaluation metrics

This project uses controlled experiments to obtain the quantitative data from which well-established statistical methods, referred to as ‘evaluation metrics’ (Zaier et al., 2008), derive any evidence.

In general, evaluation metrics are used to determine the accuracy of an algorithm by having it generate recommendations on items for which the ratings are already known (Herlocker et al., 2004). The data is divided into two independent sets, the ‘training set’ and the ‘test set’. The algorithm under investigation uses only the data in the training set to learn about relationships and preferences. Then it tries to produce recommendations for the data in the test set. Since the actual ratings are known, the accuracy of these recommendations can be evaluated with respect to a certain evaluation metric. These metrics are classified as either ‘statistical accuracy metrics’ or ‘decision support accuracy metrics’ (Sarwar et al., 2001).

Statistical accuracy metrics compare the predicted ratings calculated by the recommender system based on the set of training data to the ratings actually given by the users. They are also known as ‘predictive accuracy metrics’ (Herlocker et al., 2004). One of the most frequently used methods is to calculate the average absolute difference between predicted and actual ratings (Herlocker, 2000). This metric is called ‘mean absolute error’ (MAE) and has been used by many researchers in the field, including Sarwar et al. (2001) and Herlocker et al. (1999). Lee et al. (2006) used MAE to evaluate a LBS recommender system for restaurant recommendations. An alternative method, the ‘root mean squared error’ (RMSE) (Sarwar et al., 2001), calculates the square root of the average difference between actual and predicted ratings.
In contrast to statistical metrics, decision support accuracy metrics are concerned solely with how effective a recommender system is in predicting whether a user would classify an item as good or bad, or in other words, as relevant or irrelevant. They do not take into account the exact predicted rating. Consequently, these metrics are also referred to as ‘classification accuracy metrics’ (Herlocker et al., 2004). The most popular metric for information retrieval systems, ‘precision and recall’ (PR) falls into this category (ibid.). PR is also used to evaluate recommender systems (Basu et al., 1998, Billsus and Pazzani, 1998; Karypis, 2001) including LBS recommenders (Kuo et al., 2009, Yu et al., 2006).

In the context of PR, precision is the fraction of recommended items that is actually relevant to the user.

\[
\text{precision} = \frac{|\{\text{relevant items}\} \cap \{\text{recommended items}\}|}{|\{\text{recommended items}\}|}
\]

Usually not all items are considered, but the precision is rather calculated based on only the top-\(n\) recommended items. The metric is then referred to as ‘precision at \(n\)’ or ‘P@\(n\)’ (Wikipedia, 2010c).

Recall is defined as the fraction of relevant items that are also part of the set of recommended items.

\[
\text{recall} = \frac{|\{\text{relevant items}\} \cap \{\text{recommended items}\}|}{|\{\text{relevant items}\}|}
\]

A related metric that combines precision and recall into a single number in order to make the results easily comparable is referred to as ‘F1’.

\[
F1 = 2 \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}
\]


The suitability of each metric depends on the features of the dataset and the kind of tasks supported by the recommender system (Herlocker et al., 2004).
3.2 Evaluation strategy

As far as the author is aware, the particular aspects of recommender systems investigated in this project, specifically how certain characteristics of relevant sub-datasets which are dynamically determined at runtime might affect different recommendation algorithms, has not yet been the target of extensive scientific research. Therefore, a suitable evaluation strategy and an experimental environment had to be specially contrived and developed for the purpose of this project.

Recommender systems are commonly evaluated with a simple method that employs one of the metrics explained in section 3.1. A certain percentage of the reviews are withheld from the algorithm, which then tries to predict these ratings (Lemire and Maclachlan, 2005). The predictive accuracy of the algorithm is determined by testing it against the whole dataset of the application. In order to mitigate the influence of exactly how the data happened to get randomly divided, the test may be repeated multiple times, each time using different partitions of the data as training set and test set - a process that is known as ‘k-fold cross validation’ (Schneider, 1997). This method generally examines the whole set of users, items and ratings in the database.

However, this simple evaluation approach would be insufficient to investigate whether the hypothesis that the predictive accuracy of a recommender system can be improved by a dynamic selection or combination of different recommendation algorithms is true, because the recommendation engine has to be evaluated against various sub-datasets rather than the whole dataset. The system under investigation uses the geographical location as a hard criterion (see section 2.6) for filtering items before recommendation generation takes place. For this reason, simply using the location as a training feature is not possible. A new evaluation method and protocol that allows for fine-grained control of the characteristic features of temporary sub-datasets had to be developed. This approach uses the same well-established accuracy metrics (MAE and PR) and the same basic evaluation method (k-fold cross validation), but it performs a great many evaluation runs instead of just one, each time using a different sub-dataset. The final result is computed from all the sub-results that are relevant for the investigation of a single algorithm and a single characteristic factor. The following section explains the details of this procedure.
3.3 Setup of the experiments

The recommendation algorithms under investigation are intended to be used in a system where only a small subset of the whole dataset can be used to generate recommendations. This subset is likely to be very different for every request, and its characteristics are unknown at design time. An experimental setup which simulates different situations in a controlled environment was used to benchmark the predictive accuracy of the various algorithms under these circumstances.

The hypothesis is that the characteristics of the relevant sub-dataset have an influence on the quality of the recommendations generated by each single algorithm. If this were to be the case, then combining different recommendation techniques in an optimised way with respect to the characteristics of the sub-dataset can lead to an improvement of the predictive accuracy of the whole system. The validity of this hypothesis can be tested by investigating the following characteristic runtime factors:

a) The number of items which are potential candidates for recommendation because they are within a certain distance from the current user
b) The number of items in the sub-dataset which is used for recommendation generation
c) The number of reviews (i.e. reviewed items) that can be used to determine similarities
d) The number of users which have written a review for one of the recommendation-candidate-locations
e) The sparsity of the sub-dataset (i.e. the reciprocal value of the average percentage of items rated per user)
f) The average number of reviews per item

It was important that the experimental setup allowed control over each one of these factors, while at the same time ensuring that the other factors did not affect the result of the experiment. Ideally, the influence of each factor should be tested in isolation. This could not be achieved with a single experiment, because there would be no way to rule out the influence of the other factors. Consequently, multiple experiments were conducted, each time controlling the value of a single factor at a time, while at the same time allowing the values of the other factors to be random. The final evaluation score could then be calculated from all the evaluation results for each single factor. Since it could not be assumed that the factors are independent, enough experiments for each value of each factor under consideration had to be conducted in order to make sure that the effects of the other factors could be averaged out.
The fact that the relevant sub-dataset is unknown at design-time means that it can be regarded as random from the point of view of the recommendation engine. As a result, quasi-random sub-datasets could be used for the experiments. These sub-datasets would be selected in a way that allowed controlling a single factor at a time, while the values of all the other factors were actually random.

The experimental environment was implemented in the Java programming language and consists of three main components: ‘DataManager’, ‘Evaluator’ and ‘Analyzer’.

The ‘DataManager’ has a connection to the main database that contains all the locations, users and reviews. It builds a ‘Dataset’ object that represents a model of users and their ratings for a given number of locations. This object also contains the actual values of the factors ‘a’ to ‘e’ detailed above, for this particular dataset.

The ‘Evaluator’ conducts the actual evaluation and has references to all the recommendation algorithm implementations as well as all the evaluation metric implementations. It iterates over all the values of a factor to investigate. For each value, it uses the ‘DataManager’ as a service to get a matching sub-dataset from the whole database. Then it evaluates each algorithm against this dataset, using all the implemented evaluation metrics. The results are saved in an ‘EvaluationResult’ object and stored in a separate database.

In a second processing step, the ‘Analyzer’ consolidates and analyses all the evaluation results for each single factor. It iterates over each factor and each value or range of values under consideration, retrieving all the evaluation results that were generated with a dataset that exhibits the respective characteristics. It then calculates the average of the results for each single evaluation metric, produces the final statistics and charts and saves them to the file system.
Figure 5 depicts the two processing steps of the experiment.

Figure 5: Experimental setup

3.4 Database preparation

During the beta testing phase of regionrex.com, a large database with users, items and reviews has been created. At first, the recommendation algorithms under investigation were evaluated on this natural dataset. However, due to the nature of the controlled experiments outlined in section 3.3, additional experiments with an even larger dataset proved to be mandatory in order to avoid bias. For these experiments, it had to be guaranteed that the random sub-datasets, which were taken from the whole dataset for each single experiment, were for the most part truly independent. Consequently, a second database was prepared for the purpose of the experiments. This enlarged dataset has the same characteristics as the original dataset with respect to the specific ratios of users, items and reviews. It does not consist of purely synthetic data, but can rather be regarded as a version of the original, natural dataset which was basically just increased in terms of its size. The following example explains the reasons for this database preparation process in detail:

Assume that the original dataset contains 150 locations, and that a series of experiments are to be conducted in order to determine how factor ‘$a$’, the number of potential recommendation candidates, affects different recommendation algorithms. If the influence of this factor is examined in a range from $f_a = 10$ to $f_a = 100$ in increments of 10, then all sub-datasets containing between 10 and 20 locations are retrieved for the first experiment. The second experiment is then conducted with all datasets containing 20 to 30 locations and so forth. The last experiment uses all the datasets that contain between 90 and 100 locations. The range of
these increments will hereinafter be referred to as ‘granularity’. All values of $f_a$ within this range will be merged into a single value for the purpose of the respective experiment.

For each one of these experiments, the values of the other factors $f_b$ to $f_e$ are random. In order to avoid dependencies related to these other factors, each experiment with $f_a = x$ is conducted multiple times, and then the average of the results is calculated. It is assumed that at least 5 independent sub-datasets for each value of each factor are necessary in order to guarantee the validity of the results.

Theoretically, a very large number of different subsets $N$, even with the maximum value of $x=100$ can be generated. $N$ is calculated by the combination formula:

$$N = \binom{150}{100} = \frac{150!}{100!50!} \approx 2.01E40$$

However, some of these datasets are almost identical apart from few items, and as such are not fully independent. In order to guarantee a certain independency between two random sub-datasets, a maximum overlap of 10% is taken to be acceptable. With the original dataset of 150 items, the probability $P$ of randomly picking two sample sub-datasets containing 100 items, which fulfil this condition, is very close to zero. $P$ is defined as

$$P = \frac{N_{\text{ind}}}{N}$$

where $N$ is the total number of possible datasets (see equation above), and $N_{\text{ind}}$ prescribes the total number of datasets with a maximum overlap of 10%.

Let $m$ be the total number of items in the database (in this case 150) and $n$ be the number of items in a sub-dataset (in this case 100). Also, let $s$ be the number of identical items in both sub-datasets. Then, $N_{\text{ind}}$ can be computed as follows:

$$N_{\text{ind}} = \binom{m - n}{n} + \sum_{s=0}^{s_{\text{max}}=0.1n} \binom{n}{s} \binom{m - n}{n - s}$$

Once the first sub-dataset is defined, only $m-n$ items remain as possible candidates for the second sub-dataset, on condition that the two sub-datasets do not overlap. The resulting number of possible combinations is represented by the first term on the right-hand side.
Similarly, the second term describes all possible sub-datasets that share one or more (up to $S_{max}$) items with the first sub-dataset (see Figure 6).

Figure 6: A scenario for two randomly picked datasets

In the present example of $m=150$, $n=100$ and $s=10$, $P$ would be $5.01\times10^{-12}$.

In case of a database which is enlarged to $m=1500$ items, $P$ for $n=100$ would be 0.9371743052842227, thus approximately 94%. As such, more than nine out of ten randomly picked datasets are independent according to the abovementioned conditions.

For the current research project, a method inspired by the idea of so called ‘filterbots’ (Sarwar et al., 1998; Park et al., 2006) was employed in order to generate this enlarged dataset. Filterbots are software agents that automatically rate new items on creation. This technique is normally used to improve the performance of recommender systems with respect to sparse datasets or cold-start situations where recommendations should be generated for new users or items.

In case of this project, a software module that enlarges the original dataset by generating ‘bots’ (i.e. artificial users), items and ratings based on existing users and reviews was used. The generated users have user-profiles that are similar, but not the same as the profiles of the users they are based on. There is also some small element of randomness in the artificial reviews written by these users. For example, when a user rated a location with a ‘3’ on a 0-5 scale, then the respective bot could rate it with a ‘2’, a ‘3’ or a ‘4’ – but not with a ‘0’, ‘1’ or ‘5’. This method made it possible to increase the size of the dataset in a reasonably
controllable and configurable fashion without changing its immanent qualities or adding purely synthetic data.

3.5 Algorithms under investigation

In the course of this research, 1425 experiments were conducted, each single one evaluating Precision, F1, and MAE (see section 3.1) on six different runtime factors (see section 3.3) for six different recommendation algorithms. Some of these algorithms allow certain options to be configured (e.g. the neighbourhood size – see section 1.2). The effects of configuring these parameters with several different values were investigated. Table 1 lists the algorithms under investigation, the abbreviation by which they will be referred to in the rest of this document and any possible configuration options.
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Recommendation algorithm</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDNU_x</td>
<td>A user-based CF algorithm using a very simple similarity measure based on the standard deviation (Wikipedia, 2010a) between ratings.</td>
<td>x declares the neighbourhood size of most similar users as defined by the similarity metric.</td>
</tr>
<tr>
<td>ENU_x</td>
<td>A user-based CF algorithm using a similarity measure based on the Euclidian distance (Segaran, 2007).</td>
<td></td>
</tr>
<tr>
<td>PNU_x</td>
<td>A user-based CF algorithm using a similarity measure based on the Pearson correlation coefficient (Shardanand and Maes, 1995; Vozalis and Margaritis, 2003a).</td>
<td></td>
</tr>
<tr>
<td>TNU_x</td>
<td>A user-based CF algorithm using a similarity measure based on the Tanimoto coefficient (Strehl and Ghosh, 2000; Wikipedia, 2010b).</td>
<td></td>
</tr>
<tr>
<td>S1</td>
<td>Slope One (Lemire and Maclachlan, 2005).</td>
<td>No configuration options.</td>
</tr>
<tr>
<td>LFTN_x</td>
<td>A clustering algorithm using a Log-likelihood-based similarity measure (Dunning, 1993).</td>
<td>x declares the target number of clusters.</td>
</tr>
</tbody>
</table>

*Table 1: The recommendation algorithms under investigation*
3.6 The hybrid recommendation engine prototype

In compliance with the aims of the research outlined in section 1.4 and on the basis of the results obtained from extensive experiments with the corresponding sub-algorithms (see Chapter 4), two hybrid recommendation engine prototypes were developed and compared to the algorithms listed in Table 1. These prototypes were evaluated against the exact same datasets, with the same methods and under the same circumstances as the original algorithms. The general research approach followed in this project is visualized in Figure 7.

![Diagram of the general research method of the project]

Figure 7: The general research method of the project

An extensive survey of the relevant literature was conducted and the runtime factors outlined in section 3.3 were identified. The most promising candidates from the wide variety of recommendation algorithms available were selected and evaluated against each one of these factors in a series of controlled experiments. The obtained results made it possible to design and configure the hybrid algorithms, which were then evaluated using the same experimental protocol. The hypothesis proposed in section 1.4 could then be tested and the answer to the research question verified. This will be the subject of Chapter 6. Details of the functional
principles for the prototypes as well as comprehensive experimental results will be presented in Chapter 5.

3.7 Possible alternative research methods

Herlocker et al. (2004) suggest going beyond measuring only the objective accuracy of a recommender system if possible. In case of this project, it was considered to conduct a field experiment by adding functionality to the regionrex.com application that would have allowed users to give feedback on the subjective quality of the recommendations they received. Alternatively, actual users could have been observed and interviewed in a laboratory setting while using the system. However, this approach was rejected for the purpose of this project because it would have required a very large number of human participants. Time and resources were not available.

3.8 Summary

The predictive accuracy of recommendation algorithms can be determined by means of either ‘statistical’ or ‘decision support’ accuracy metrics. This dissertation uses these metrics to analyse the influence of six different runtime factors on different recommendation algorithms. For this purpose, a special evaluation protocol and experimental setup was developed.
Chapter 4  Evaluation of recommender systems

In the course of this project, systematic experimental research was conducted in order to determine how each single algorithm under investigation behaves under different circumstances. This chapter presents and analyses the results of this research.

4.1 Boundary conditions to the experimental setup

The dedicated experimental setup described in Chapter 3 allowed the controlled experiments to be carried out for the present study. For each single experiment, several evaluation metrics (MAE, Precision and F1) were used to determine the effects of certain runtime factors (see also section 3.3) on the predictive accuracy of the recommendation algorithms under investigation (see section 3.5). Table 2 lists these factors along with the name under which they will be referred to in the rest of this document together with the range of values that was analysed. The fourth column, ‘Granularity’, shows the range of values that were averaged and merged into a single data point in order to ensure the validity of the results (see section 3.3). For example, a granularity of 10 in a range of values starting at five means that all values between five and 14 would be merged. In order to avoid bias, a respective data point was only created if more than five values were available.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Range of values</th>
<th>Granularity</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of recommendation candidate locations</td>
<td>The number of items that is within a certain distance from the current user and consequently could be recommended by the system.</td>
<td>5 – 100</td>
<td>10</td>
</tr>
<tr>
<td>number of locations in dataset</td>
<td>The number of items in the sub-dataset that the engine can use to calculate recommendations.</td>
<td>500 – 1500</td>
<td>50</td>
</tr>
<tr>
<td>number of reviews</td>
<td>The number of reviews in the sub-dataset that the engine can use to calculate recommendations.</td>
<td>1800 – 3400</td>
<td>100</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
<td>-------------</td>
<td>-----</td>
</tr>
<tr>
<td>number of users</td>
<td>The number of users in the sub-dataset that the engine can use to calculate recommendations.</td>
<td>20 – 100</td>
<td>5</td>
</tr>
<tr>
<td>sparsity</td>
<td>The reciprocal value of the average percentage of items rated per user in the sub-dataset that the engine can use to calculate recommendations (i.e. the inverse of the density of the particular user-item-matrix).</td>
<td>2.5 – 10</td>
<td>0.5</td>
</tr>
<tr>
<td>average number of reviews per location</td>
<td>The average number of reviews per item in the sub-dataset that the engine can use to calculate recommendations.</td>
<td>1.5 – 2.5</td>
<td>0.1</td>
</tr>
</tbody>
</table>

**Table 2: The runtime factors under investigation**

Although a predictive accuracy metric (MAE) as well as classification accuracy metrics (Precision and F1) were evaluated for every experiment, the following sections will concentrate on the latter, because they are a better fit for the use case. The application that provides the background and the database for the research project at hand, regionrex.com, is designed to recommend a certain number of items as an unordered list. For this kind of task, which is termed ‘Find Good Items’ by Herlocker et al. (2004), classification accuracy metrics are more suitable than predictive accuracy metrics, because the predicted preference value and the exact ordering of recommendations is not important as long as the system is able to correctly classify items as relevant or irrelevant (ibid.). Precision and Recall were evaluated based on the top-10 results (‘Precision at 10’ or ‘P@10’ – see section 3.1).
In the following sections, the results are presented in the form of F1 scores, which combine Precision and Recall into a single number (see section 3.1) starting at zero and reaching its best value at one. For the sake of completeness, the remaining results are included in Appendix B. The observed trends make it possible to analyse and compare the behaviour of the various algorithms. With respect to the current research, these trends are more significant than the actual scores, which depend largely on the number of topmost results evaluated. In this case, this number is 10 (i.e. the F1 score is calculated from the “precision at 10” - see section 3.1). A different number, for example P@5 or P@3, is likely to lead to different scores while the trends would more or less be the same (Kuo et al., 2009).

4.2 Evaluation results

The following figures depict the F1 scores of all factors listed in Table 2 for all algorithms listed in Table 1. Different values for the neighbourhood-size of one CF algorithm under investigation (ENU) are included in the results to demonstrate the effect of this configuration option. In case of the other algorithms, the best performing option is plotted.

![F1 for factor 'number of recommendation candidate locations'](image)

**Figure 8: F1 scores for factor ‘number of recommendation candidate locations’**

The results for the first factor, ‘number of recommendation candidate locations’ shown in Figure 8 make a certain trend visible. For very small datasets, LFTN_12 is clearly the best performing algorithm. However, its performance decreases as the dataset size increases. According to the averaged results, ENU_2 is the most suitable choice for datasets with more than 21 candidate locations. ENU_5 and TNU_5 achieve almost the same accuracy, the latter...
performing slightly better than the former, which increases with the size of the dataset. It can be presumed that they would lead to even better results than ENU_2 for datasets with significantly more than 95 candidate locations, but this would have to be confirmed by further experiments. The remaining 5 algorithms show inferior results over the whole range of values investigated.

Figure 9: F1 scores for factor 'number of locations in dataset'

Figure 9 shows that similar trends can be observed for the second factor, 'number of locations in dataset'. The lines are less straight as compared to Figure 8 due to the greater statistical spread resulting from the greater range of values. However, the basic results are the same: LFTN_12 performs best for small datasets while ENU_2 performs best for medium sized to large datasets. The curves representing ENU_5 and TNU_5 approach ENU_2 towards larger datasets, and it is very likely that they would intersect if datasets with more than 1500 locations were investigated.

Figure 10 to Figure 13 confirm the same trends. It can be concluded that all the independent factors behave in a similar way.
Figure 10: F1 scores for factor ‘number of reviews’

Figure 11: F1 scores for factor ‘number of users’
Figure 12: F1 scores for factor ‘sparsity’

Figure 12 shows the factor ‘sparsity’. Higher sparsity values mean that the dataset is less dense, or in other words, that not so many locations have been rated by each user. Consequently, the fact that LFTN_12 performs best in situations where the user-item matrix is relatively dense while the predictive accuracy of ENU_5 and TNU_5 decreases with higher sparsity values confirms the tendencies that the other factors made evident.

Figure 13: F1 scores for factor ‘average number of reviews per location’

While the sparsity of the dataset refers to the number of locations rated per user, the last factor under consideration, ‘average number of reviews per location’ depicted in Figure 13
looks at the user-item-matrix from the perspective of the location. It confirms the same trends that can be observed in Figure 12.

4.3 Data smoothing

The effect of runtime factors on the predictive accuracy of the recommendation algorithms can be made even clearer by applying a curve fitting function on the respective data points. For the sake of completeness, the details of this process are explained in Appendix C. Figure 14 to Figure 19 show the resulting charts including only the best performing four algorithms ENU_2, ENU_5, LFTN_12 and TNU_5. The remaining results are provided in Appendix B.

The data points are displayed as shapes without a connecting line. The respective curves are marked with the suffix ‘pf2’ (i.e. second degree polynomial fit) in the legend of each diagram. They represent the mathematical function that has the best fit to the data points representing the results of the respective algorithm. In the course of this research, polynomial curve fitting functions of first to fifth order were implemented and evaluated. The second order polynomial function including a root term, \( y = ax^2 + bx + c + dx^{0.5} \), proved to exhibit the least mean error while still best emphasizing the trends of runtime factor effects on the predictive accuracy of recommendation algorithms.

Compared with Figure 8, Figure 14 makes it more obvious that the performance of LFTN_12 rapidly decreases as the number of recommendation candidate locations approaches a value of 65. After that, the degradation stagnates and the predictive accuracy remains more or less static.
Figure 14: Curve fitting functions (F1) for factor ‘number of recommendation candidate locations’

Figure 15: Curve fitting functions (F1) for factor ‘number of locations in dataset’

Figure 15 best demonstrates the value of the curve fitting function. The statistical noise is ‘smoothed’ (Wikipedia, 2010d) and the effect of the runtime factor ‘number of locations in dataset’ on the recommendation quality becomes more apparent. While still being the most suitable algorithm for small to medium sized datasets, LFTN_12 apparently performs not so well when there is only a very small number of locations available for recommendation generation. Its performance seems to reach a peak around a value of approximately 700 items and then decreases again. However, this effect does not seem to apply to the remaining factors (see Figure 14 and Figure 16 to Figure 19), although it can be observed in a similar but less...
distinct fashion in the behaviour of ENU_2 across all factors. More specific investigations would be necessary in order to probe the causes of this phenomenon.

Figure 16: Curve fitting functions (F1) for factor ‘number of reviews’

Figure 17: Curve fitting functions (F1) for factor ‘number of users’
4.4 Analysis of the observed results

It can be observed that the predictive accuracy of the CF algorithms under investigation (e.g. ENU_5 and TNU_5) tends to increase with the available amount of data. This finding is consistent with the results obtained by other researchers in the field (e.g. Rao and Talwar, 2008 – see section 2.4). However, the results related to CF with a very small neighbourhood size (ENU_2) were found to be rather static over the whole range of values. A conceivable
explanation is that there is always enough data for the algorithm to find two relatively similar users. For the specific dataset and use case under consideration, this configuration proved to be superior as compared to larger neighbourhood sizes. The implication is that the dataset contains a wide bandwidth of different types of users and that not many users share the same taste with respect to the locations they like. The fact that the clustering algorithm (LFTN_12) performs best as long as the datasets are small confirms this assumption. This algorithm builds clusters of similar users until the number of clusters equals twelve. Recommendations are then generated for the cluster that the current user belongs to (Montaner et al., 2003 – see section 2.3). For large datasets it can be assumed that users which are actually not very similar are assigned to the same cluster in order to reach the target of having twelve distinct clusters. It is very likely that this has a negative effect on the predictive accuracy.

The fact that ENU_5/TNU_5 approaches ENU_2 towards the end of the range of values under investigation (for the factor ‘sparsity’ this corresponds to the start of the range of values) again suggests the assumption that ENU_5/TNU_5 would perform best for even larger datasets where a neighbourhood size of five is a more suitable choice than two.

4.5 Summary

The results of each experiment were analysed in terms of the evaluation metrics MAE, Precision and F1. However, F1 was found to be the most suitable metric. While for small and sparse datasets, LFTN_12 proved to be the best performing algorithm, ENU_2 shows superior results for medium to large or very dense datasets. An appropriate curve fitting function was applied to make these trends more easily visible. The observed tendencies can be accounted to the fact that, while the performance of CF increases with the available amount of data, predictive accuracy depends largely on an optimal value of the neighbourhood size.
Chapter 5  The hybrid recommendation engine

In order to answer the research question of this project (see section 1.4), two different hybrid recommendation algorithms were developed. Both of these prototypical implementations take the current situation of use into account. This chapter explains their functional principles and analyses how well they perform as compared to each single sub-algorithm under investigation.

5.1 The rationale for the hybrid algorithms

The rationale for both hybrid algorithms is based on the insight that for each runtime factor, clustering proved to be the most effective recommendation method until a certain value of the respective factor was reached. From there on, CF produced better results than clustering. The findings are summarized in Figure 20.

![Figure 20: The principle trend observed in the experiments with conventional algorithms](image)

The hybrid algorithms exploit these findings and determine the suitability of the available sub-algorithms with respect to the surrounding circumstances.

5.2 The switching hybrid prototype

The first hybrid recommendation algorithm developed in the course of this project is a ‘switching’ hybrid as defined by Burke (2002) (see section 2.5). This means that it chooses a
single recommendation technique depending on some switching criterion. In this case, it not only uses one criterion but six. These criteria are the runtime factors that were defined in section 3.3 (see also Table 2). Figure 21 shows the functional principle of this hybrid recommender.

![Figure 21: The functional principle of the switching hybrid recommender](image_url)

Given a recommendation request, the recommender system first analyses the relevant sub-dataset. It determines the values of all six runtime factors and based on this information, it decides which particular recommendation method will be most effective. For each runtime factor $f$, a suitability score $s_i$ is assigned to each single sub-algorithm. The system then chooses the sub-algorithm with the highest score $s = \sum_f s_f$ and delegates recommendation generation to the software module implementing this algorithm (referred to as “Recommender” in Figure 21).
5.3 The weighted hybrid prototype

In hopes of achieving even better results as compared to the switching hybrid algorithm, a second prototype was developed and evaluated. Its functional principle is visualised in Figure 22.

![Diagram of the weighted hybrid prototype]

**Figure 22: The functional principle of the weighted hybrid recommender**

This algorithm can be classified as a ‘weighted’ hybrid as defined by Burke (2002). Instead of choosing the single most suitable recommendation technique, it uses multiple methods simultaneously and then combines the results of these sub-algorithms under consideration of the degree of each one’s suitability for the current situation. This combination is performed by a software module referred to as “ResultMergerStrategy”. The relevant sub-dataset for the current recommendation request is analysed with respect to the six runtime factors defined in section 3.3. For every single sub-algorithm and configuration, a score $s$ that reflects the sub-algorithm’s suitability is assigned. A higher value of $s$ implies that the respective sub-algorithm is expected to produce more accurate results.
The system compiles a list of items containing all the recommendations from each single sub-algorithm. Each item on this list is described by a total score $ts$ calculated from the sub-algorithm’s estimated rating $r$ for the respective item and the sub-algorithm’s suitability score $s$. Let $N$ be the total number of sub-algorithms under consideration, then the total score $ts$ is shown formally as:

$$ts = \sum_{i=1}^{N} s_i \cdot r_i$$

Those items with the highest total score $ts$ are then assigned to the final recommendation list which is presented to the user.

Theoretically, this hybrid algorithm has the potential to produce better results than any single sub-algorithm it uses, because it chooses only the best results from each sub-algorithm and in addition to that exploits the information whether an item was recommended by more than one sub-algorithm at the same time. On the other hand, of course the computation time required is the time taken by each sub-algorithm plus the time for assembling the final list of recommendations.

### 5.4 Analysis of the observed results

The switching hybrid (SH) produces equally good results over the whole range of runtime factor values under investigation when compared with the respective best performing sub-algorithm. This result was to be expected, because SH always uses the optimal sub-algorithm in every particular situation. However, the weighted hybrid algorithm (WH) in practice fails to achieve further improvements when compared with SH and, in fact, performs suboptimally in some cases.
Figure 23: F1 scores for factor ‘number of recommendation candidate locations’ including the hybrid algorithms

Figure 23 shows that for datasets with a very small number of recommendation candidate locations, SH reaches the predictive accuracy of LFTN_12. WH performs better than the next best sub-algorithm ENU_2, but worst than SH and LFTN_12. The most probable explanation for this behaviour is that LFTN_12 performs so much better than ENU_2 in these situations that even the worst predictions of LFTN_12 are more accurate than the best estimations of ENU_2. Apparently, in this case it can happen that WH nevertheless uses some of the recommendations produced by ENU_2, which adversely affects the F1 score achieved. For larger datasets WH performs equally well as SH or ENU_2 but never outperforms SH as hoped. Obviously, it fails at successfully exploiting its additional sources of information.

The results shown in Figure 24 to Figure 28 confirm these conclusions. SH always reaches the highest F1 values together with the best-performing sub-algorithm. For relatively large or dense datasets, WH often reaches the same accuracy. For smaller or more sparse datasets, WH usually performs as well or better as compared to ENU_2, but not as well as LFTN_12 or SH.
Figure 24: F1 scores for factor ‘number of locations in dataset’ including the hybrid algorithms

Figure 25: F1 scores for factor ‘number of reviews’ including the hybrid algorithms
Figure 26: F1 scores for factor ‘number of users’ including the hybrid algorithms

Figure 27: F1 scores for factor ‘sparsity’ including the hybrid algorithms
Looking at Figures 18, 19, 22 and 23, it becomes apparent that whenever LFTN_12 performs better than ENU_2, WH benefits from the results produced by LFTN_12. In situations where ENU_2 is the most suitable algorithm, WH is not affected in a negative way by considering the results of LFTN_12, but does not profit from them either. It can be concluded that in situations where CF is the better choice, this choice is very clear. Obviously, the suitability score determined by WH (see the formula in section 5.4) is very distinct in these cases. In other words, most or all of the runtime factor values appear to be on the side of the CF algorithm. On the other hand, in situations where LFTN_12 is a better choice than ENU_2, some runtime factor values still seem to be on the side of the CF algorithm. As a consequence, WH still places some weight on the results of ENU_2.

In conclusion, it can be noted that SH proves to be the most effective algorithm in every situation of use. This principle trend observed in the results of all the experiments with the hybrid algorithms is depicted in Figure 29.
The red line in Figure 29 represents the results of SH while the orange line marks the point where it switches between recommendation methods.

5.5 Summary

In the course of the research project at hand, a switching hybrid recommendation algorithm was developed and evaluated. In contrast to the other algorithms analysed, it was the only recommendation method to achieve top results over the whole range of runtime factor values under investigation. Hoping to further improve on these results, a second hybrid algorithm utilizing a different technique to combine sub-algorithms was designed and implemented. In practice this approach failed to further increase the predictive accuracy of the recommendation engine. However, the experimental results obtained from this algorithm provide interesting insights and implications that can be valuable in the light of future research.
Chapter 6  Discussion

In Chapter 1, a hypothesis was proposed and a research question was raised. Confirming or disproving this hypothesis and finding an answer to the respective research question is the aim of this project. Based on the results presented in Chapter 5, it is now possible to achieve this aim.

6.1 Analysis of the results with respect to the research aims

The hypothesis claimed that the predictive accuracy of a recommender system depends on several runtime factors. It was assumed that a hybrid system which dynamically selects the optimal sub-algorithm based on the respective values of these runtime factors for every particular recommendation request would produce more accurate results overall when compared with a system that uses only a single recommendation method. Based on the results presented in Chapter 5, this seems to be the case. For example, the runtime factor ‘number of recommendation candidate locations’ (see Figure 23), shows that a system using only LFTN_12 as a recommendation technique produces suboptimal results for large datasets, while a system using only ENU_2 produces suboptimal results for small datasets. On the other hand, a system that uses the hybrid algorithm SH achieves good results in every situation. The same is true for the other runtime factors under investigation. Therefore, the research question of this project can be answered in the affirmative.

6.2 The current research in a wider context

Rao and Talwar (2008) report that the predictive accuracy of CF increases with the amount of data available. This finding can be confirmed in the light of the current research project. The results show that, as long as the neighbourhood size is reasonably small, the F1 scores achieved by the CF algorithms under investigation usually increase with the amount of information that they have at their disposal. In contrast to that, the accuracy of Slope One and the clustering algorithm decreases with the available amount of data – at least in the case of the regionrex.com dataset.

Burke (2002) claims that, while introducing additional complexity into the recommendation process, switching hybrid recommendation techniques bring the benefit of making the system sensitive to the strengths and weaknesses of its constituent recommendation algorithms. The
research project at hand reinforces this claim. The switching hybrid prototype developed in
the course of this project compensates for the proven weaknesses of CF by using a more
suitable recommendation technique when this is advantageous, thus improving the accuracy
of the system as a whole. On the other hand, it requires more computational time because the
relevant sub-dataset has to be analysed before recommendations can be generated. For small
datasets as in the case of regionrex.com, this overhead is probably acceptable, but this might
not be the case for very large datasets. However, it could be considered to only take runtime
factors into account that are relatively fast and easy to determine.

With respect to weighted hybrids, Burke (2002) argues that the benefit of this method is that
all of the system’s capabilities are brought together in a straightforward way, and that post-
hoc adjustments can be made. Looking at the design of the weighted hybrid developed in the
course of this project (see Figure 22), this can be confirmed. It would easily be possible to
supply alternative implementations of the ResultMergerStrategy-module to adjust the
recommendation process. Burke also notes that one drawback of this approach is that it makes
the implicit assumption that “the relative value of the different techniques is more or less
uniform across the space of possible items” (ibid.). The present research shows that this does
not have to be the case. The proposed algorithm dynamically determines the relative value of
each recommendation technique, taking the space of possible items for the respective
situation into account.

6.3 Summary

The aim of this research project has been achieved. The hypothesis that an overall
improvement of a recommendation engine is possible by analyzing the relevant sub-dataset
for each recommendation request has been confirmed and thus the research question could be
answered positively. This is in conformance with the findings of other researchers in the field.
Chapter 7   Conclusions

The research project can be regarded as a success. The proposed hypothesis could be confirmed and the research question could be answered positively. The following sections justify the choice of research methodology, discuss the limitations of the selected approach and suggest possible areas for future research.

7.1 Project Review

This research project was striving for an ambitious goal: To introduce a novel optimisation approach for recommender systems that improves the results accomplished by some of the most elaborate algorithms from an extremely active field of research. Within the well-defined limits it set for itself, it can be regarded as successful with respect to that goal. No single recommendation algorithm under consideration achieved more accurate results over the whole range of experimental conditions investigated than the switching hybrid algorithm developed in the course of this project. The runtime factors identified as well as the experimental environment proved to be suitable to confirm the hypothesis given at the beginning of the project.

The chosen research method and, in particular, the evaluation metric used (F1, or rather PR) has been extensively utilized by researchers in the field to evaluate solutions to exactly the same kind of problem as investigated in this project (Basu et al., 1998, Billsus and Pazzani, 1998; Karypis, 2001, Kuo et al., 2009, Yu et al., 2006). This metric is also known to be more suitable for the use case of unordered recommendation lists as compared with other evaluation metrics like MAE or RMSE (Herlocker et al., 2004). Although it was not obvious at the beginning of the project how this might be achieved, a solution for implementing the research method in a way that avoided bias as good as possible was eventually found. As a result, the amount of data that formed the database for the final experiments and the total number of experiments conducted was sufficient to assure the independence of the investigated runtime factors.

Nonetheless, there may be some limitations regarding the generalisability of the obtained findings. First of all, the research was conducted on the basis of one specific database only. The data was generated by real users in a real-world application, but this application was still in a beta-testing phase and the database was synthetically enlarged for the research project.
This approach was useful to guarantee the statistical validity of the quantitative data collected through experiments, but it cannot be guaranteed that the data gathered in the beta-test is also typical for operational business. For example, real-world datasets might exhibit different ratios of users and items. However, the general optimisation approach and the basic insights regarding the behaviour of the various recommendation techniques can in principle be generalised to operational business or other application domains.

Another point to consider is that different evaluation metrics measure different aspects of recommendation quality. This research concentrated on the decision support accuracy metric F1, because this method is believed to be most appropriate for the task performed by this particular recommender system. As a consequence, the generalisability of the method and findings to applications with different use cases has its limits. Other metrics might be more significant in these cases.

In addition to that, the recommendation methods under investigation can be implemented in different ways. Different implementations can lead to slightly different results and thus the obtained findings necessarily depend, to some degree, on implementation details.

7.2 Evaluation of the scientific method

The scientific method used in this research project consists in devising a theory which is formulated as a hypothesis and a research question and then tested and reviewed in the light of appropriate evidence. In order to gather this evidence, a prototypical implementation was built and validated in a series of controlled experiments. The conclusions drawn from analysing the obtained findings were placed in the context of existing work and communicated in a form that allows other researchers to scrutinise and build upon them.

This scientific method was implemented in a systematic way. A careful study of the existing literature has shown that an empirical approach based on the experimental evaluation of a hypothesis is an established method for this kind of research. Consequently, a theory how recommender systems could be optimised for dynamic environments was developed and specified in the form of a hypothesis and a research question.

The underlying conditions were precisely defined so that the research question “Can the overall predictive accuracy of a recommender system be improved by dynamically selecting
or combining different recommendation algorithms depending on the current situation of use?” could be interpreted without ambiguity. In particular:

- A metric to assess the **predictive accuracy** of recommender systems was chosen. Predictive accuracy was defined as the system’s objective precision in predicting ratings.
- The term **overall improvement** was defined as an improvement over the whole range of runtime factor values.
- The concept of a **dynamical selection or combination of recommendation algorithms** within hybrid algorithms was explained.
- Six runtime factors that characterize the **current situation of use** were identified.

Based on this research question, an appropriate research method was chosen. The evaluation strategy that was developed and implemented produced the necessary evidence to establish the hypothesis. This evidence is believed to be relevant in the context of the defined criteria, because the richness of the data obtained by an automatic analysis of evaluation metrics is adequate to answer a question regarding the predictive accuracy as defined in this project. Appropriate measures to avoid bias were taken, and the obtained results proved to be repeatable. Therefore, the scientific method is sufficient to confirm the hypothesis.

Of course it can be argued that no research can ever be regarded as complete. For example, other research methods could be applied to the same problem or it could be investigated whether the hypothesis is also true in different contexts. The following section suggests some promising starting points for further research.

### 7.3 Future Work

The research project at hand showed that the overall predictive accuracy of a recommender system can be improved by introducing a hybrid recommendation algorithm. This complies with the research aims, but due to time and resource constraints, some interesting and promising aspects of the issue-area could not be addressed:

- **Evaluation of qualitative data**: During this project, only objective, quantitative evidence was evaluated. Without doubt, evaluating subjective feedback from actual users in addition to that would have increased the relevance of the research with respect to practical applications. Future research projects could use alternative research methods to capture and analyse a more qualitative form of data.
• **Design, Implementation and Evaluation of a more effective hybrid algorithm:** As the results detailed in section 5.4 show, the switching hybrid recommendation algorithm developed in the course of this project achieved the expected results. However, the weighted hybrid algorithm failed to improve on the switching hybrid algorithm. Therefore, it could benefit from further research with the aim of translating the additional data it gains from combining different recommendation methods into a higher predictive accuracy.

• **Design, Implementation and Evaluation of a clustering algorithm with a dynamic number of clusters:** The clustering algorithm LFTN_12 uses a target number of 12. This means that clusters of similar users are formed until a number of twelve clusters is reached. This configuration proved to be the most suitable choice for the given dataset and LFTN_12 turned out to be the most effective algorithm for situations where only a small amount of information is available. However, an algorithm that uses a dynamic target number, which is adjusted depending on the characteristics of the relevant subset of items, could produce even better results. Further investigations would be necessary in order to investigate the potential of such an algorithm.

• **Design, Implementation and Evaluation of a generic framework for an optimised application of recommendation techniques:** This research project concentrated on a specific application and dataset. It could be possible to extend this approach to build a generic framework that autonomously learns over time which recommendation technique would be the optimal choice for a specific situation.

• **Support for multicriteria ratings:** Although the regionrex.com database used in the course of this project contains ratings in different subcategories, only the overall rating for each location was used. As Adomavicius and Tuzhilin (2005) point out, incorporation of contextual information and the support for multicriteria ratings in recommendation systems are still a field of research that leaves much space for improvements. A future research project could investigate whether exploiting this additional source of information helps to improve the quality of the recommendations.
References


Adomavicius, G. & Tuzhilin, A. (2005), 'Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions', *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734-749.


Brunato, M. & Battiti, R. (2003), *A Location-Dependent Recommender System for the Web*, University of Trento, Trento, Italy.


Netflix (2009) *Netflix Prize Data Set* [online],


Padilla, A. (2008) *Comparing Distance Formulas Part III* [online],


Strehl, A. & Ghosh, J. (2000), 'Value-based customer grouping from large retail data-sets', *PROC SPIE INT SOC OPT ENG*, The University of Texas, Austin, TX, USA, pp. 33.


Index

classification accuracy metric, 22, 34
Clustering, 15, 30, 43, 53
collaborative filtering, 1, 11
decision support accuracy metric, 21, 22
Euclidian distance measure, 13, 30
evaluation metric, 21–22, 25
F1, 22, 29, 33, 34, 35, 56
filterbot, 3, 28
granularity, 26, 33
hard criterion, 5, 19, 23
hybrid recommender system, 2, 9, 17–18, 31–32, 44–52
information overload, 1, 11
location-based service, 3, 18–19
mean absolute error, 21, 29, 33, 55
nearest neighbour. See neighbourhood
neighbourhood, 1, 29, 30, 35, 42, 53
objective accuracy, 32
Pearson correlation coefficient, 14, 30
precision and recall, 22, 29, 34, 55, 56
predictive accuracy, 8, 21, 23, 33, 34, 39, 53, 57
recommendation technique
collaborative, 2, 11, 16
content-based, 2, 11, 16
demographic, 2, 11, 16
hybrid, 2, 11
item-based, 12, 16
knowledge-based, 2, 11
memory-based, 12
model-based, 12
user-based, 12, 16
utility-based, 2, 11
regionrex.com, 4, 9, 26, 32, 34, 53
root mean squared error, 21, 55
runtime factor, 7, 24, 31, 33–34, 39, 53, 57
similarity measure. See similarity metric
similarity metric, 13–15
similarity score. See similarity metric
Slope One, 15, 30, 53
sparsity, 3, 24, 34, 38
standard deviation, 13, 30
statistical accuracy metric, 21
switching hybrid, 17, 44, 47, 53
Tanimoto coefficient, 14, 30
weighted hybrid, 18, 46, 47, 54
Introduction

Modern information technology systems often contain huge amounts of data. Recommendation engines provide means to utilize this data in a useful way by identifying the pieces of information which are most relevant for each particular user. Various methods for achieving this aim have been developed over the past 20 years, but each one of these methods has its own weaknesses in certain situations. Combining multiple recommendation methods within a hybrid recommendation algorithm is a technique that can be used to compensate for these individual weaknesses.

The research project introduces a novel approach for hybrid recommender systems that makes use of the fact that for each individual recommendation request, the surrounding circumstances are different. This means that the relevant subset of all the available information items is also different, and by analyzing the characteristics of this subset, the recommender system can select the most suitable recommendation method. Evaluating whether this approach actually improves the quality of the generated recommendations is the research question investigated in the course of this project.
Method

So far, this approach of analyzing the relevant subset of items before generating recommendations has not been the subject of scientific investigations. For this reason, a novel method for scientifically evaluating such a system is proposed in the course of the research project. The interesting aspect of this method is that it runs a huge number of independent experiments and then derives meaningful and statistically unbiased results from the large amount of data they produce.

Six distinct factors characterizing the relevant subset of items for each particular recommendation request are identified. The respective values of these factors are unknown when the system is designed and have to be determined at runtime. On this account, these factors are referred to as “runtime factors”. The behaviour of six basic recommendation methods and various configuration options for these methods are investigated with respect to varying runtime factor values. The obtained results are used to design and configure two hybrid recommendation algorithms, which are then themselves evaluated in direct comparison with conventional algorithms in order to answer the research question of this project.

The following schema shows the general research method followed in this project:
Results

The experimental results reveal certain trends in the behaviour of the recommendation algorithms under investigation. While some recommendation methods and configuration options produce relatively bad results over the whole range of runtime factor values, most of them improve with the amount of information that they can work with. The more data they have at their disposal, the more accurate are their recommendations. However, one particular recommender behaves in an oppositional way. This software module uses a technique that is known as clustering. It forms groups of likeminded users and then generates recommendations for these groups. While clearly being the most effective technique for situations where there is not much data to work with, the performance of this algorithm decreases when the amount of information increases. From a certain point on, a module based on a different recommendation method known as collaborative filtering performs much better.

The first hybrid recommendation algorithm investigated in the course of this project makes use of the obtained findings in order to improve the quality of recommendations over the whole range of runtime factor values. It is a switching hybrid that selects the single most suitable recommendation method by analyzing the characteristics of the relevant subset of items for each recommendation request.
The principle trend observed in the experimental results

Similar trends can be observed for all the independent runtime factors. These trends are depicted in principle in the chart above. While the clustering algorithm (the green line in the chart) is more suitable for situations where only a small amount of data is available, collaborative filtering (the blue line) produces better results when the relevant subset of items contains more information. The red line shows the accuracy of the recommendations produced by the switching hybrid. The point where it switches the utilized recommendation method is marked with an orange line.

It can be observed that the switching hybrid is the most effective algorithm in every situation. Trying to improve these results further, a second hybrid algorithm is investigated in this project. Instead of choosing a single recommendation method, this algorithm combines multiple techniques. It is referred to as a weighted hybrid, because it determines the suitability of each technique with respect to the characteristics of the relevant subset of items and then combines their results by using a formula that takes into account their respective effectiveness. However, contrary to the expectations, this algorithm fails to outperform the switching hybrid and in fact achieves only equal or worst results depending on the situation of use.
Analysis

As described above, the accuracy of the collaborative filtering algorithms under investigation tends to increase with the available amount of data while the accuracy of the clustering algorithm decreases. No single recommendation technique achieves optimal results under all circumstances. As a result, the quality of the overall system can be improved by using the switching hybrid algorithm.

The hope that the weighted hybrid algorithm could further improve the quality of the overall system beyond the sum of its parts did not come true. Although it utilizes a greater amount of information, this approach does not produce better results. However, the experimental results provide interesting insights that can be valuable for future research projects.

Discussion

The research project succeeds in confirming the hypothesis it constructed and in positively answering its research question. The proposed optimisation approach for recommender systems proves to be suitable to increase the quality of the investigated recommender system. The obtained findings also confirm the conclusions reached by other researchers in the field.

Although there are some limitations to the generalisability of the results, not least because of restricted time and resources, the developed procedure and research method is expected to be valuable for a great number of researchers and practitioners. The novel aspects of this approach can easily be adapted to other application domains and the dissertation provides starting points for interesting and promising future research projects.
Appendix B: Complete result plots

Curve fitting functions (F1) for all factors and algorithms under investigation
Curve fitting functions (Precision at 10) for all factors and algorithms under investigation
Curve fitting functions (MAE) for all factors and algorithms under investigation
Appendix C: Derivation of the polynomial curve fitting function

The polynomial curve fitting function applied for the purpose of data smoothing is based on the least square method.

\[ y = ax^2 + bx + c + dx^{0.5} \]

The last root term proved to be viable for describing the actual trends of the data points. In order to minimise the mean square error \( S \) for all \( n \) data points

\[ S = \sum_{i=1}^{n} [y_i - (ax^2 + bx + c + dx^{0.5})]^2 \]

all possible partial derivatives of \( S \) with respect to the unknown coefficients \( a, b, c \) and \( d \) have to vanish. This means the following conditions have to be fulfilled:

\[ \frac{\partial S}{\partial a} = 0; \quad -2 \sum_{i=1}^{n} [y_i - (ax^2 + bx + c + dx^{0.5})]x_i^2 \]

\[ \frac{\partial S}{\partial b} = 0; \quad -2 \sum_{i=1}^{n} [y_i - (ax^2 + bx + c + dx^{0.5})]x_i \]

\[ \frac{\partial S}{\partial c} = 0; \quad -2 \sum_{i=1}^{n} [y_i - (ax^2 + bx + c + dx^{0.5})] \]

\[ \frac{\partial S}{\partial d} = 0; \quad -2 \sum_{i=1}^{n} [y_i - (ax^2 + bx + c + dx^{0.5})]x_i^{0.5} \]

This system of linear equations can be written in matrix form in order to solve for the unknown coefficients \( a, b, c \) and \( d \).
In order to solve the system, a Gauss algorithm which transforms the given matrix into a diagonal form was implemented. Once the diagonal form of the matrix is present, the coefficients can be easily determined. Inserting these coefficients into the polynomial function produces the smoothed curves.