CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Evaluation of Machine Learning Algorithms in Recommender Systems

Candidate Recommender Systems in the Staffing Industry

Master's Thesis in Software Engineering

Adam Myrén

Piotr Skupniewicz Neto

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2017

MASTER'S THESIS 2017

Evaluation of Machine Learning Algorithms in Recommender Systems

Candidate Recommender Systems in the Staffing Industry

ADAM MYRÉN

PIOTR SKUPNIEWICZ NETO

Department of Computer Science and Engineering
Division of Software Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2017

Evaluation of Machine Learning Algorithms in Recommender Systems

Candidate Recommender Systems in the Staffing Industry

ADAM MYRÉN
PIOTR SKUPNIEWICZ NETO

© ADAM MYRÉN, 2017.

© PIOTR SKUPNIEWICZ NETO, 2017.

Master's Thesis 2017

Department of Computer Science and Engineering
Division of Software Engineering
Chalmers University of Technology

SE-412 96 Gothenburg, Sweden
Telephone: + 46 (0)31-772 1000

Chalmers Reproservice

Gothenburg, Sweden 2017

Abstract

Recommender systems are widely discussed in literature as they provide a solution to problems of information overload in a variety of contexts and application areas. When designing such systems, there are a wide range of options regarding what algorithms, approaches and techniques to use. This study addresses the problem of making key design choices when building candidate recommender systems in the staffing industry. Furthermore, the impact of using a variety of metrics to measure different properties of recommender systems is addressed. The study applies a design research approach, at a company providing an online recruiting platform, in which three different candidate recommender systems are implemented and evaluated. The results show that by varying the design of a candidate recommender system, different properties, such as accuracy, coverage, or diversity of recommendations, can be prioritized. By combining more than one recommender system into a larger system, however, many of the weaknesses of applying any individual approach can be circumvented. Also, broadening the scope of evaluation to include other properties than accuracy increases the ability chose a recommender system that performs in a way that is aligned with the business goals.

Acknowledgements

The authors would like to thank the staff at Jappa, and especially our supervisor Fredrik Borg, for the opportunity to conduct this thesis and for their encouragement and willingness to provide assistance whenever it was needed.

The authors would also like to thank Miroslaw Staron, supervisor at Chalmers University of Technology, for his continuous support throughout the project.

Adam Myrén and Piotr Skupniewicz Neto, Gothenburg, 2017

Table of Contents

1 Introduction

Recommender systems are data analytics applications that can provide solutions in a wide variety of contexts and application areas where users are exposed to information overload (Ricci, et al., 2015; Gunawardana & Shani, 2009). Such systems filter vital information from large amounts of data regarding user preferences and behavior, and by doing so, it can predict what the user is searching for and produce recommendations accordingly (Isinkaye, et al., 2015). For instance, recommender systems can serve as a good complement to search engines when these fail to produce promising results from vast amounts of information, by supporting users in finding preferred items and deciding about items in a collection (Ricci, et al., 2015).

The concept of recommender systems is widely discussed in the literature, and they have proven to be successful in various contexts and among many large information-based companies, such as Google, Twitter, and LinkedIn. (Al-Otaibi & Mourad, 2012; Portugal, et al., 2015). Since the beginning of the 2000s, the application of recommender systems has been profoundly researched, mainly by focusing on the design of new types of algorithms and techniques to produce recommendations (Gunawardana & Shani, 2009). Recommender systems are generally classified into content-based (CB) and collaborative filtering (CF) (Adomavicius, et al., 2007). The former analyzes the similarities between users and between items by their descriptions (Pazzani & Billsus, 2007), while the latter analyzes the similarity between users' actions to produce recommendations (Ricci, et al., 2015). Alternative approaches such as knowledge- and demographic-based approaches (Burke, 2007), or even hybrid variants of all the mentioned approaches have also been studied (Adomavicius & Tuzhilin, 2015). As the research around recommender systems has evolved, the use of machine learning algorithms in these systems has become one of the main topics due to their potential to improve such systems (Portugal, et al., 2015). Since there are a variety of options and application contexts, the designer of a recommender system must make decisions about the most appropriate techniques and algorithms to use (Gunawardana & Shani, 2009).

Previous research has been invested into the application of the general concepts of recommender systems in specific domains with the objective to test the generalizability of those concepts. One of the domains where recommender systems have proven useful is in the staffing industry, where it can aid job seekers in finding jobs that they are interested in. By several previous studies, it has been shown that hybrid recommender systems have outperformed the application of individual techniques in job recommender systems with regards to accuracy (Al-Otaibi & Mourad, 2012; Gupta & Garg, 2014; Hong, et al., 2013). Furthermore, clustering techniques for grouping users in these systems have also turned out to increase the accuracy and effectiveness of such systems (Hong, et al., 2013). However, even though research has been invested into understanding the design of recommender

systems in the staffing industry, few studies have investigated recommender systems that recommend candidates to potential employers, so-called candidate recommender systems.

When evaluating the performance of recommender systems, there are a range of properties to take into consideration, such as accuracy, diversity, novelty, and serendipity (Gunawardana & Shani, 2015). Traditionally, research has invested much effort into evaluating the accuracy of predictions based on historical data by calculating precision and recall. Recently however, more sophisticated metrics of accuracy have been proposed that have received less attention in research (Powers, 2011; Schröder, et al., 2011). Furthermore, to our knowledge, no comprehensive theory appears to exist regarding how to measure other properties of recommender systems and how these properties should be prioritized in relation to accuracy.

The purpose of this study is to evaluate different applications of candidate recommender systems. Furthermore, the aim is to assess the applied evaluation metrics in relation to how well they reflect the recommender system's actual performance and business goals.

The study is carried out through implementation and evaluation of several prototype recommender systems at Jappa, a Swedish start-up company providing an online recruiting platform. Based on that context, the following questions will be addressed:

> ***RQ1.*** *What are the key design choices when building a candidate recommender system?*
>
> ***RQ2.*** *What are the most relevant metrics and evaluation methods for assessing the performance of the recommender systems in the studied domain?*

In this report, first a brief summary of related work is presented to further introduce the landscape of relevant research and findings for candidate recommender systems. Thereafter, a more extensive theoretical framework is described, exploring the concepts necessary to understand, implement, and evaluate data-driven recommender systems. Then, chapter 4 explains the methodology for conducting this study. Chapter 5, describes the prototypes that have been implemented, and chapter 6 presents the results from the evaluation of those prototypes. Finally, a discussion section is presented along with the final conclusions of the study.

2 Related Work

This chapter presents a review of the results from the most relevant work related to recommender systems and algorithms in the domain of staffing and job recruitment. The purpose is to outline the state of the art regarding techniques used for creating job recommendations and candidate recommendations respectively. In the end of the chapter, the reviewed articles are reflected upon in relation to the problems that are investigated in this study.

1. From a literature review conducted by Al-Otaibi & Ykhlef (2012), it was concluded that recommender system technologies have achieved significant success in a broad range of applications and contexts. Within job recruitment, they identified a call for improvement in such systems, especially in the candidate-employer matching mechanism. They claim that a great potential solution for satisfying these needs lies in the application of machine learning algorithms. It was also concluded that hybrid-based recommender systems, using content-based and collaborative filtering approaches, is the best way to undertake the problem of job recommendation in general. They stated that this was due to the hybrid approach overcoming the limitations of applying any approach individually.
2. Heap et al. (2014) investigated how a job recommender system could be designed to match a candidate profile by using the candidate's previous job transitions during its career. This *transition model* approach was compared with a more common way of producing job recommendations to candidates, using the *cosine similarity* method to measure the suitability of a user profile towards job advertisements. The analysis was conducted by examining 2400 LinkedIn users, and the results yielded that the transition model outperformed the cosine similarity-based approach in terms of successful recommendations.
3. In a research made by Gupta & Gard (2014), efforts were put on examining the strategy of combining the description of a user profile with user preferences and behavioral attributes in a hybrid recommender system using both content-based and collaborative filtering approaches. The user preferences served as a filter to produce job recommendations that excluded the irrelevant jobs for the user. In their case, applying these techniques implied a significant increase in prediction accuracy in the recommender system.
4. In the paper of Hong et al. (2013), their investigation of different techniques in recommender systems also showed that a hybrid approach produced the best recommendations. Besides that, grouping different users using *clustering* techniques in such systems, turned out to increase the accuracy and effectiveness of the system that they proposed.

5. The research paper by Yu et al. (2011) is, to our knowledge, one of few papers that have investigated the application of a candidate recommender system. They introduced these types of recommender systems as *reciprocal recommender systems* in the field of recruitment. By combining data about users' attributes, explicit preferences, and implicit preferences, these characteristics supported the recommender system in the calculation of similarity between entities in order to create recommendations. The researchers showed that applying their proposed system in a candidate recruitment context increased the system's accuracy, in terms of precision and recall.

Considering the research papers reviewed above, it is revealed that the use of hybrid based job recommender systems frequently outperforms systems that are implemented using one individual approach, for example, a content-based or a collaborative filtering-based approach. Like previous research, this study is an attempt of analyzing the performance of such techniques, in the specific domain as described earlier. However, there is a conceptual difference between most of the recommender systems that appears among literature for job and recruitment contexts, and the recommender systems that will examined in this study. Just like considered by Yu et al. (2011), instead of producing job recommendations in a unidirectional way, from an employer or job posting towards a candidate, this study aims at evaluating different algorithms producing recommendations in the opposite direction: recommending a candidate to an employer. This practically makes the system a candidate recommender system, not a job recommender system. As mentioned earlier, the research behind such systems is very limited, and more research is needed to understand the appropriate design and evaluation choices in such systems from more perspectives than emphasized by Yu et al. (2011).

3 Theoretical Framework

This chapter introduces theory from literature related to the area of this study and explores concepts necessary to understand, implement, and evaluate data-driven candidate recommender systems in the studied domain.

First, the main concepts of data analytics are presented, and the application of recommender systems is defined in relation to the data analytics taxonomy. Thereafter, the main concepts of recommender systems, required for performing and understanding this study, are explained. Finally, technical aspects regarding the established methods for implementing and evaluating recommender systems in the studied domain are presented.

3.1 Data Analytics

Data analytics (DA) is the process of activities designed to collect and evaluate data to extract useful information (ISACA, 2011). When data sets are so large that they become unpractical for manual processing by humans, DA can be applied in different forms. For companies and other organizations, DA allows them to make better business related decisions (Mujawar & Joshi, 2015).

In literature, authors normally talk about three types of data analytics: *descriptive, predictive,* and *prescriptive* (Isson & Hariott, 2015; Delen & Demirkan, 2013). According to Isson & Hariott (2015), there is a fourth type called *diagnostic analytics*. However, only the three main types of analytics will be described here, as predictive analytics already implicitly includes diagnostic analytics, according to Delen & Demirkan (2013).

3.1.1 Descriptive

Descriptive analytics is the most common and basic form of DA. The main purpose of descriptive analytics is to provide a static understanding of the present and the past (Mujawar & Joshi, 2015) by describing different knowledge patterns using statistical methods, e.g. the mean value or the frequency of a certain attribute in a data set. By condensing large amounts of data, it can then be transformed into valuable parts of information. Applications of descriptive analytics occur in the form of data visualization tools such as dashboards, scorecards, KPIs and other reporting methods (Watson, 2014).

Applying descriptive analytics, the user seeks to aggregate data and transform it to information that explains the past, or the present. In other words, it aims at answering the following questions:

1. What happened?
2. What is happening?

For example, an HR-executive can perform retrospective analysis about how much money that was spent in hiring people from different occupational groups, regions, or age groups (or any other category) during the past years, and set it in relation to the business outcome of each group. Another example could, for instance, be to collect and analyze data about employers' daily activities in a "real-time" similar manner, which could support HR-divisions in their day-to-day operations and decisions.

3.1.2 Predictive

Predictive analytics is based on different learning methods that uses the understanding of the past, by uncovering relationships and patterns within large volumes of data (Eckerson, 2007), to make predictions about future events (Mujawar & Joshi, 2015). It can, for instance, be applied to make predictions about customers' behavior, or even when a factory floor machine is likely to break. Today, predictive analytics is mostly used within marketing for customer acquisition, campaign management, budgeting and forecasting models, cross selling etc. (Eckerson, 2007). Predictive analytics is also referred to as *exploratory* or *discovery* analytics (Watson, 2014).

By analyzing the past and the present, the user can extend descriptive analytics with predictive analytics, which will provide him with possible answers to the questions:

1. What will happen?
2. Why will it happen?

Consider the example given previously about the HR-executive in the context of descriptive analytics. By applying predictive analytics, the HR-executive might also predict the future business outcome gained from hiring people from a certain group, or the probability that a person will leave the company within the next months.

3.1.3 Prescriptive

Prescriptive analytics is the most advanced form of DA technologies (IBM Software, 2013), and probably the most valuable one. It uses data and mathematical algorithms to determine a set of high-value alternative courses-of-actions to help decision-makers perform the best actions given a set of desired outcomes (Delen & Demirkan, 2013). These systems can rely solely on data, solely on expert knowledge, or by combining both. This technique has commonly been studied within operations research or management sciences with the aim of optimizing the performance of a system (Sharda, et al., 2013).

Systems based on prescriptive analytics often include different recommender systems, or systems prescribing answers to a *yes/no* problem. Such system can also prescribe a specific "amount-property" of an item, e.g., how many items that should be bought in some context. Prescriptive analytics-based systems can even be used for prescribing how to design a plan

(e.g. a production plan). The prescriptions produced from these systems can be presented in a report, or computer-based user-interface as in the case of a recommendation, or may be applied directly through an automated decision rules system, e.g. an airline pricing system (Sharda, et al., 2013).

By applying prescriptive analytics, users' behavior, decisions and actions could be significantly impacted, since the prescriptive system will provide users with answers to:

1. What should I do?
2. Why should I do it?

Once again, consider the example of the HR-executive, in the context of predictive analytics. Apart from understanding the future behavior and performance of certain hires, the HR-executive will be advised by the system about which types of workers to hire, ideally preventing any irrational decisions or actions in the employment process. A high-quality candidate recommender system is such an application that could help to achieve this.

3.1.4 The Relationship Between Different Types of Data Analytics

The different types of DA are built on one another, and organizations that apply DA usually start with descriptive then moves to predictive and finally to prescriptive (Watson, 2014; IBM Software, 2013). This progression could be interpreted as an organization's level of maturity in applying DA. Isson & Hariott (2015) mention predictive analytics as an enabler of prescriptive analytics, and that they should be used in conjunction. Their view on prescriptive analytics, therefore, slightly differs from the view of Watson (2014) and IBM Software (2013) by trying to answer the question: "What should you do, knowing what will happen?". Mujawar & Joshi (2015) also emphasizes a similar view to the one of Isson & Hariott (2015). They claim that prescriptive analytics requires predictive analytics with two additional components: actionable data, and a feedback system from the actions taken. Considering the common theoretical frameworks, and their slight differences as described by different researchers and authors, our own interpretation of these concepts and frameworks are summarized in Figure 1.

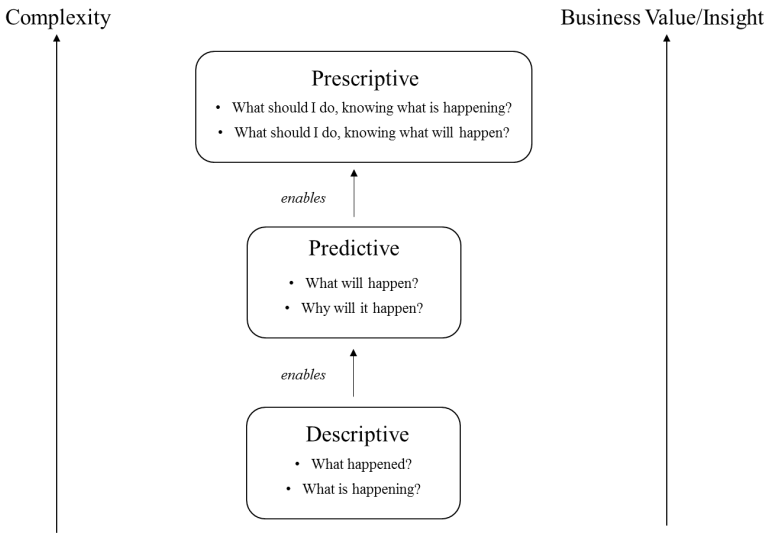

Figure 1: Framework of data analytics types.

3.1.5 Recommender Systems in Data Analytics

Due to the technological advancement in recent years, there has been an exponential rise in the use of online recruitment systems (Tripathi, et al., 2016). However, these systems commonly suffer the problem of information overload. When search engines fail to produce promising results from vast amounts of information, a recommender system is an application of data analytics that provides a solution to the information overload problem (Ricci, et al., 2015).

In general, job seekers upload their personal job profile into an online recruitment system, and the system then usually applies a filtering mechanism to generate a list of applicable jobs to the candidate. However, most of the time, these filtered job lists do not satisfy the job seekers needs and expectations (Tripathi, et al., 2016). For companies like Jappa that applies a system under the same principle, but from another perspective with employers searching for suitable candidates, similar problems arise. Therefore, instead of only applying trivial filtering techniques to produce job matchings, a recommender system will also take unstructured data into account, such as data derived from users' behavior in the system, to produce better results. The process of building a recommender system is, therefore, a data analytics process of collecting, filtering, managing, and transforming data using different techniques to produce high-quality recommendations to users.

3.2 The Concepts of Recommender Systems

There are many ways to design recommender systems. Entities and recommendation approaches are central concepts for understanding how a recommender system is constructed.

3.2.1 Entities

In recommender systems literature, researchers typically distinguish between two entities: *users* and *items*. Items are the objects, or entities, that are being recommended in the system. Items can vary in value and complexity and are thereby modeled differently depending on the domain (Ricci, et al., 2015). Typical low-complexity and low-value items are: books, CDs and movies while high-complexity items include: insurance policies, financial investments and jobs (Ricci, et al., 2015).

A user is an entity that receives recommendations of items from the system. Whenever more than one user is involved in the recommendation process, the specific user that receives recommendations is referred to as the *active user*. User profiles may consist of several types of information (Pazzani & Billsus, 2007). Information that could be used to model a user is age, gender, location and profession but also information about the behavior of the user such as ratings and purchases.

In the domain of this study, a user entity is an employer and an item is a candidate that is registered on the platform. When the term entity is used in this report, it can refer to either a user or an item.

3.2.2 Approaches

The core function of any recommender system is to predict an item that will be found useful by a user. The basis for recommendation, however, vary from one recommender system to another. There exist several taxonomies to distinguish between different approaches to build recommender systems. Ansari et al. (2000) identify two approaches, *content-based* and *collaborative filtering,* which are the most common in literature.

A **content-based recommender system** provides recommendations of items based on similarity to other items that a user previously has shown interest in (Ricci, et al., 2015). In order to establish similarity of items, these types of recommender systems analyze descriptions of items (Pazzani & Billsus, 2007). Content-based systems are therefore not dependent on analyzing similarity between users' and items' actions or behavior in the system, rather it focuses on the attributes that are associated with users and items. A classic application of a content-based recommender system aims to match attributes of a profile with attributes of other profiles to provide appropriate recommendations (Ricci, et al., 2015). In a movie database, for instance, content-based recommender systems can recommend movies from the same genre as a movie previously viewed or rated. In the domain of job seeking and recruiting, Lu et al. (2013) have attempted to create a partly content-based recommender system that maps similarity between candidate profiles by comparing their resume content. Furthermore, they calculate possible matchings between job seekers and jobs based on resume content that matches the job description.

A limitation of the content-based recommender systems is the dependence on information provided about the items. In many cases, there is not enough information, or attributes, assigned to an item to establish similarity. Without enough information, a content-based system will not be able to separate between items that a user does like from item that the user does not like (Pazzani & Billsus, 2007). Furthermore, the system often requires deep domain knowledge as the system developer needs to know what information is relevant to capture all aspects of an item (de Gemmis, et al., 2015).

In **collaborative filtering recommender systems**, similarity between users is established by analyzing the actions that the users have performed in the system (Ricci, et al., 2015). That is, the system will recommend items that are liked by other users that have a similar purchase or rating history. In the movie database example, the collaborative filtering recommender system would assume that two users have similar taste if they watch the same movies. Then, the system would use one of the user's preferences to create recommendations for the other user. Collaborative filtering is considered one of the most successful approaches for building recommender systems (Al-Otaibi & Mourad, 2012). The approach allows the recommender system to find unexpected recommendations that are not similar to items already viewed by the user in question (Linden, et al., 2003). As an example, a user can be recommended a movie from a genre they have not watched before, as recommendations are based on users with similar behavior of movie likings in the past.

A limitation of recommender systems that are based on collaborative filtering is that historical data of user actions and interactions is often sparse, especially in new systems, and it can therefore be difficult to make accurate predictions. Also, even if the system has a large amount of data as basis for calculation of similarities, the system will struggle to provide reliable recommendations for new users for whom it cannot yet establish a profile based on behavior (Al-Otaibi & Mourad, 2012).

Besides these two main approaches, content-based filtering and collaborative filtering, Burke (2007) mentions two additional approaches: *demographic* and *knowledge-based*. As the name suggests, demographic filtering bases recommendations on a demographic profile of the user. This is commonly occurring on websites that apply certain personalization based on language, country or age (Ricci, et al., 2015). Knowledge-based systems, on the other hand, uses domain knowledge on how item features meet customer needs to make recommendations. The similarity score in such a system can be interpreted as the utility of the recommendation for the user (Ricci, et al., 2015). The nature of knowledge-based systems implies that they work well at the beginning of deployment, but that the utility decreases if a learning component is not implemented.

When designing a recommender system, combination of different paradigms is an option to increase the quality of the predictions. These **hybrid recommender systems** use the advantages of one approach to avoid the disadvantages of another (Ricci, et al., 2015). In the above-mentioned approaches, the content-based technique's strength of accurate recommendations of new items could be combined with collaborative filtering's ability to learn from user behavior over time. Nilashi et al. (2015) claim that such an approach could achieve significant improvement in recommender systems.

For both content-based and collaborative filtering approaches, two common sub-approaches can be applied: *user-based* or *item-based* recommendations. Within literature, these two concepts originate from the application of collaborative filtering based recommender system, as explained by Sarwar et al. (2001). However, we do not see reasons for solely including these concepts in collaborative filtering based recommender systems, since they are well fitted in content-based approaches as well.

User-based recommender systems find users that are similar to the active user and then produce recommendations based on the preferences of those similar users. The item-based approach, instead, produces recommendations to the active user by finding similar items to those that the active user previously has expressed preferences for.

3.3 Data Sources and Categorization

Data is the foundation of DA (Isson & Hariott, 2015), and therefore represents the backbone of recommender systems. By collecting data from various sources, a recommender system can learn to make high-quality item suggestions to specific users. A data source could basically be any source providing the recommender system with data that helps it to determine the relevance score between a user and an item. Different sources are exploited depending on the context of the application. The data source to be exploited is also highly dependent on which recommender system technique that is used (Ricci, et al., 2015).

Isson & Hariott (2015) states that there are six major categories of critical data for workforce planning analytics in the related domain of talent sourcing and acquisition: talent data, market data, business data, economic and industry data, labor statistics data, and university graduation data. In this study, the data sources and categories have been divided into two main categories: *candidate data* and *employer data*. They will be defined and explained in the following sections.

3.3.1 Candidate Data

Candidate data is any type of data that can describe, or directly be related to, an individual candidate. Candidate data is mainly collected by input from the candidate upon registration to an online job platform and then stored in a database. The different candidate profile input

fields on the platform could be of mandatory or non-mandatory type, and consist of candidate *attributes* (explained in section 3.3.3) or *preferences* (explained in 3.3.5). Basic candidate data consist of personal data about the candidate's age, gender, education, or work experience etc., which provides a static description of the candidate. Additionally, data about candidates' personal traits could be used to enrich the candidate data profiling, as that type of data constitutes crucial factors that should be considered when matching candidates and employers (Buettner, 2014).

Data regarding the preferences of candidates is also important as it will provide information about the characteristics of a desired employer. Such data entry points could be salary level, industry field, or a particular job title. These preferences could for instance act as a filter when matching an employer, or a job, with candidates.

In addition to the personal information and preferences provided to the system by the candidate, Isson & Hariott (2015) state that another source of high importance for gathering specific candidate information is through the publicly available data, which is not provided by the candidate to the online job platform. That type of data includes the digital footprint that a candidate leaves on the web, or social media sites like Twitter, Facebook, and LinkedIn. According to some studies, 50 percent of the employee attrition could be explained by this type of external information (Isson & Hariott, 2015).

The third data category is *transactional data,* explained in section 3.3.4. This type of data can be collected by the data generated from automatic discovery of user behavior patterns, and user interactions with an employment website. This type of data collection and analysis is also referred to as *Web mining* (Srivastava, et al., 2005). The aggregated data about users' transactions from different sources is said to constitute the user model which profiles the user candidate (Billsus & Pazzani, 1997; Fisher, 2001).

3.3.2 Employer Data

Like candidate data, employer data is any type of data that can describe, or directly be connected to, a specific employer. This data includes general company information that has been submitted onto the platform. Also in this case, the employer data is mainly collected when employers register on an online job platform and specify their company attributes and preferences. A company data profile could consist of, for instance, data points about its industry field, geographical location, number of employees, company description keywords etc. A preference data point could be a candidate's years of work experience within a job field, gender, or age.

3.3.3 Attributes

Content-based recommendation techniques, in their simplest forms, aim at matching the attributes of a user profile with the attributes (and preferences) of an item profile (Ricci, et al., 2015). These attributes could be structured data that are explicitly typed into data entry fields (age, gender, industry field, company size, location etc.), or consist of keywords that are scanned and extracted from profile descriptions.

Attribute data that is referred to in this study, constitute a *static data* set, meaning that it does not change frequently over time, and thus describes a static view of the user or the item.

3.3.4 Transactions

In the domain of recommender systems, Ricci et al. (2015) refer to transactions as recorded interactions between the user and the recommender system. In contrast to attribute and preference data, transactions constitute a *dynamic data* set, which is highly variable over time. They argue that such information is useful to the system when making predictions about possible recommendations. The most popular form of transactions that a system collects is ratings, either implicitly or explicitly collected (Ricci, et al., 2015). When collecting ratings explicitly, a user is asked to provide an opinion on items. Implicit ratings on the other hand, is collected by analyzing users' actions. For example, if a user searches for appropriate candidates to certain a job, and then chooses to look more closely at one candidate from a long list, that action indicates that the user is interested in that candidate to some extent. Transactions can be considered an important complement to attributes and preferences when it comes to predicting appropriate recommendations to users, since it provides the system with clear indications of what the user likes and does not like. This data can help recommender systems to predict users' preferences in the future.

3.3.5 Preferences

In the domain of the study, both candidates and employers can have *explicit preferences* expressed in their profiles regarding what types of matches they are interested in. One example is that a candidate can have a preference to work as a machine operator and is therefore not interested in job offers in the retail or warehouse sector. Felfernig et al. (2015) refer to this type of filtering as constraint-based recommenders, recommenders that take into consideration domain knowledge in the form of explicit rules about how to match users and items. Like attributes, preferences also belong to the static data set of a user profile.

There are also *implicit preferences*, which some CF-based algorithms try to predict and base their recommendations on. These implicit preferences are derived from users' interaction history with the system.

3.3.6 Context

Traditional recommender systems have attempted to create recommendations solely based on models of users and items, taking characteristics and previous behavior into consideration. There are, however, systems that also take the concept of context into account to make recommendations more compelling and useful (Linas, et al., 2012). For instance, traditional systems might take previous ratings into account when making predictions, while a context-aware system would also consider contextual factors, such as time and location of a rating. Figure 2 shows how context can affect what item is recommended to a user. Without the context (in this case, the day that the user rated the item), the system would not be able to know whether to recommend Item B or Item D. By including the contextual information of previous ratings, however, the system can provide a suitable recommendation depending on what day it is.

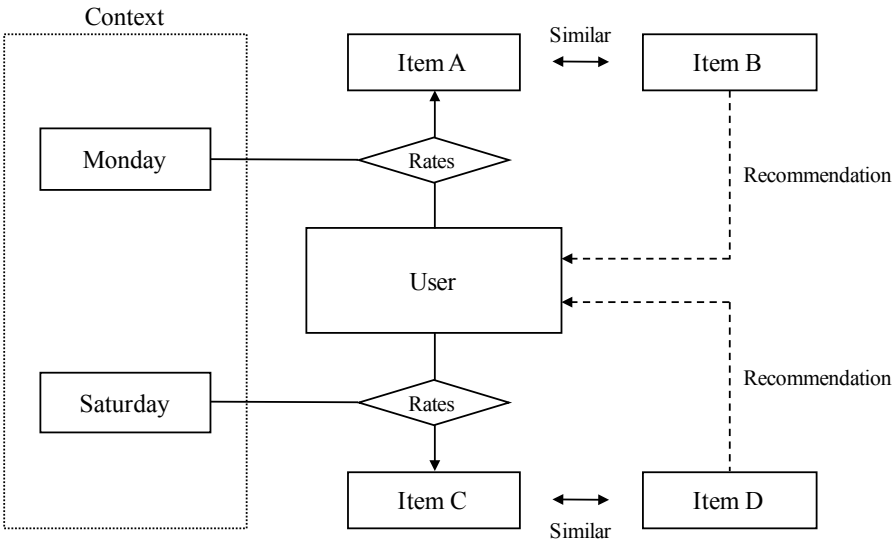

Figure 2: Illustration of a context-aware recommender system.

Adomavicius & Tuzhilin (2015) exemplify this in a case of a movie theater, where a customer arguably would like to be recommended one type of movies when looking for something to watch on a Saturday night with a date, and another type of movies on a weekday with friends. In the domain of staffing and recruiting, a possible application for a context-aware system is to consider changes in demand depending on season.

3.4 Dataset Transformation

Machine learning algorithms often require input in the form of vectors consisting of numerical values. However, features that represent entities are often not numerical, which demands a transformation of the data set in order for it to be used in many machine learning algorithms.

A standard way of classifying text into numerical values is the *bag-of-words approach* (Attenberg, et al., 2009). That approach creates a large vector containing all textual values, such as every word in a text, this vector is called the dictionary. Then, each entity is represented by an equally large vector of numerical values. In the case of words in a text, each value in the entity vector would represent the number of occurrences of a word in the dictionary. The method solves the problem of representing textual information in a numerical vector with the drawback that each vector will be very sparse as most words do not appear in a specific text (Attenberg, et al., 2009). The same principle can be applied for categories instead of occurrence of words. Each entry in the dictionary is then a categorical value, and once transformed, the entity vector will consist of binary values indicating the occurrence of categorical values.

Another technique that can be applied to represent textual, or categorical, features in a numerical vector is referred to as *the hashing trick* (Attenberg, et al., 2009). In this method, the dictionary is removed and instead a function is used that uses hashing to map a string to an integer (Ganchev & Dredze, 2008). This reduces the size of the vector but introduces the possibility of collisions between mappings (two different features might be randomly assigned the same integer). However, Ganchev & Dredze (2008) have shown that this approach can drastically limit the vector size with only minor impact on the accuracy of the learning algorithm.

An approach to reduce the size of a vector, after having transformed it into a vector of numerical values, is to eliminate features through dimensionality reduction. Amatriain & Pujol (2015) define two problems that dimensionality reduction helps to resolve:

1. *Data sparsity*, the fact that vectors are huge and mostly empty and thereby taking up an unnecessary amount of memory;
2. *The curse of dimensionality*, the fact that distance between points in vectors become less meaningful as the number of dimensions increase.

A commonly occurring algorithm to reduce dimensionality in a vector is *principal component analysis* (PCA). PCA is a statistical method to find patterns in high-dimensionality data sets (Amatriain & Pujol, 2015). The goal of the algorithm is to find how much of the variance each feature contributes with. The logic is that if a feature does not add variance, it does not explain similarity or dissimilarity between entities and therefore it can be eliminated without big impact on the final recommendation.

3.5 Algorithms and Techniques

Producing recommendations involves applying statistical and knowledge discovery techniques in a database of user data (Sarwar, et al., 2000). There exists a collection of

established techniques and algorithms that can be applied to solve the challenges of producing useful recommendations to users. In this study, the solution of the problem is investigated by the application of different classification, clustering and nearest neighbor-based algorithms, all which belong to the set of common machine learning (ML) algorithms in recommender systems. According to Portugal et al. (2015), ML is about using computers to simulate human learning by allowing them to identify and acquire knowledge (learn) from the real world. Based on that learning, a computer will improve the performance of a task, in this context, a recommendation.

Depending on the learning type of the applied algorithm, it can be classified into one of four main categories (Portugal, et al., 2015): *supervised, unsupervised, semi-supervised, and reinforcement learning.*

1. Supervised learning is applied when algorithms are provided a training set with input values that are mapped to "correct" answers. The training set and the answers are used to create a function that can map new input values to any of the existing labels.
2. Unsupervised learning does not utilize a training set. Based on input data, unsupervised machine learning algorithms need to learn from this data set by themselves, as there are no "answers" in the data set as for the supervised algorithms.
3. Semi-supervised algorithms use incomplete training sets from which predictions are made.
4. Reinforcement learning is when learning algorithms produce desired results based on external feedback. When the algorithm performs in a certain way, it receives either a positive or negative feedback and adjusts its parameters accordingly.

Through the application of classification algorithms, a model can be built to predict which item to recommend for a specific user (Zhang & Iyengar, 2002). In machine learning, classification is used to identify a category (class) to which a new entity belongs, based on a training data set. Classification can done be via two procedures: binary or multiclass classification (Har-Peled, et al., 2002). The former only involves assigning a new entity to one of two classes, while the latter is a question of assigning a class to a new entity when more than two classes are involved. In the studied domain, classification assigns a candidate (item), or a group of candidates, which will be the class, to an employer (user). An entity is represented as a feature vector, defining the properties of an entity. Every feature in the feature vector can be of binary, integer-valued, real-valued, ordinal, or categorical data type. Depending on the algorithm of implementation, discrete or continuous numeral values are preferred.

In the following sections, algorithms and techniques that were implemented and evaluated in the study are described in more detail. Three classifiers were chosen: *Bayesian, Decision*

tree, or *Nearest neighbor*-based classifiers. In recommender system and machine learning development, they belong to a set of the most common techniques, due to their recent popularity and their relative low complexity in calculation and implementation (Portugal, et al., 2015). Clustering algorithms, used to group entities into clusters based on similarity (Kim & Ahn, 2008), which supports other algorithms in recommendation tasks, are also introduced. Finally, a certain kind of collaborative filtering technique using a user-item preference matrix, as described by Sarwar et al. (2000), will be also be presented. According to Isinkaye et al. (2015), this is the most common and mature technique for producing recommendations.

3.5.1 K-Nearest Neighbor

K-nearest neighbor (k-NN) algorithms are unsupervised algorithms, and are very popular in recommender systems, due to their simplicity and efficiency (Ricci, et al., 2015). These algorithms are applied in both content-based and collaborative-filtering approaches to compute and find the most similar entities of a given entity in the system. Let $T = \{e_1, e_2, \dots, e_n\}$ be a set of observations with e_i being an entity consisting of a set of features $\{f_1, f_2, \dots, f_n\}$ defined as numeric values. The k-NN algorithm will take a new entity $\hat{e}$ and compute the *k* most similar entities of $\hat{e}$ in *T* using a similarity function. The results are then ranked, and the selected set of *k* most similar users are called the *k*-nearest neighbors.

Common similarity functions used in k-NN algorithms for recommender systems are the *Euclidean distance,* the *Cosine similarity,* and the *Jaccard similarity* (Qamar & Gaussier, 2012). Another popular similarity method used in CF-based recommender algorithms is the *Pearson correlation* (Hahsler, 2011).

The distance *d* using the Euclidean distance between a vector e_i and $\hat{e}$ is calculated as:

$$d_{ED}(e_i, \hat{e}) = \sqrt{\left(f_1 - \hat{f_1}\right)^2 + \left(f_2 - \hat{f_2}\right)^2 + \cdots + \left(f_n - \hat{f_n}\right)^2} \qquad \textbf{Eq. 1}$$

When comparing the similarities between two entities, it can be more meaningful to use the *Normalized Euclidean distance*:

$$d_{NED}(e_i, \hat{e}) = \frac{d_{ED}(e_i, \hat{e})}{|e_i|} \qquad \textbf{Eq. 2}$$

According to some researchers, the cosine similarity should be preferred over Euclidean distance when dealing with non-textual data (Qamar & Gaussier, 2012). It is calculated by applying the following formula:

$$sim_{cosine}(e_i, \hat{e}) = \frac{\sum_j f_j \hat{f}_j}{\sum_j f_j^2 \times \sum_j \hat{f}_j^2} \quad \textbf{Eq. 3}$$

where $sim_{cosine}(e_i, \hat{e}) \in [0,1]$ and j being the index of the jth feature of an entity.

When comparing vectors consisting solely of binary data, it is only interesting to consider the intersection of features between entities, then the Jaccard similarity is a favorable measure (Hahsler, 2011). It is calculated as:

$$sim_{jaccard}(e_i, \hat{e}) = \frac{|e_i \cap \hat{e}|}{|e_i \cup \hat{e}|} \quad \textbf{Eq. 4}$$

The Jaccard similarity represents the intersection of the entities e_i and $\hat{e}$ in relation to their union.

3.5.2 K-Nearest Neighbor Classifier

K-Nearest Neighbor classifier (k-NN classifier*)* is one of the oldest and simplest classification rules (Qamar, et al., 2008). It is a non-parametric function (Ouyang, et al., 2006), meaning that it does not involve estimation of parameters. The k-NN classifier is based on the regular k-NN algorithms as described in the previous section, but the two should not be confused. The k-NN classifier can be seen as an extension of the regular k-NN algorithm, by adding a classification rule. K-NN classifiers identify *k* observations in the training data set that are similar to a new observation, and then assign a class to the new observation based on the classification of entities in the training set. More formally, that is, having a training data set of observations $T = \{e_1, e_2, \dots, e_n\}$ with e_i being an entity with a set of features $\{f_1, f_2, \dots, f_n\}$ defined as numerical values, and each e_i belonging to a class c, then we want to assign some class c in *T* to a new observation $\hat{e}$. For example, with $k = 1$ (1-NN) we have the simplest case that finds the single nearest neighbor relative to $\hat{e}$, then the class c of the single nearest neighbor of $\hat{e}$ will be assigned to $\hat{e}$.

For the general k-NN case, a class membership *c* is assigned to $\hat{e}$ by applying a *majority decision rule.* For example, $\hat{e}$ can be classified with respect to the most frequent class among its *k*-nearest neighbors. A weighting scheme can also be applied to the neighbors by giving them a weight, e.g., using the relation $1/d$, where *d* is the distance to a neighbor. The advantage of this technique is that higher values of *k* provide smoothing that reduces overfitting risks due to noise in the training data set (Ouyang, et al., 2006). In typical applications, a value of *k* in units of tens are preferable. Having a training set of size *N*, we can notice that if $k = N$, the information from the independent variables will be ignored, and $\hat{e}$ will be assigned the class c that is most frequently assigned to the observations in T.

An appropriate value of *k* can be chosen by computing and comparing the error rates for different *k*-values from a training and validation data set.

3.5.3 Naïve Bayes Classifier

Naïve Bayes classifiers are probabilistic learning methods based on *Bayes Theorem,* with the "naïve" assumption of independence between features, meaning that one feature does not influence another feature. They are suitable for cases with a high input of features. Applying a Bayesian classifier, the probability of an entity e, representing a set of features $\{f_1, f_2, \dots, f_n\}$, and belonging to a class c_j is computed as:

$$P(c_j|e) = \frac{P(c_j)P(e|c_j)}{P(e)}$$

$$\Leftrightarrow$$

$$P(c_j|f_1, f_2, \dots, f_n) = \frac{P(c_j)P(f_1, f_2, \dots, f_n|c_j)}{P(f_1, f_2, \dots, f_n)} = \frac{P(c_j)\prod_{i=1}^{n} P(f_i|c_j)}{P(f_1, f_2, \dots, f_n)}$$

Since $P(f_1, f_2, \dots, f_n)$ is a constant determined by the input of a given entity e, the following classification rule can be used:

$$P(c_j|f_1, f_2, \dots, f_n) \propto P(c_j)\prod_{i=1}^{n} P(f_i|c_j) \qquad \textbf{Eq. 5}$$

An estimation for $P(c_j)$ can be computed from a training set T of entities as:

$$\hat{P}(c_j) = N_{c_j}/N_T \qquad \textbf{Eq. 6}$$

where N_{c_j} is the number of entities e in T that belongs to a class c_j, and N_T the total number of documents in T. $P(f_i|c_j)$ represents the frequency of a feature f_i in a class c_j in relation to the number of total features that belongs to a class c_j. There are several different types of naïve Bayes classifiers, which mainly differs in their assumptions regarding the distribution of the parameter $P(f_i|c_j)$. Three common types of naïve Bayesian classifiers used in machine learning are *Gaussian, Multinomial,* and *Bernoulli*-based types.

Gaussian implements a naïve Bayesian classifier based on the assumption that the occurrence of features for a specific class follow a *normal distribution* $N(\hat{\mu}_{C_j}, \hat{\sigma}_{C_j})$.

Multinomial is used for multinomially distributed data, ideally when it is interesting to consider how many times a feature occurs within a document that belongs to specific class.

Multinomial naïve Bayes algorithms implements a smoothed version of $P(f_i|c_j)$, similar to the estimation of $\hat{P}(c_j) = N_{c_j}/N_T$, to estimate a parametrized distribution represented by the vector $\theta_{C_j} = (\theta_{c_{j1}}, \theta_{c_{j2}}, \dots, \theta_{c_{jn}})$, where n is the number of features belonging to a class c_j. The parameters $\theta_{C_{ji}}$ are calculated as:

$$\hat{\theta}_{C_{ji}} = \frac{N_{c_{ji}} + \alpha}{N_T + \alpha N_{c_j}} \qquad \textbf{Eq. 7}$$

Setting the parameter $\alpha \geq 0$ will prevent zero-probabilities when a feature is not present in the training data. The standard value is $\alpha = 1$, and is called *Laplace smoothing*.

Bernoulli is a special case of the multinomial version, and therefore employs the same underlying assumptions as for the multinomial classifier. However, the Bernoulli classifier applies the logic that every feature has a binary value in a feature vector. Instead of counting how many times a feature occurs, it accounts for if a feature occurs or not for a class. The decision rule for Bernoulli naïve Bayes is defined as follows:

$$P(f_i|c_j) = P(i|c_j)f_i + \left(1 - P(i|c_j)\right)(1 - f_1) \qquad \textbf{Eq. 8}$$

3.5.4 Decision Tree Classifier

The decision tree classification method constructs decision trees in order to classify an entity, based on its set of features, to a set of predefined classes (Rokach & Maimon, 2007). A decision tree classifies instances, which are specific combinations of a feature set $\{f_1, f_2, \dots, f_n\}$, by sorting them from a root node to a leaf node that will contain the class of that instance (Mitchell, 1997). In other words, each node represents a feature in a feature vector e_i and each branch from a node represents a possible value for that feature. Nodes and branches are created for every feature and feature value respectively, so that each vector can be associated with a leaf in the tree. The class of a vector is then saved in a leaf that the vector leads to. The basic principle is that, from a training data set T, this process will be applied to every feature vector e_i in T, defining all possible paths from the top root node to the leaf nodes. By using a training data set, decision rules can be learned from it, that predicts the class of a new test entity $\hat{e}$. Each path from the root node to the leaf node corresponds to a conjunction of feature instances, and the tree itself a disjunction of these conjunctions (Mitchell, 1997). An example of a compiled decision tree classifier for a training set T_{ex} as defined by Table 1 is illustrated in Figure 3.

Table 1: Example of a training set used as input to a decision tree algorithm.

Entities	Features		Class
e_i	$city\ (f_1)$	$education\ (f_2)$	$industry\ (y)$
e_1	gothenburg	mechanic	automotive
e_2	gothenburg	chef	food
e_3	stockholm	mechanic	automotive
e_4	gothenburg	mechanic	automotive

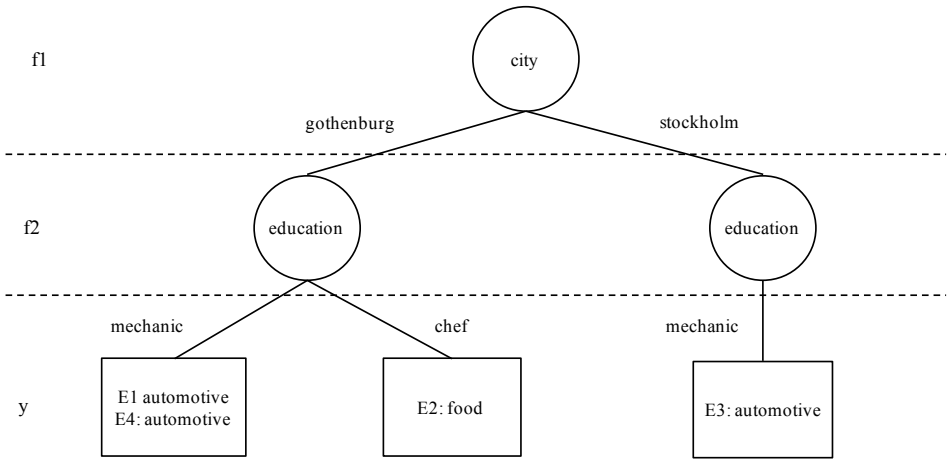

Figure 3: Example of a decision tree classifier.

The decision tree in Figure 3 represents one instance generated from the training set example. The representation of the decision tree can be varied by choosing another feature for the top root node, and by branching nodes to other nodes in other possible sequences. In this case, the subdivision of features was made in the same order as they occur in the feature vector e_i.

3.5.5 K-means Clustering

K-means clustering is a partitioning method that splits a data set of items into an arbitrary number of *k* subsets so that the items in each subset are as close to each other as possible and as far as possible from items belonging to other subsets (Amatriain & Pujol, 2015). The algorithm is widely used because of its simplicity of application (Kim & Ahn, 2008). Kim & Ahn (2008) explain k-means in four iterative steps.

1. *K* number of centroids are chosen randomly for the initial clusters.
2. Each item is assigned to the closest centroid, thus forming *k* clusters.
3. For each cluster, a new centroid is calculated based on the *items* in that cluster.
4. Step 2 and 3 is repeated until the clusters stop changing or until another stop condition is satisfied (such as a pre-defined maximum number of iterations).

When the iteration is completed, each cluster is defined by its members and its final centroid (Amatriain & Pujol, 2015). Several different distance metrics can be used when computing which items are close and which items are far away from each other, such as Euclidean distance or cosine similarity.

Xue et al. (2005) and O'Connor & Herlocker (1999) present two different applications of clustering in the context of recommender systems. Xue et al. (2005) use clustering to create groups of items while O'Connor & Herlocker (1999) use it as pre-processing step for neighborhood formation amongst users.

3.5.6 Collaborative Filtering Algorithm Using a User-Item Preference Matrix

The goal of CF-based techniques is to recommend unknown items for a particular user in the system, based on the user's and its similar users' previous preference data in the system. Sarwar, et al. (2001) apply a certain well-established approach of a collaborative-filtering based recommender system for achieving this goal, through a user-item preference matrix. Having a set of users $U = \{u_1, u_2, \dots, u_m\}$ and a set of items $I = \{i_1, i_2, \dots, i_n\}$, each user vector u_j contains the user's preferences for all items in *I,* while i_k contains all the users' preferences for it. Null valued preferences are included in both sets. A preference from a user u_j towards an item i_k is denoted as p_{jk}. The preference scores are stored in a user-item matrix *P* as illustrated in Figure 4, with each row representing a user u_j, and each column representing an item i_k.

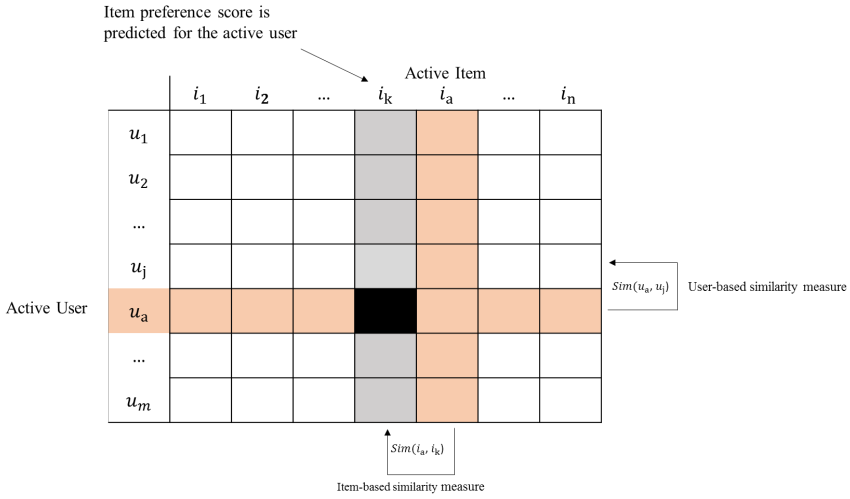

Figure 4: Illustration of the collaborative filtering user-item matrix.

Through literature, user preference scores are generally referred to as ratings (Hahsler, 2011). In this study however, they are called preferences since users' opinions and interactions towards items do not necessarily need to be ratings, as in the case of movie recommender systems. Applying CF techniques in the context of this study, a preference is

a denotation that a user has explicitly expressed an interest in an item to some degree (preference score). The preference score can be a value according to a certain numerical scale, indicating the preference strength of a user towards an item, or of Boolean kind, only denoting if a user has had a preference or not towards an item.

The null scored preferences of users, which are the missing values from users, normally constitute a large fraction of the total number of cells, since users mostly have preferred significantly fewer items than there are available items in the system.

CF can be applied using either a *user-based* or an *item-based* approach, both of which are well-established and proven methods in recommender systems (Herlocker, et al., 2004). The former is based on the assumption that users with a history of similar preferences also will rate and prefer new items in a similar way. The latter CF approach was first introduced by Sarwar et al. (2001), and predicts preferences by selecting the most similar items of an active item i_a for an active user u_a. Having in mind the matrix in Figure 4, the main difference between the two approaches is the application of row-wise or column-wise similarity measures for the user-based respectively item-based approach.

The task of the user-based collaborative filtering recommender algorithm is to predict the preference scores for the missing values of an unknown item set I_a to an active user u_a. The next step, as emphasized by Sarwar et al. (2001), is to rank the predicted preference scores, and then recommend the top-*N* items to the user. The prediction of preference scores is made by first finding the most similar users of an active user, by applying a k-NN algorithm or threshold criteria. Different k-NN algorithms as those described in section 3.5.1, can be applied. Once the set of k most similar users $N(a)$ is found, the missing ratings of I_a from u_a are computed by aggregating the preference scores from the users in $N(a)$. The simplest way to aggregate a prediction preference score $\hat{p}_{ak}$ from a user u_a towards an unknown item i_k, is to average the preference scores of an item i_k from all the similar users, using the following formula:

$$\hat{p}_{ak} = \frac{1}{|N(a)|} \sum_{j \in N(a)} p_{jk} \qquad \textbf{Eq. 9}$$

However, the fact that some users in the neighborhood are more similar to the active user than to other users, can be taken into account by introducing weights into Eq. 9 (Hahsler, 2011) as follows:

$$\hat{p}_{ak} = \frac{1}{\sum_{j \in N(a)} s_{aj}} \sum_{j \in N(a)} p_{jk} s_{aj} \qquad \textbf{Eq. 10}$$

where $s_{aj} \in [0, 1]$ is the similarity value between the active user u_a and a neighbor user u_j in $N(a)$.

An example of the prediction procedure in a user-based approach, using 5 users and 6 items, is illustrated in Figure 5. Given the 5×6 users-item preference matrix below, a k-NN algorithm is applied to find the most similar users (grayed out rows) to the active user (colored row) in the matrix.

	i_1	i_2	i_3	i_4	i_5	i_6	
u_1	0	0	0	1	0	3	
u_2	3	3	2	1	0	0	1th NN
u_a	4	3	**1.5**	2	**1**	**0**	Active User
u_4	0	0	3	0	4	2	
u_5	4	0	1	0	2	0	2th NN

Figure 5: User-based collaborative filtering example for predicting a user's preferences for unknown items.

The k-NN algorithm performs a row-wise similarity comparison of the active user u_a with all the other users u_1, u_2, u_4 and u_5 in the matrix. Setting the $k = 2$, the users u_2 and u_5 are computed as the most similar users to u_a, given their user preference score vectors. The missing preference ratings for items i_3, i_5 and i_6 are calculated by applying the average preference scores method to the active user's nearest neighbors. This implies $\hat{p}_{a3} = \frac{(2+1)}{2} = 1.5, \hat{p}_{a4} = \frac{0+2}{2} = 1$, and $\hat{p}_{a6} = \frac{0+0}{2} = 0$, which are highlighted as bold values in the matrix. Given $N = 2$, the algorithm will recommend the two items with highest predicted preference scores to the active user, namely item i_3 and i_6.

3.6 System Evaluation

In literature, there exists a wide range of options for evaluating recommender systems. In some cases, a recommender system can be evaluated on basis of how well it achieves its overall goals (Gunawardana & Shani, 2015). For instance, a recommender system on an e-commerce site can be evaluated by measuring the revenue of the platform, with and without the implemented recommendation engine. In many cases, however, it is useful to evaluate specific properties of recommender systems to compare the performance of different algorithms, or to improve properties where algorithms do not perform as well as intended (Gunawardana & Shani, 2015).

In the early days of recommender systems research, much effort was invested into measuring the accuracy of predictions (Herlocker, et al., 2004). However, it is now widely agreed that while accuracy is a crucial metric, it is insufficient to serve as the only indication

of a good recommender system (Gunawardana & Shani, 2015). Other popular metrics, more recently discussed in literature include: Coverage, Serendipity, Diversity, Confidence, Novelty, Risk, Privacy, Scalability and Attractiveness (Pu, et al., 2011; Adomavicius & Tuzhilin, 2015; Gunawardana & Shani, 2015; Ge, et al., 2010). In this study, three properties of the systems have been evaluated: *Accuracy*, *Coverage*, and *Diversity*.

3.6.1 Accuracy

Accuracy is the most discussed property in recommender systems literature (Gunawardana & Shani, 2015). This is because many recommender systems are built on top of a prediction engine that is trying to predict whether an item will be liked or not liked by a specific user, and recommends items accordingly. When recommending a list of items to a user, a common way to calculate the accuracy is to compare the predicted items to a list of items that we know that the user likes. When doing such a comparison, each item can be categorized into a True-Positive, False-Positive, True-Negative, or False-Negative as showed in Table 2. Such a matrix is often referred to as a confusion matrix (Burke, et al., 2011).

Table 2: Confusion Matrix

	Recommended	**Not Recommended**
Good Recommendation	True Positive (tp)	False Negative (fn)
Bad Recommendation	False Positive (fp)	True Negative (tn)

From the confusion matrix, two metrics can be calculated that indicates how well the system performs in terms of accuracy:

$$Precision = \frac{tp}{tp + fp} \quad \textbf{Eq. 11}$$

$$Recall = \frac{tp}{tp + fn} \quad \textbf{Eq. 12}$$

As can be derived from Eq. 11 and Eq. 12, *precision* measures the proportion of the recommended items that are considered useful to the user. Therefore, in systems where each recommendation is an investment of time or resources for the user, or where it for other reasons is important not to show irrelevant recommendations, the precision metric can be an important indication of the system's performance. In other systems, the users might not mind looking through a longer list of irrelevant items before finding one that is of their liking. In those cases, *recall* can be a better choice of metric as it measures the fraction of items liked by the user that the recommender system is able to find.

When producing recommendation, the strongest recommendation, based on some criteria, is often the first one to be recommended. This means that the larger the list of recommendations is, the weaker the recommendations in the list become. So, while recall always increases with increased number of recommendations, precision tends to decline for larger values of *N*. Therefore, when comparing the performance of different algorithms, they should be compared for equal values of *N* (Gunawardana & Shani, 2015).

The *F-measure* summarizes the precision-recall performance into a single metric and is used to find the harmonic mean of the equally weighted precision and recall (Ge, et al., 2010).

$$F = \frac{2 \cdot precision \cdot recall}{precision + recall} \quad \textbf{Eq. 13}$$

When using the F-measure to evaluate the accuracy of different algorithms, or systems, it is important to notice that the metric does not take into consideration the quantity of the true negatives. Davis & Goadrich (2006) reflects over this fact but concludes that this is not an issue as long as there is a fixed number of positive and negative samples for all algorithms that are to be evaluated. Their logic is that if the number of positive and negative samples are fixed, the false negatives can be uniquely determined given the three other entries in the confusion matrix. However, this does not hold true when the evaluated algorithms do not have the same number of positive and negative samples which often is the case.

An illustration of the problem from another context is: Say that your mission is to blindly pick green balls from a pool consisting of both green and red balls. In a scenario where you pick five balls in total, precision would measure the fraction of the balls you picked that were indeed green while recall would measure how many of the green balls in the pool you were able to find. The problem here is that without knowing how many red balls were in the pool to begin with, it is hard to say whether your precision and recall were good or bad.

In order to work around this problem, Powers (2011) proposes two complementary metrics: *inverse precision* and *inverse recall*. The two metrics are given by Eq. 14 and Eq. 15.

$$Inverse\ Precision = \frac{tn}{fn + tn} \quad \textbf{Eq. 14}$$

$$Inverse\ Recall = \frac{tn}{fp + tn} \quad \textbf{Eq. 15}$$

Inverse precision can be expressed as the probability that an item which is not recommended is indeed irrelevant and inverse recall as the probability that an irrelevant item is indeed not recommended (Schröder, et al., 2011). In the same scenario as before, a way to paraphrase

your mission is to say that you are to blindly pick five balls and your mission is not to pick any red balls. It then makes sense to measure inverse precision as the fraction of balls left in the pool that is indeed red and the inverse recall as the fraction of the red balls that you managed not to pick.

So, by looking at precision and inverse precision at the same time, we can evaluate how well the system can predict negatives or positives. In the same way, by looking at both recall and inverse recall we can draw conclusions of how well the system performs in terms of finding a user's preferred or not preferred items. Powers (2011) suggests that precision and inverse precision are combined into *markedness,* and recall and inverse recall are combined into *informedness* as:

$$Markedness = Precision + Inverse\ Precision - 1 = \frac{tp}{tp + fp} + \frac{tn}{fn + tn} - 1 \quad \textbf{Eq. 16}$$

$$Informedness = Recall\ + Inverse\ Recall - 1 = \frac{tp}{tp + fn} + \frac{tn}{fp + tn} - 1 \quad \textbf{Eq. 17}$$

Finally, in an attempt to combine all aspects of accuracy into one unbiased metric, the Matthews correlation coefficient, MCC, combines markedness and informedness by calculating their geometric mean, ranging between -1 and +1. A value of +1 represents a perfect prediction and -1 represents a total disagreement between the observation and the prediction (Schröder, et al., 2011).

$$MCC = \pm\sqrt{Informedness \cdot Markedness} = \frac{(tp \cdot tn) - (fp \cdot fn)}{\sqrt{(tp + fn)(fp + tn)(tp + fp)(fn + tn)}} \quad \textbf{Eq. 18}$$

Matthews correlation coefficient can be compared to the F-measure in the way that they strive to provide a single metric to evaluate the accuracy of predictions in a recommender system. MCC, however, has the advantage of avoiding the introduction of underlying biases into the results (Powers, 2011; Schröder, et al., 2011).

3.6.2 Coverage

While accuracy measures the system's ability to predict user preferences, it is important to take coverage into account when evaluating the results. Some recommender systems can provide highly accurate recommendations, but only for a small portion of the items, often due to lack of data (Gunawardana & Shani, 2015). Other systems can recommend all items

but only to a small number of users, users that have multiple recorded interactions with the system. These two types of coverage are referred to as *item space coverage* and *user space coverage*.

Item space coverage, also referred to as prediction coverage, can be explained as the proportion of available items that the system is able to recommend (Ge, et al., 2010). It can be calculated as:

$$Item\ space\ coverage = \frac{Items\ that\ the\ system\ can\ recommend}{All\ available\ items} \quad \textbf{Eq. 19}$$

Item space coverage is highly dependent on the design of the recommendation engine (Ge, et al., 2010). For instance, a collaborative filtering system that requires user interaction with items before they can recommend them is likely to have a lower item space coverage than a content-based system that can recommend items based on preexisting attributes.

User space coverage can be defined as the proportion of all users that the system can provide recommendations for (Gunawardana & Shani, 2015).

$$User\ space\ coverage = \frac{Users\ that\ can\ receive\ recommendations}{All\ users} \quad \textbf{Eq. 20}$$

Much like item space coverage, user space coverage varies with the technique and design of the recommender system. Once again, systems that require more user interactions to map behavior is more likely to have a lower user space coverage.

There exists a potential trade-off between systems with high coverage and high accuracy (Gunawardana & Shani, 2015). By lowering the threshold of data required to make recommendations a system can increase its coverage but the accuracy is likely to suffer as a result.

3.6.3 Diversity

In general, diversity is defined as the opposite of similarity (Gunawardana & Shani, 2015). In the context of recommender systems, sometimes it is desirable that the system recommends items with high diversity when making multiple recommendations. Diversity is most often calculated as the sum, average, min, or max distance between item pairs in a list of recommendations (Gunawardana & Shani, 2015). Furthermore, Gunawardana & Shani (2015) state that the item-item similarity measure used to calculate the diversity can be different from the one possibly used to create the list of recommendations. This means that both algorithms that use item-item similarity when providing recommendations and those which do not can be compared using the same method.

Increased diversity may come at the expense of decreased accuracy (Zhang & Hurley, 2008) and, therefore, the trade-off between the two measures can be an interesting one to observe.

3.7 Ethical Considerations

In Sweden, processing of data is regulated by the *Personal Data Act* (Datainspektionen, 1998). The intent of the Personal Data Act is to protect people against violation of personal integrity by processing of personal data. In this context, processing means everything one does with personal data, such as: collection, registration, storage, disclosure by transfer, compilations, or joint processing.

The act covers processing of data, with aid from computers, of information that is directly or indirectly referable to a living person. The act does not, in principle, cover journalistic, artistic, or literary activities. Certain subsets of personal data are considered sensitive data and more strict regulations apply to processing of such information. Without explicit consent, or previous publication, it is prohibited to process data that discloses race or ethnic origin, political opinions, religious or philosophical convictions, membership of trade unions or data related to health or sexual life. Furthermore, it is prohibited for bodies other than authorities to process data regarding violations of laws, judgment in criminal cases, penal procedural coercive measures, or administrative deprivations of liberty.

Personal data, of non-sensitive nature, may be processed with the consent of the person if:

1. the process is lawful;
2. the process is in accordance with good practice;
3. the use is not incompatible with that for which the data was gathered;
4. the process is necessary;
5. the data is up-to-date;
6. the data is gathered for specific purposes;
7. and if the data is not kept for a longer period than necessary.

In conclusion, data of non-sensitive nature may be processed with permission from the user, but for sensitive data, as defined by Datainspektionen (1998), more strict regulations apply.

4 Methodology

The following chapter explains the research methodology used in this study. First, the chosen research approach is presented. Then follows a description of how the feature selection, algorithm selection, and model evaluation were conducted. Finally, the data set used for testing and evaluation is described.

4.1 Design Research

The most appropriate way to approach the identified problem, and answer the research questions is by conducting a design study. Design science is a part of the information systems research cycle that creates and evaluates IT artifacts intended to solve identified organizational problems (Hevner, et al., 2004). The key difference between a design research study and routine design of applications and systems is the identification of a contribution to the archival knowledge base of foundations and methodologies (Hevner, et al., 2004), in this case the application of recommender systems.

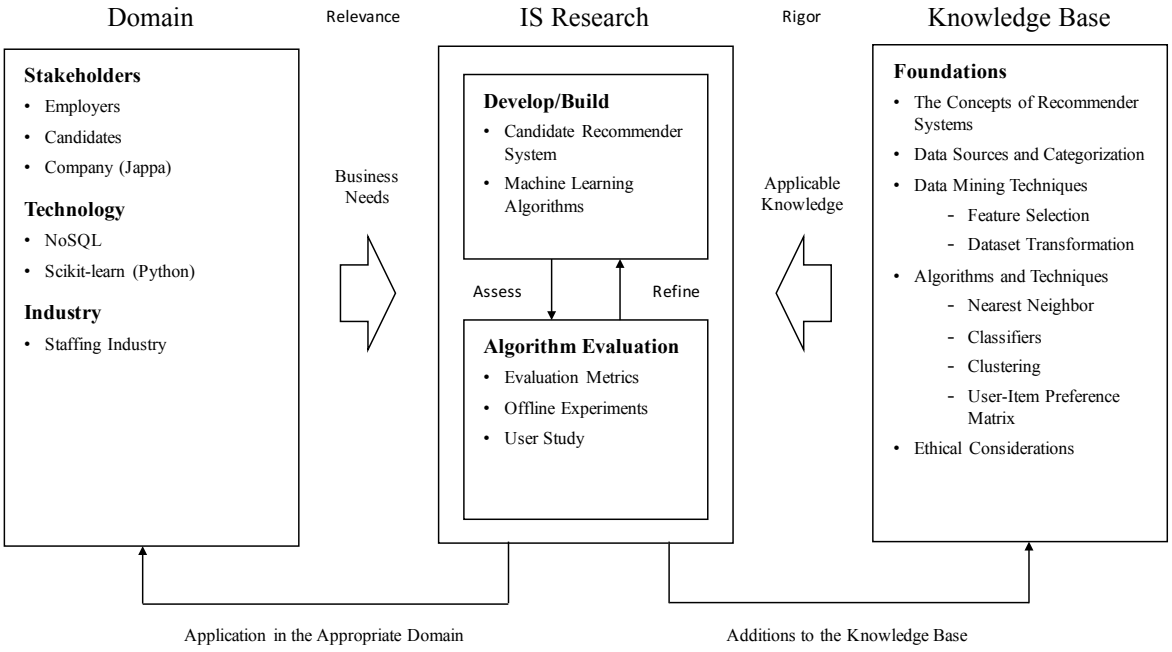

Figure 6: Overview of research approach based on a framework for design research studies by Hevner et. al (2004, p.80).

Hevner et al. (2004) suggest a framework for conducting design science research. An adoption of this framework is shown in Figure 6. By combining existing knowledge from relevant research with the identified business need, a prototype can be developed and evaluated. The results can then be applied in the organization and contribute to the existing knowledge base. By performing a literature review of relevant research and best practices in similar domains, the existing knowledge base will serve as basis for how a prototype can

be developed. This is an important part of the study, as rigor in design research is achieved by applying existing foundations and methodologies (Hevner, et al., 2004).

The business need behind the research topic is investigated at a Swedish start-up company called Jappa. As a company in the staffing industry who provides an online recruiting platform, Jappa has identified potential benefits of implementing a candidate recommender system.

The review of the existing knowledge base and environmental factors indicated that the remaining part of the study could be divided into three parts: *feature selection*, *algorithm selection,* and *model evaluation*. In order to quickly identify risks and difficult parts of developing a prototype (Ries, 2011), a *minimum viable product* (MVP), was created early in the project. The MVP was then incrementally extended until a final prototype was ready for final evaluation.

4.2 Feature Selection

In this study, the feature selection was carried out in three phases in order to exhaustively collect all features that could be used to define the entities used in the recommender system. The first phase was an inspection of the company's service that was carried out to understand what information the employers and candidates passed on to the system and what interactions with the system that could be recorded as a basis for recommendation. The second phase was an inspection of the database where all currently existing data was stored. The third phase was a workshop designed as a brainstorming session including eight employees of the company. The participants in the workshop were three developers, two members of the sales organization, a project manager, a business developer and the vice president of the company. The intention was to further extend the list of possible features, not limited to data that Jappa were collecting at the time, and without consideration taken to how difficult the data would be to collect.

When an exhaustive list of possible features had been compiled, all features were labeled using the following coding:

1. the entity the feature was associated with;
2. whether the feature was an attribute, preference or a transaction;
3. whether data about the feature was collected today;
4. whether data about the feature could be collected automatically or must be entered manually into the system;
5. and whether the feature was categorical, numerical, boolean, or textual type.

A complete list of the collected features can be found in Appendix A (candidate features) and Appendix B (employer features). A limited set of features was selected for the

development of an MVP. Initially, three features were selected to represent each entity, with consideration taken to the availability of data and ease of implementation. Features with high availability of data were features that were already collected and stored in a database, and that had a high volume of entries. Features that were considered easy to implement were features that did not require a high degree of preprocessing. After the MVP had been implemented, additional features were added incrementally, once again with consideration taken to the availability of data, ease of implementation and relevance to recommendations.

4.3 Algorithm Selection

Since the purpose of this study was to evaluate different implementations of recommender systems, as a way of increasing the knowledge of applying such techniques in the domain, a set of interesting algorithms and techniques had to be selected for implementation and evaluation. Algorithm selection requires an understanding of the factors that affect the performance of an algorithm on a specific problem in order to make the decisions that the algorithm selection problem requires (Kotthoff, et al., 2012).

The selection was conducted by identifying and selecting a handful set of different algorithms though as having a reasonable application potential within the scope of the study. It was conducted by identifying the most common algorithms and techniques used in recommender systems from a general perspective across different domains. The systematic review studies by Portugal et al. (2015) and Jannach et al. (2012), which investigate existing trends and application areas of recommender algorithms, provided directions for defining the scope of interesting algorithms to be studied.

The following algorithms and techniques were chosen: a naïve Bayes classifier, a decision tree classifier, a k-means clustering algorithm, and a nearest neighbor-based algorithm. The algorithms were applied in different content-based, collaborative filtering, or hybrid approaches when building the recommender systems. Through the set of selected algorithms and techniques, the following recommender systems were implemented:

1. *System 1*. A hybrid recommender system combining CB and CF approaches using both item- and user-based techniques. A clustering algorithm is used to categorize similar items into clusters and a classifier is then used to predict the most appropriate cluster for the active user, based on preferences of similar users.
2. *System 2*. A hybrid and pure item-based recommender system. An item, previously preferred by the user, is chosen and a neighborhood algorithm is then applied to find similar items that can be recommended to the user.

3. *System 3*. A pure CF and user-based recommender system using a user-item preference matrix. A cosine similarity function is used for comparing the similarity of users.

4.4 Model Evaluation

In this study, two types of experiments were performed to evaluate the performance of the different recommender systems, offline experiments and a user study.

An offline experiment is a relatively inexpensive way of evaluating recommender systems, typically evaluating systems based on historical data of recorded user interactions with the system (Gunawardana & Shani, 2009). The drawback of offline experiments is that they cannot directly measure the perceived quality of evaluations from a user perspective. Gunawardana & Shani (2015) state that offline experiments typically answer a very limited set of questions and that the goal of such experiments, therefore, should be to filter out inappropriate approaches, leaving a smaller set of algorithms to be tested by user studies.

User studies are conducted by recruiting test subjects and letting them interact with the recommender system while recording behavior or asking qualitative questions regarding the system's performance (Gunawardana & Shani, 2015). User studies are expensive to conduct, but in return, assumptions about user preferences is not necessary, as it is in offline experiments (Gunawardana & Shani, 2015).

4.4.1 Offline Experiments

To evaluate different algorithms using offline experiments, the established method of dividing the data set into a training set $T_{training}$ and a test set T_{test} was applied when evaluating the recommender systems, as proposed by several studies (Hahsler, 2011; Sarwar, et al., 2001; Herlocker, et al., 2004). The training/test data set ratio during the evaluation experiments was chosen to 2/1. Given that training/test data set proportion, training and test data sets were randomly selected, and the algorithms were executed to produce a list of top-N recommendation items based on the training set from which the model was built on.

The evaluation procedure was executed in 10 replications for every algorithm and configuration (algorithm parameter settings), due to the randomness associated with each evaluation. The results from the 10 replications were aggregated to compute the mean score of the applied offline metrics: Accuracy, Coverage and Diversity.

As described in section 3.6.1, in order to apply metrics for measuring the accuracy, a confusion matrix must be created for each test user that receives recommendations from the system. A recommendation was categorized as a *good recommendation* (preferred) if the

user previously had expressed some kind of preference for the candidate. The criteria for if a user had expressed an explicit preference for an item were if a user had: 1) *bookmarked a candidate*; 2) *booked a meeting with a candidate*; 3) *sent a job offer to a candidate*; or 4) *hired a candidate*. For every test user, the components of the confusion matrix shown in Table 3 were calculated.

Table 3: Confusion matrix used in offline experiments.

	Recommended	**Not Recommended**
Preferred	True Positive (tp)	False Negative (fn)
Not Preferred	False Positive (fp)	True Negative (tn)

For each recommender system and test user, a list of *N* recommendations was produced with *N* varying between 1 and 10. The list of recommendations C_r was then compared to a validation set, consisting of a set of candidates C_p that the test user previously had expressed preferences for, and which were not used for building the models. The quadrants in the confusion matrix can be calculated by the following equations:

$$True\ Positives = C_r \cap C_p \quad \textbf{Eq. 21}$$

$$False\ Positives = C_r - C_p \quad \textbf{Eq. 22}$$

$$False\ Negatives = C_p - C_r \quad \textbf{Eq. 23}$$

$$True\ Negatives = All\ recommendable\ items - (tp + fp + tn) \quad \textbf{Eq. 24}$$

From the resulting confusion matrix, the precision, inverse precision, recall, inverse recall and the Matthews correlation coefficient was calculated for each user. Finally, for each system, the average of each metric was calculated at every value of *N*.

The item space coverage and user space coverage for each system were calculated using the formulas presented in section 3.6.2.

The diversity was calculated, for each value of *N*, using the normalized Euclidean distance between all pairs of recommended candidates. Each feature of the items was transformed to a value in the range [0,1] so that each feature would contribute equally to the Euclidean distance between each pair. The diversity in a list containing a single recommendation (*N* = 1) will, therefore, be 0.

4.4.2 User Study

The user study involved four users who were all familiar with Jappa's platform. A unique questionnaire was then created for every user. The questionnaire contained links to a number of candidate profiles on Jappa's platform and the user was prompted to rate the relevance of each candidate based on the content of that profile. The rating was an ordinal scale from 1 to 4, 1 indicating that the candidate was "not at all relevant" and 4 indicating that the candidate was "very relevant".

Each system recommended four candidates, with whom the company had had no earlier interactions, and for each system, another four candidates that the system had not recommended were randomly chosen. Therefore, in total 24 candidates were presented to each user. If a user did not meet the requirements necessary to receive four recommendations from a system, that user received a shorter list of recommendations. Based on the ratings provided by each user, a confusion matrix was created, as shown in Table 4, and each candidate in the questionnaire was categorized into one of the four quadrants.

Table 4: Confusion matrix used in user studies.

	Recommended	**Not Recommended**
Rated 3-4	True Positive (tp)	False Negative (fn)
Rated 1-2	False Positive (fp)	True Negative (tn)

From the resulting confusion matrix, the precision, inverse precision, recall, inverse recall and the Matthews correlation coefficient was calculated for each user, as in the case of the offline experiments.

4.5 The Data Set

The experimental data used for building the models and performing the evaluations were solely based on data from a database snapshot of Jappa's online platform for candidate recruitment (www.jappa.jobs). The data was filtered, transformed, and transferred to an ad-hoc Elasticsearch database, containing only the required data for performing this study. The ad-hoc database consisted of 11 249 employee and 226 employer data profiles at the time of the evaluations. The selected data set was the result of the conducted feature selection phase as described in section 4.2. In Table 4, the elements of candidates' and employers' data profiles that were used are described.

Table 5:Overview of Candidates' and employers' data profiles.

Profile	Data
Candidate	Industry affiliation, Age, Education, Work Availability (day, night, evening, weekend), Rating, Driver License, Truck license, Years of experience
Employer	Search behavior (city, region, industry), Location (city), Industry affiliation, Salary history, Bookmarking, Job offerings, Previous hiring

5 Implementation

All algorithms in this study were implemented using Scikit-learn, a collection of machine learning libraries for Python. Scikit-learn is built on NumPy, SciPy and matplotlib, popular Python libraries for scientific and mathematical computing and visualization. Scikit-learn contains tools for classification, regression, clustering, data pre-processing, dimensionality reduction and model evaluation.

Three recommender systems were implemented and each type had several variations. The following sections explain each system and the difference in implementation that existed within each type. For a more detailed explanation of how the algorithms were configured, see Appendix C.

5.1 System 1

The first system uses a collaborative filtering approach in combination with a content-based approach to produce recommendations. It also combines user-based and item-based techniques. The system starts by clustering all available items into k number of clusters, based on similarity of the item features shown in Table 6. Thereafter, a bridge is used to map each user, in a list of users, to clusters. An example of a bridge can be that a user has hired a candidate that belongs to a certain cluster, identified by a cluster number in the range $[1 - k]$. A classification algorithm is then used to predict the most appropriate cluster for the active user. From the predicted cluster, N number of recommendations is chosen based on an attribute that is introduced as *profileStrength*. Profile strength is a score associated with each user based on the completeness of their profile on the platform. An overview of how System 1 works can be seen in Figure 7.

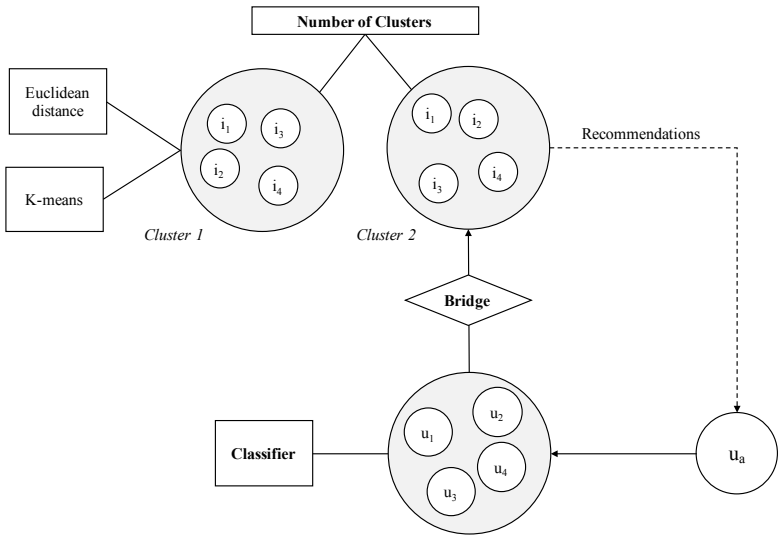

Figure 7: Overview of System 1.

In the system, both users and items are represented in vector forms, where each vector represents a set of features associated with either a user or an item. The features used to describe a user and an item can be found in Table 6 and Table 7 respectively. Both the user vectors and the item vectors consist of a combination of attribute features and transactional features. In this system, the transactional information has been aggregated into features that represent a behavior. For instance, locational search history of an employer has been aggregated to *mostSearchedCity*. This differs from the approach used in System 3 where, for a type of action, all transactions are taken into account when modeling a behavior.

The vectors are two-dimensional as some features can contain multiple values. For instance, an item can belong to more than one industry and can have more than one type of driving license while a user can have a mean wage for more than one industry. However, before the vectors can be used to train a classifier or clustering algorithm they must be transformed into one-dimensional vectors containing only numerical values. In this implementation, the bag-of-word approach is used for transformation as described in section 3.4.

Table 6: Item features.

Feature Name	Feature Type	Data Type
Industries	Attribute	Categorical
Age	Attribute	Numerical
upperSecondaryEducation	Attribute	Boolean
universityEducation	Attribute	Boolean
worksDay	Attribute	Boolean
worksNight	Attribute	Boolean
worksEvening	Attribute	Boolean
totalAvailability	Attribute	Numerical
averageRating	Transaction	Numerical
drivingLicenses	Attribute	Categorical
truckLicenses	Attribute	Categorical
yearsOfExperience	Attribute	Numerical

The item vectors are used for partitioning all items into k number of clusters. The k-means clustering algorithm takes a set of all items in vector form and creates an optimal partitioning as described in section 3.5.5. When evaluating the system, three different values of k were tested: 60, 100 and 140.

As described above, a list of users is selected that have some connection to a candidate in a cluster via a bridge. Just as the number of clusters varied, the bridge used for this association also varied. Bridges used in the experiments were bookmarks, booked meetings, sent job offers and hires of candidates. Variants of the system were evaluated by implementing one of these bridges individually, or by combining them into a single bridge. A user could appear

more than one time in the list if he or she had expressed preferences for several candidates. An example of how such a mapping, via a bridge, could look is shown in Figure 8.

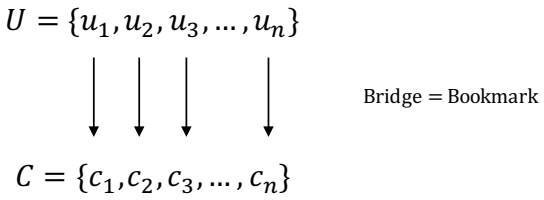

Figure 8: User-Cluster mapping via a bookmark bridge.

The mapping from a user to a cluster is then used to train a classifier so that the classifier will be able to make predictions regarding what cluster a certain user will prefer. Three different classifiers were used in the experiments: k-nearest neighbor, decision tree, and naïve Bayesian. Each user was represented by a vector u_i containing the features described in Table 7, and each cluster was represented by the cluster number c_j assigned by the k-means algorithm.

Table 7: User features.

Feature Name	Feature Type	Data Type
mostSearchedIndustry	Transaction	Categorical
mostSearchedCity	Transaction	Categorical
city	Attribute	Categorical
mostSearchedIndustry	Transaction	Categorical
meanWagePerIndustry	Transaction	Numerical
industries	Attribute	Categorical
poolsCreated	Transaction	Categorical

In conclusion, when the system makes a recommendation for an active user u_a, the classification algorithm first chooses an appropriate cluster based on the preferences of other similar users. Then *N* number of recommendations are chosen from the cluster given by the classification algorithm. In the offline evaluation, 45 variants of this system were tested, testing every combination of number of clusters (60, 100, 140), bridges (bookmark, meeting, job offer, job), and classification algorithms (k-nearest neighbor, decision dree, naïve Bayesian).

5.2 System 2

The second system is also a hybrid system; it uses a collaborative filtering approach in combination with a content-based approach to make recommendations. It uses a purely item based technique, not taking similarity between users into consideration. From an active user u_a, a bridge is used to select an active item i_a, and then *N* number of items that are

considered similar to the active item is recommended to the active user. An overview of how the system works can be displayed in Figure 9.

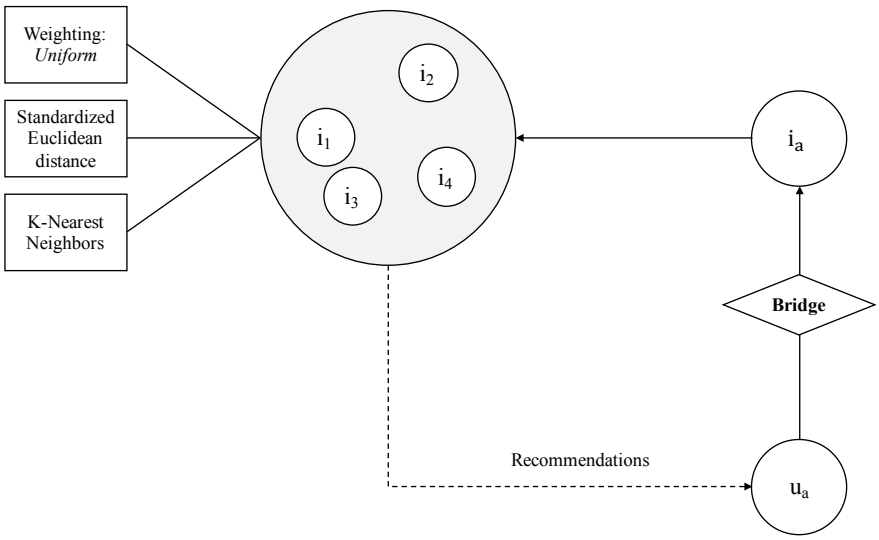

Figure 9: Overview of System 2.

The idea behind this algorithm is to select an active item that the active user has expressed a preference for in the past. In the experiments, five different bridges were used to select the active item:

1. a random candidate that the user had bookmarked;
2. the candidate that the user had booked the most meetings with;
3. the candidate that the user had sent most job offers to;
4. the candidate that the user had hired most times;
1. and a combination of all bridges where the strongest available indication of preference was chosen for each user. A hire was considered the strongest preference, then job offer, then booked meeting, and finally a bookmark.

When an item has been selected, the k-nearest neighbor algorithm is used to create a neighborhood of items based on item similarity. The neighborhood is constructed using the brute force version of the k-NN algorithm, which means that similarity is calculated for every pair of items. The features that are used to describe an item can be seen in Table 8.

Table 8: Item features.

Feature Name	Feature Type	Data Type
Industries	Attribute	Categorical
Age	Attribute	Numerical
upperSecondaryEducation	Attribute	Boolean
universityEducation	Attribute	Boolean
worksDay	Attribute	Boolean
worksNight	Attribute	Boolean
worksEvening	Attribute	Boolean
totalAvailability	Attribute	Numerical
averageRating	Transaction	Numerical
drivingLicenses	Attribute	Categorical
truckLicenses	Attribute	Categorical
yearsOfExperience	Attribute	Numerical

As in the case of System 1, each item was represented in the form of a vector that was transformed into a one-dimensional numerical vector using the bag-of-words approach described in section 3.4. When calculating distances between vectors, standardized Euclidean distance was used as the distance metric. When producing the recommendations, the *k* nearest neighbors of the active item were selected and recommended to the active user.

5.3 System 3

The third system is a "pure" collaborative filtering system based on the principle of using a user-item preference matrix as described in section 3.5.6. The chosen implementation was a user-based version of the system. The possibility of successfully implementing an item-based version of this CF system was discarded due to data sparsity problem in this case, since that approach would have required that every item (candidate) would have been preferred by many companies, which was not the case in this study. In other words, it is more likely that a company has preferences towards many items than many companies having preferences towards a single item, why the user-based approach was more preferable.

As described in section 3.5.6, first a user-item preference matrix is built in the system. The criteria for a user being part of the matrix was that it had at least one preference data point. A preference in the matrix is defined in terms of a preference score on an ordinal scale from 1-4. The value of each preference score corresponds to an explicitly expressed user preference per the criteria defined in section 4.4.1 and is showed in Table 9.

Table 9: Scores assigned for different types of preferences.

Preference ID	Preference Criterion	Preference Score
1	Has bookmarked employee	1.0
2	Has sent meeting request to employee	2.0
3	Has sent job offer to employee	3.0
4	Has hired employee	4.0

A simplified version of the implemented user-item preference matrix is illustrated in Figure 10. Note that the preference scores in Figure 10 do not represent the real data values in the system, but is only used for demonstration purposes.

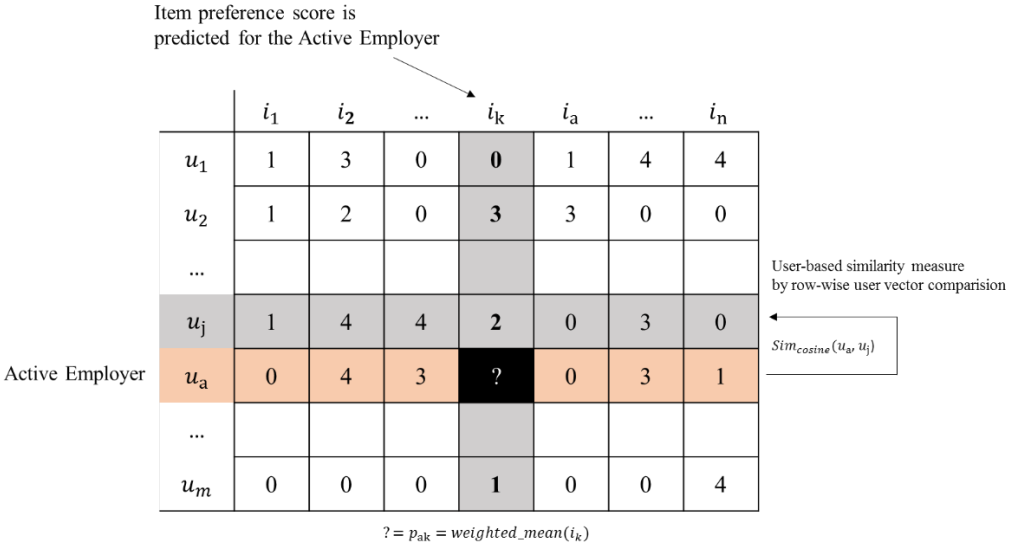

Figure 10: User-item preference matrix for the implemented system.

The cosine similarity function that computes weighted similarity is applied for performing the row-wise comparison of the user vectors, using the brute force method (computing similarity with all other users in the matrix). The parameters to be determined include the *k*-value, the number of most similar users to make predictions from, and the *N*-value, defining the length of the recommendation list to a user.

6 Results

This chapter presents the results from the conducted evaluations of the implemented systems and their variants. First, the results from the offline experiments are presented for each system and its variants, followed by the results of user study. Key results from the two evaluation methods are then emphasized in a separate section, comparing the theoretical results of the offline metrics with the empirical results of the user tests. Finally, the research questions of the study are addressed in the light of the presented results.

6.1 Offline Tests

In this section, the results from the offline experiments evaluating System 1, 2, and 3 are presented in separate sections for each system. It is then followed by a summarizing section comparing the results from all the system evaluations. Within each system, different variants are evaluated and compared to each other.

All systems were evaluated using the common offline metric framework presented in section 4.4.1. However, depending on the type of system, specific validation procedures had to be performed to avoid biases in the evaluation. The systems' specific evaluation procedures are described in the beginning of each system's evaluation section. As a recap, three properties were evaluated for each system: accuracy, coverage, and diversity. The metrics used for measuring accuracy were precision, inverse precision, recall, inverse recall and the MCC. Item space coverage and user space coverage were used for measuring coverage, and diversity was measured as the standardized Euclidean distance between the recommended items' in vector form.

6.1.1 System 1

To avoid biases in the classification and prediction of users' cluster preferences, a training set of users $T_{Training} = \{u_1, u_2, \dots, u_n\}$ with the corresponding training set of clusters $C_{Training} = \{c_1, c, \dots, c_n\}$ were used to train model. The system was then evaluated using users from a test set $T_{Test} = \{u_1, u_2, \dots, u_m\}$. 230 users were included in the test and training set.

In total, 45 variants of System 1 were evaluated. The number of variants was given by the combination of: the number of clusters, the classification model, and the bridge that were used.

$$\#variants = \begin{bmatrix} 140 \\ 100 \\ 60 \end{bmatrix} \times \begin{bmatrix} KNN \\ Decision\ Tree \\ Naïve\ Bayes \end{bmatrix} \times \begin{bmatrix} bookmark \\ meeting \\ job\ offer \\ hire \\ * \end{bmatrix} = 45 \qquad \textbf{Eq. 25}$$

The configurations for the five best performers, according to the MCC are displayed in Table 10. Out of the five best performers, all variants divided the items into 100 clusters, while both the bridge and classifier varied. The k-NN classifier, however, was the most frequently occurring classification model amongst the best performing variants.

Table 10: Configurations for five best performers in the Matthews correlation metric.

Name	Number of clusters	Classifier	Bridge
Variant 1.1	100	Decision Tree	Meeting
Variant 1.2	100	K-Nearest Neighbor	Job
Variant 1.3	100	Naïve Bayes	Meeting
Variant 1.4	100	K-Nearest Neighbor	*
Variant 1.5	100	K-Nearest Neighbor	Job Offer

Figure 11 displays the Matthew correlation coefficient for every value of N for the variants 1.1 – 1.5. The results show that there is very little variation, both between the variants and between the different numbers of recommendations. All variants lie between 0 and 0,08 for all values of N. For different values of N, different variants of the system have the best accuracy. Variant 1.3 however, has the best accuracy for all $N > 4$.

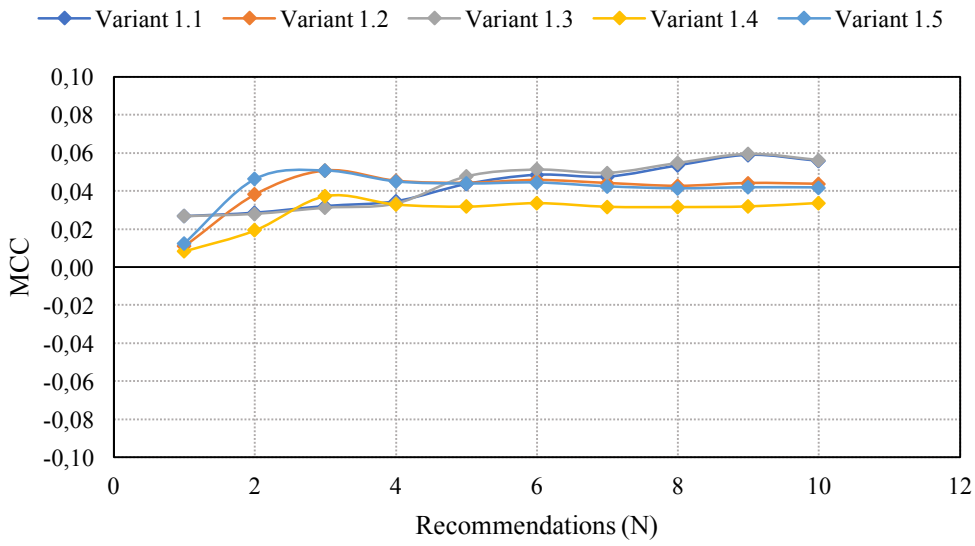

Figure 11: Matthews correlation coefficient for the five best performing variants.

Figure 12 shows the precision and inverse precision for variants 1.1 – 1.5, indicating how well the variants predict the preferences of a user.

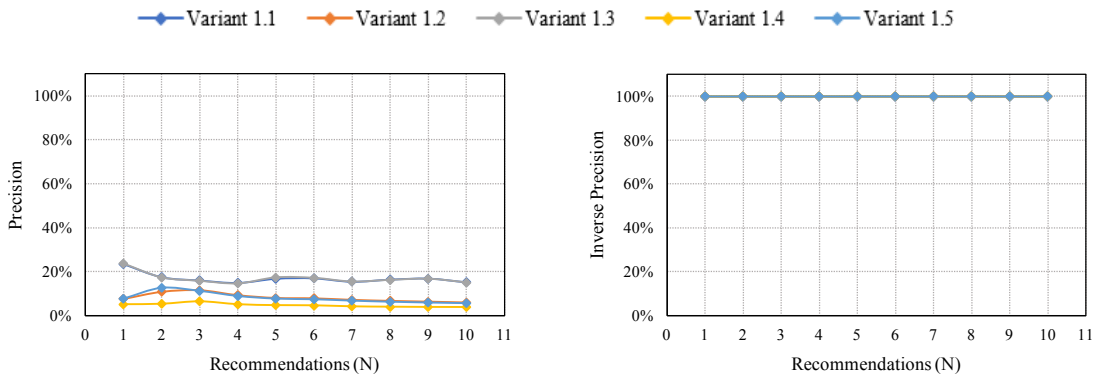

Figure 12: Precision and inverse precision for variants 1.1 – 1.5.

The precision varies amongst variants, with variant 1.3 having the highest score for all values of *N*. A trend can be spotted, indicating that the precision slightly declines for higher values of N, this is true for all variants. The inverse precision is close to 100% for all variants of the system, a result of many negative samples in relation to *N*. That is, only a small number of recommendations are being produced in relation to the total number of available items. A similar pattern can be seen in Figure 13, which shows the recall and inverse recall for the same variants. The large number of negative samples in relation to *N* causes a low recall and high inverse recall for all variants. When looking at the scores in tabular form, a trend of increased recall and decreased inverse recall with higher number of *N* is prominent. Furthermore, it becomes visible that 1.2, 1.4 and 1.5 outperform the others in recall while performing equally well in inverse recall.

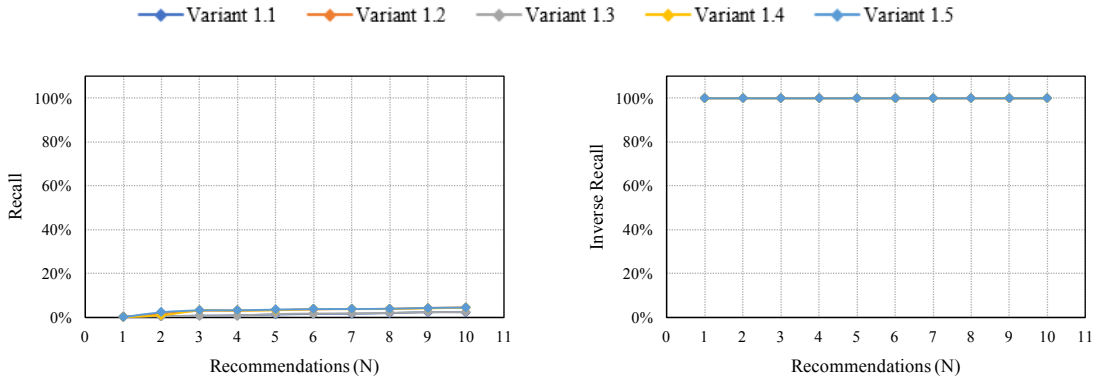

Figure 13: Recall and inverse recall for variants 1.1 – 1.5.

Table 11 shows the variants' user space coverage and item space coverage. Due to the usage of a classification algorithm and a clustering algorithm combined, both metrics of coverage

are 100%, independently of which configuration that is used. The result is not surprising as for every user, a cluster can be predicted based on similarity to other users and every item belongs to a cluster.

Table 11: User space coverage and item space coverage for variants 1.1 – 1.5.

Name	User Space Coverage	Item Space Coverage
Variant 1.1	100%	100%
Variant 1.2	100%	100%
Variant 1.3	100%	100%
Variant 1.4	100%	100%
Variant 1.5	100%	100%

Figure 14 shows the diversity in the produced recommendations for every variant. Two things can be noticed in the graph. Firstly, there is very little difference in diversity between the system variants and secondly, the diversity seems to be higher for smaller values of *N*.

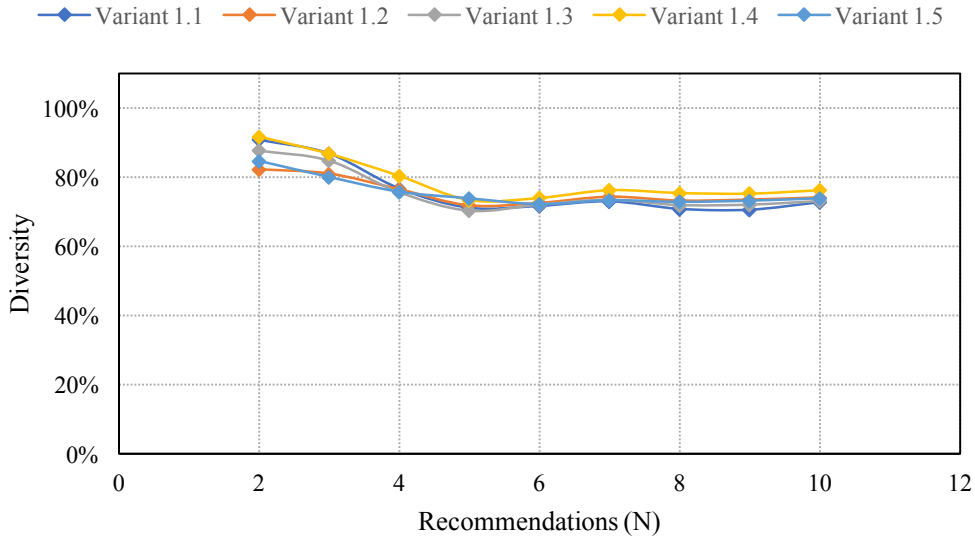

Figure 14: Diversity of recommendations for variants 1.1 – 1.5.

6.1.2 System 2

Five variants of System 2 were tested and evaluated. The five variants were evaluated for all 230 users in the system. The varying factor was how the active item was selected. As described in section 5.2, the active item is always selected based on the user's earlier preferences, however, the type of preferences taken into account varied. The configurations of the different system variants are shown in Table 12.

Table 12: Configurations for the five variants of System 2.

Name	Bridge
Variant 2.1	Bookmark
Variant 2.2	Meeting
Variant 2.3	Hire
Variant 2.4	Job Offer
Variant 2.5	*

The Matthews correlation coefficient for all five variants and are shown in Figure 15. As in the case of System 1, the MCC varies only slightly depending on which variant of the system that is tested or the number of recommendations that are produced. The metric is between 0 and 0,06 for all variants. The graph indicates that variants 2.3 and 2.4 performs better than other variants for values of $N > 4$.

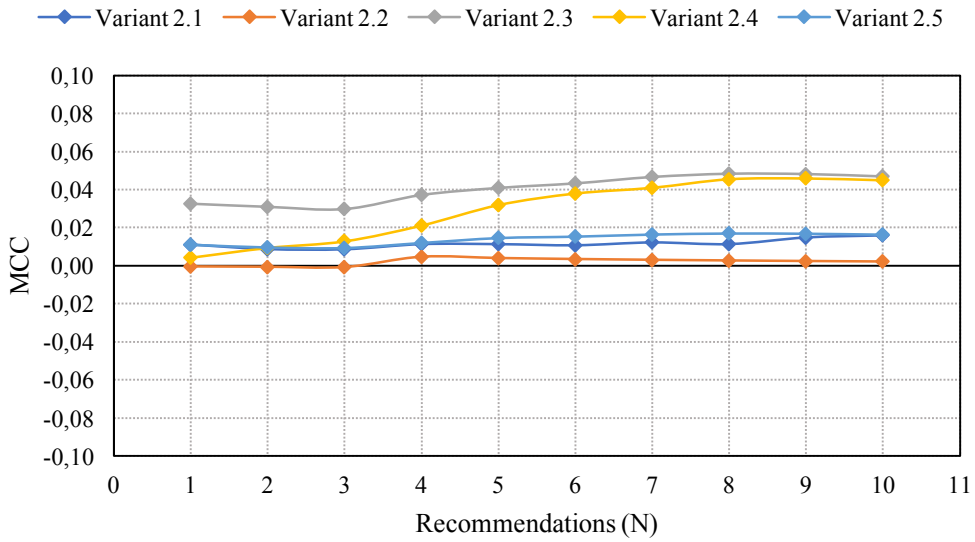

Figure 15: Matthews correlation coefficient for all five variants 2.1 – 2.5.

The precision and inverse precision curves, shown in Figure 16, show that variant 2.3 and variant 2.4 have the best performance in terms of precision while all variants perform well in inverse precision due to the large number of negative samples in relation to N.

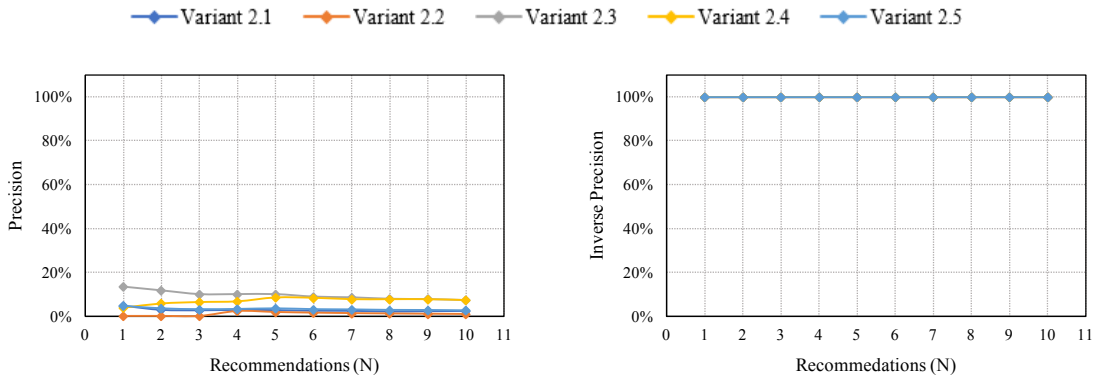

Figure 16: Precision and inverse precision for variant 2.1 – 2.5.

Recall and inverse recall, shown in Figure 17, are again heavily influenced by the high number of negative samples. However, when looking more closely at the results, variant 2.3 and variant 2.4 performs better in the recall metric, especially for values of $N > 3$, with approximately the same performance in the inverse recall. This partially explains why these two variants have a better MCC for higher values of N.

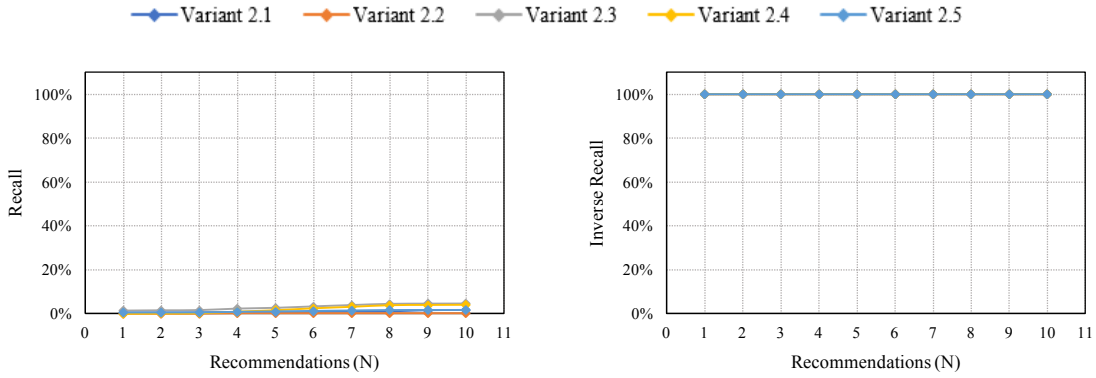

Figure 17: Recall and inverse recall for variant 2.1 – 2.5.

As can be seen in Table 13, the user space coverage varies greatly between the different variants.

Table 13: User space coverage and item space coverage for variants 2.1 – 2.5.

Name	User Space Coverage	Item Space Coverage
Variant 2.1	37,6%	100%
Variant 2.2	4,9%	100%
Variant 2.3	13,3%	100%
Variant 2.4	11,5%	100%
Variant 2.5	38,5%	100%

The component that influences the user space coverage, in this case, is the bridge used in the system. When a certain bridge is used, only users who have performed that specific action in the system can be provided recommendations for. For instance, if job offer is used as the bridge, a user is required to have sent at least one job offer to a candidate in order to receive recommendations of other similar candidates. Therefore, variant 2.5 has the greatest user space coverage as it uses all recorded actions as bridge. Variant 2.1 also has high user space coverage, indicating that many of the users in the system have used the bookmark function. The item space coverage, however, does not vary between the variants since all variants are able to recommend all candidates in the database.

As can be seen in Figure 18, the diversity of recommendations does not vary for the different variants of System 2. Furthermore, it stays constant for all values of *N*.

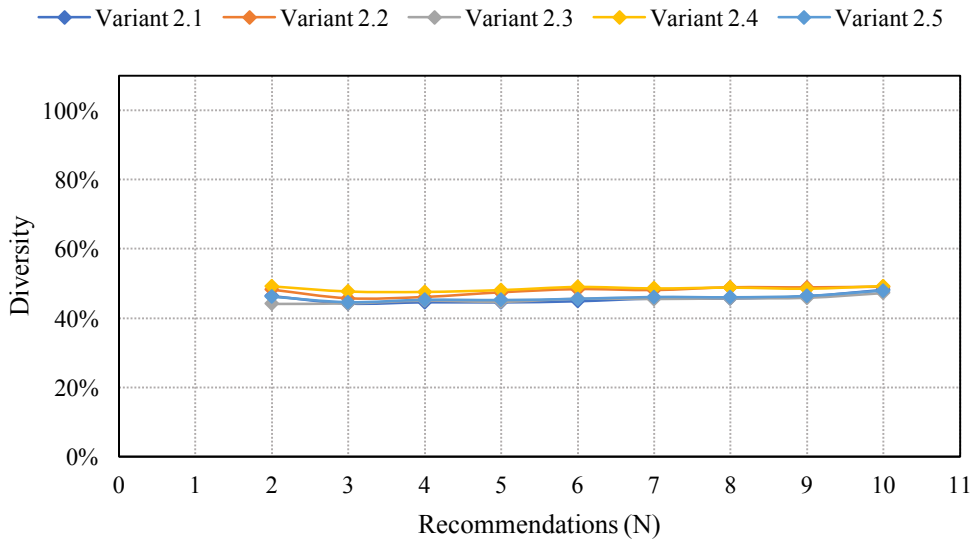

Figure 18: Diversity of recommendations for variants 2.1 – 2.5.

6.1.3 System 3

The evaluation of System 3, the CF technique based on a user-item preference matrix, followed the guidelines for evaluating such systems as described by Herlocker et al. (2004) and Hahsler et al. (2011). The data set was also divided into training and test sets like in the case of System 1, but a different approach was taken to validate the recommended items. Since preferences scores of a test user vector are used to train the CF algorithm to produce recommendations, a set of preference scores was hidden for every active test user during the evaluation. Breese et al. (1998) introduced a method for this called the *Given x* protocol. For the Given x protocol, *x* values (preference scores) are randomly selected in the test user vector and the remaining preference scores in the vector are hidden for the CF algorithm. In this study, a similar approach was applied, but we applied the "*hidden x protocol*" instead,

meaning that x number of preference scores were hidden in the active test user vector. Three preference scores were hidden for every active test user, implying the minimum amount of required preference scores among the test users to be 4, and thus giving a user-item preference matrix consisting of 24 users and 397 items. This discrimination criterion was considered reasonable since it implied an average number of preferences per user of 23, 33 and a median value of 8. If the recommender system succeeded to recommend a hidden preference, then it counted as a TP. Having the confusion matrix in mind for this scenarios during the evaluation, the row of preferred items constituted the set of all hidden preference scores of an active test user.

System 3 was evaluated by varying the k-NN parameter. K-values of 1-3 were tested, as shown in Table 14.

Table 14: Configurations for the three variants of System 3.

Name	k
Variant 3.1	1
Variant 3.2	2
Variant 3.3	3

Testing System 3 for ten values of N, the MCC curves were computed as shown in Figure 19.

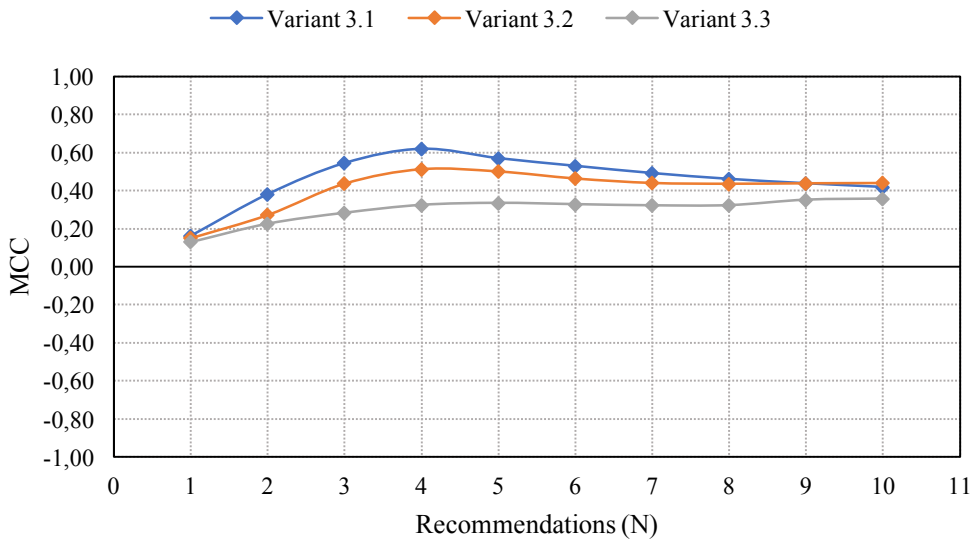

Figure 19: Matthews correlation coefficient for variants 3.1 – 3.3.

The MCCs reveal that System 3 performs optimally when using a single nearest neighbor ($k = 1$), this is true for all values of N. It can also be seen that increasing k will make System 3 perform worse in terms of MCC. The curve reaches its optima when the size of the

recommendation list is set to $N = 4$ for the $k = 1$ case. Selecting $N = 1$, all the system configurations perform almost equally with an MCC value close to 0,15. An optimal value of N seems to increase as the k-value increases, reaching an optimal value for $k = 2$ when $N = 4$, and for $k = 3$ when $N = 10$ for respectively.

The precision and inverse precision, shown in Figure 20, indicates the same performance ranking among the different variants, giving a maximum precision of about 55% for the $k = 1$ variant when $N = 3$. This optimal N-value of precision slightly differs from the optimal N-value in the MCC curve. The inverse precision curves, follow an almost constant line close to 100% for all variants and every N. However, looking at the raw data, it shows that the inverse precision decreases as N increases. The characteristics of the inverse precision curves are due to the of number negative samples being significantly higher than the number of recommended items for all values of N.

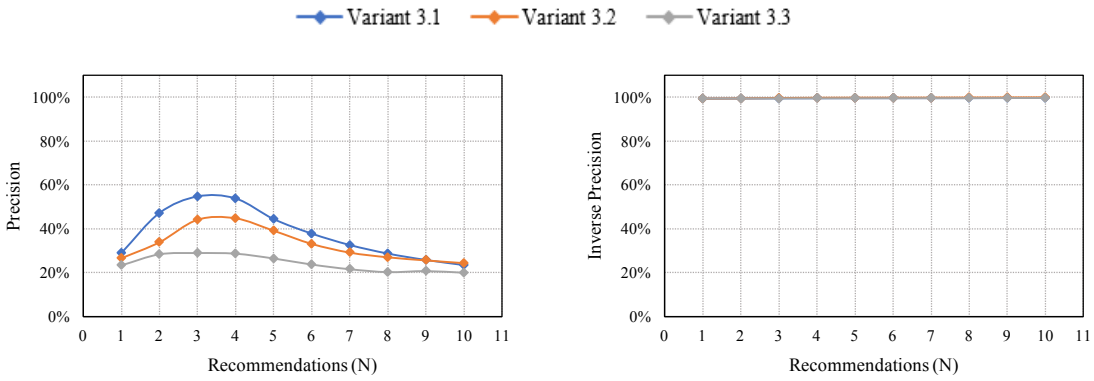

Figure 20: Precision and inverse precision for variants 3.1 – 3.3.

The recall curves in Figure 21: Recall and inverse recall for variants 3.1 – 3.3. show that recall values increase as N increases, and once again, higher k-values lead to a worse recall for all the variants. Also, in this case, the system performs well in this metric for $N = 4$. The curves derivative, in the $k = 1$ case, has a high positive value in the interval $0 < N < 4$, and then decreases for $N > 4$. From these curves, it can be that revealed that almost 80% of the hidden item preference scores (totally 4 hidden preferences) were recommended at most. At $N = 4$, more than 70% of the hidden items were recommended. For the inverse recall curves, their characteristics showed similar results as for the inverse precision curves, an almost straight line with minimal decrease as the N-value increases. This is also due to

the number negative samples being significantly higher than the number of recommended items for all values of N.

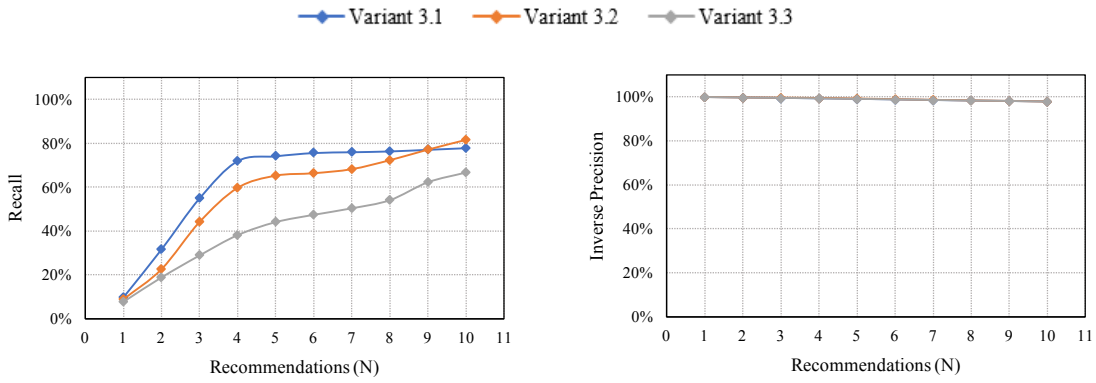

Figure 21: Recall and inverse recall for variants 3.1 – 3.3.

The user and item space coverage, shown in Table 15, explain that System 3 excludes many users in the system that do not fulfill the requirements for gaining a recommendation from this type of system. This threshold was set to a minimum of four unique preferences per user towards an item for it to be included in the user-item matrix. The low user space coverage therefore means that the system had many inactive users. In this case, the low user space coverage also had a negative impact on the item space coverage, since many items have been preferred by users that do not fulfill the requirements, and are therefore excluded from the system's set of recommendable items.

Table 15: User space coverage and item space coverage for variants 3.1 – 3.3.

Name	User Space Coverage	Item Space Coverage
Variant 3.1	38,5 %	3,5%
Variant 3.2	38,5 %	3,5%
Variant 3.3	38,5 %	3,5%

Figure 22 shows the diversity in recommendations for the different variants.

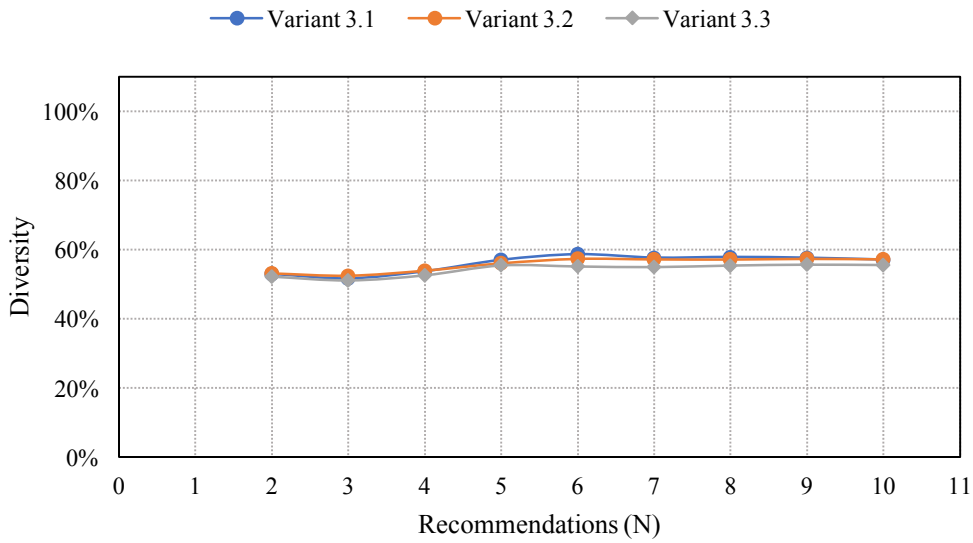

Figure 22: Diversity of recommendations for variants 3.1 – 3.3.

There is little to no variation in diversity between variants, and diversity seems to increase slightly for greater values of N.

6.1.4 Summary of Offline Test Results

In this section, one variant from each system was chosen and the three variants were compared to each other. Variants were chosen that performed well in terms of MCC for N = 4, the N-value used in the user study. Furthermore, the results were compared to two different random algorithms. The first random algorithm recommended candidates randomly from the complete item space, and the second recommended random candidates from the industry that was associated with the active company's profile. Variants of each system showed little variation in the terms of accuracy. When comparing between systems, however, there was a significant difference. As Figure 23 shows, variant 3.1 outperforms both other systems for every value of N. All algorithms have better accuracy, MCC, at all values of N than both random algorithms used for comparison.

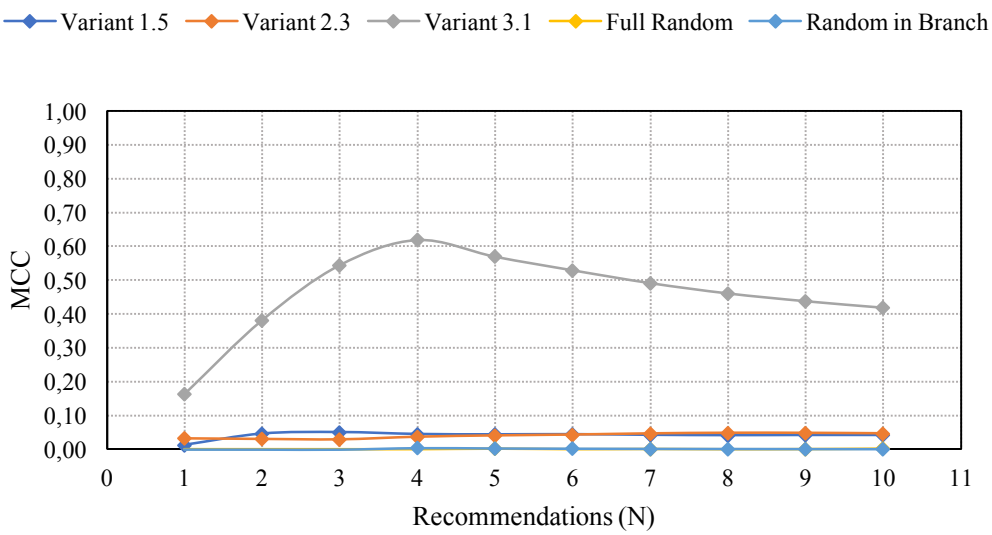

Figure 23: The Matthew correlation coefficient for the chosen variant of each system.

Variant 3.1 also has the highest precision for values of $N > 1$ while there is only a small difference in inverse precision performance. That is, variant 3.1 is better in predicting preferred items, while the systems perform similarly in terms of predicting not preferred items. There is a slight decrease in precision for all variants as N increases. Inverse precision performance is high in all variants, once again an expected result due to N being small in relation to the item space of each system. Even the random algorithms show strong performance in the reverse precision, however, they have worse precision than all system variants.

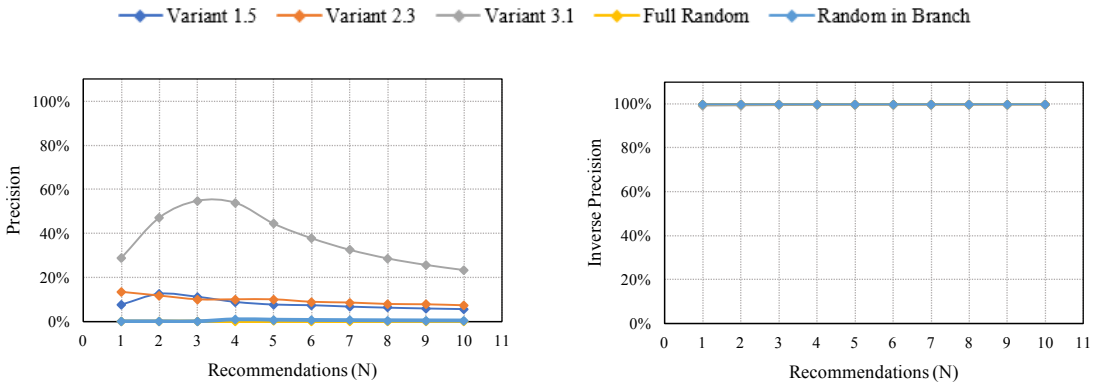

Figure 24: Precision and inverse precision for the chosen variant of each system.

Figure 25 shows the recall and inverse recall for the three systems. Variant 3.1 has significantly higher recall than other system variants for all values of N. Furthermore, it increases at a greater rate than variants from the other systems. Variant 1.3 and 2.3 performs

slightly better in the inverse recall. However, all systems have an inverse recall higher than 97%. As in the case of precision and inverse precision, the random algorithms perform well in the inversed metric due to a large number of negative samples, but all systems variants have higher recall scores than the random variants.

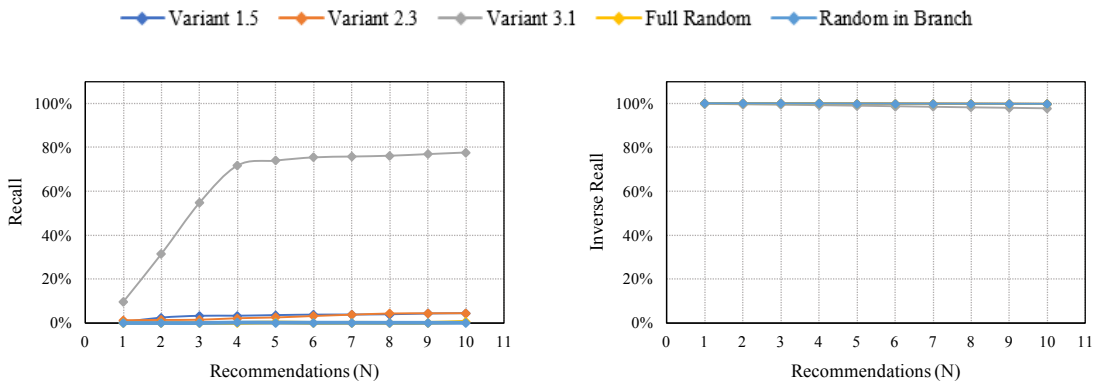

Figure 25: Recall and inverse recall for the chosen variant of each system.

As shown in Table 16 variant 1.2 has the best overall coverage with a user space coverage and item space coverage of 100%. Variant 2.3 is able to recommend all items but the available data is only sufficient to provide recommendations for 13,3% of the users. Variant 3.1 has a better user coverage than variant 2.3, 38,5% of the users can receive recommendations from that system. The item space coverage of variant 3.1, however, is very low at 3,5%, meaning that only a small number of the available items can be recommended to the users.

Variant 3.1 had the best MCC but had low coverage, especially item space coverage. The results indicate that there is a trade-off between the performance of the system and its item space coverage. These results, however, are not so surprising as variant 3.1, and System 3 in general, is more restrictive in terms of which users it includes, in combination with the low activity of users towards items (user preferences) in the system.

Table 16. User space coverage and item space coverage for the chosen variant of each system.

Name	User Space Coverage	Item Space Coverage
Variant 1.2	100%	100%
Variant 2.3	13,3%	100%
Variant 3.1	38,5%	3,5%
Full Random	100%	100%
Random in Industry	5,2%	62,2%

Figure 26 shows the diversity of recommendations produced by the variants. There is a larger difference between the systems than between variants of the same system. Variant

1.5 shows the greatest diversity of recommendations. However, the graph results indicate that the diversity for that variant declines for larger values of *N*. Variant 2.3 has the lowest diversity of recommendations and the diversity for that variant seems to be constant for different values of *N*. The diversity of variant 3.1 seems to increase slightly for larger values of *N*.

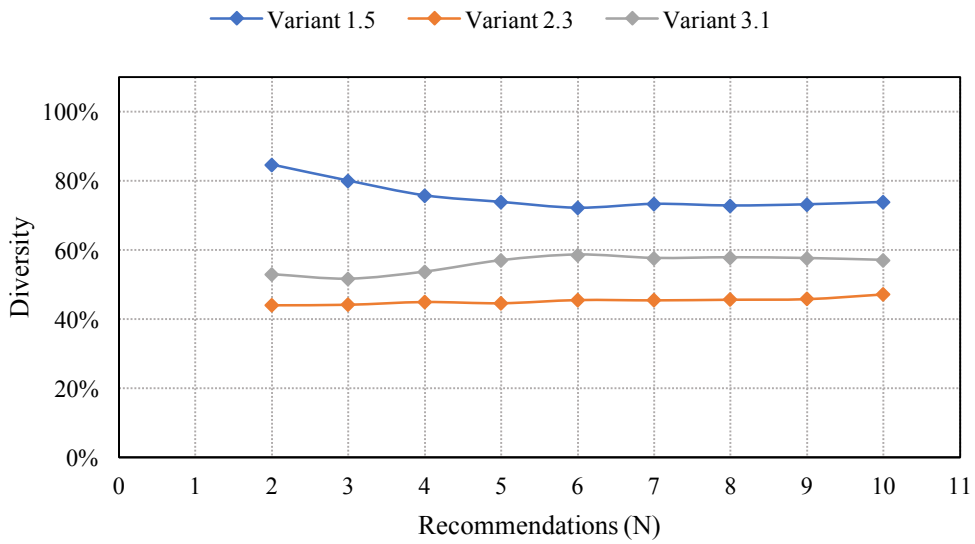

Figure 26: Diversity of recommendations for the chosen variant of each system.

6.2 User Study

The variants more closely examined in section 6.1.4 were also evaluated in a user study. In the user study, the variants were evaluated for $N = 4$. The results, see Table 17, show that variant 1.5 and variant 3.1 both had significantly higher MCC than variant 2.3.

Table 17: Results from the user study, $N = 4$.

	Variant 1.5	Variant 2.3	Variant 3.1
MCC	0,5774	0,1072	0,5296
Precision	50%	12,5%	54%
Inverse Precision	100%	93,8%	93,8%
Recall	100%	67%	87,5%
Inverse Recall	67%	52%	71%

Looking at the precision and inverse precision, all variants have a high inverse precision but 2.3 have a very low precision. Variants 1.5 and 3.1 have a precision of approximately 50% indicating that users perceive half of the recommendations as relevant to them. The inverse recall was significantly lower for all variants than in the offline experiments, possibly due to a smaller number of negative samples in relation to *N* in the user study.

6.3 Addressing the Research Questions

Based on the presented results obtained from the offline experiments and the user study, this section will provide answers to the research questions of this study.

RQ1. *What are the key design choices when building a candidate recommender system?*

The key design choices behind an implementation of a candidate recommender system are highly dependent on the system adapter's business goals, and the conditions in terms of data availability of users and items in the system. To achieve specific goals, different algorithms, techniques, and features can be selected and combined in different ways to build a candidate recommender system.

Applying classifiers and clustering in a hybrid and user-based approach (variant 1.5) was confirmed to be a viable way of producing recommendations in this context, as well as applying a pure collaborative filtering technique using a k-NN algorithm and a user-item preference matrix (variant 3.1). Variant 1.5 performed worse than variant 3.1 in terms of accuracy in the offline experiments, but turned out to produce a high degree of relevant recommendations in the user study. In contrast, variant 3.1 showed high accuracy scores, both during the offline and user evaluations. In this case, it was achieved at the expense of lower user and item space coverages, caused by a stricter discrimination criterion for users, which required a larger amount of recorded interactions with the system.

During this study, a new concept, referred to as a bridge, was introduced. A bridge connects a user to an item in the system, and the system's performance can differ a lot depending on the choice of bridge, which therefore makes it a key factor when designing a candidate recommender system. In conclusion, the candidate recommender system designer must choose appropriate algorithms and techniques, depending on the desired levels of accuracy, coverage, and diversity that reflects the recommender system's business goals.

RQ2. *What are the most relevant metrics and evaluation methods for assessing the performance of the recommender systems in the studied domain?*

As emphasized by the answers to **RQ1**, the choice of metrics when designing a candidate recommender system is very important, since they reflect different aspects of the system's performance and satisfy different business goals. To assess the performance of recommender systems, offline experiments is an effective way of choosing the right set of recommender systems to evaluate before release, when there are a variety of alternatives available. However, it should be followed up by a user study, since the offline evaluation might not always reflect the real accuracy of the systems. The accuracy was mainly assessed by the MCC score, that turned out be a trustworthy metric in most cases during offline evaluation when comparing to the user study results. Because, in the case of variant 1.5, the

MCC metric showed an underestimated result in terms of its accuracy to recommend relevant items to users in the offline experiments compared to the user study. If the results instead had been overestimated, the MCC metric's credibility could have been questioned.

Measuring the systems user and item coverages is also relevant during evaluation. The systems accuracy must be set in relation to its coverages, especially when comparing evaluations of different systems, to make a more comprehensive comparison. The metric used to measure diversity of recommendations is also an important aspect, since it introduces a new way to evaluate the systems in relation to their business goals. The results showed that the different systems were differentiable in terms of diversity and therefore introduced a new dimension to the complete assessment of the systems.

7 Discussion

The results from the evaluation of the three proposed systems, using offline experiments and a user study, revealed interesting insights regarding the systems' performance. The most prevalent trend, both in the offline experiments and in the user study, is the strong performance in terms of accuracy for System 3, a pure collaborative filtering algorithm. In previous research, pure collaborative filtering algorithms have been proven to be the most successful approach in several contexts (Al-Otaibi & Mourad, 2012), mainly by emphasizing the systems' accuracy (Herlocker, et al., 2004). By considering these studies, in could have been directly concluded that System 3 is the superior system in the context of this study. However, taking more properties of the recommender system into consideration highlights the complexity of recommender systems evaluation. System 1 has significantly higher coverage compared to System 3, both when measuring user space coverage and item space coverage. High item space coverage could potentially be a desirable property in the studied domain. Because, in contrast to a movie recommender system or an e-commerce recommender system, each item in a candidate recommender system has limited work availability and can, therefore, only be utilized by a single user, or few users, at any given time. Rather, recommending the same items to users in such a system, or recommending a candidate already employed by one user, might cause a lower perceived quality of the service as a whole. Furthermore, as the items in a candidate recommender system are real people who also might abandon the platform if it fails to create value for them, a recommender system that manages to engage a greater proportion of the available candidates could be desirable from a business point of view. However, even though System 1 has significantly higher coverage than System 3, it is important to note in this discussion that System 3 still has a high theoretical coverage but is highly dependent on the availability of data gained from users' interaction with the system. With more recorded interactions, the difference in coverage between the systems might have been less prominent.

From the results, a likely trade-off between a system's accuracy and coverage is observable, in accordance with the results previously revealed by Gunawardana & Shani (2015). A candidate recommender system designer must consider this trade-off, and determine an optimal point between maximum accuracy or maximum coverage, depending on the domain's business goals. In a recruitment context, a candidate recommender system will support a recruiter in finding relevant candidates effectively with respect to time, which makes the time factor a critical aspect when assessing the system's utility. Having this business goal in mind, high accuracy will prevent users (recruiters) from spending time in inspecting non-relevant candidate recommendations. High accuracy is therefore of high importance in such contexts. However, when optimizing the candidate recommender system to perform in terms of accuracy, it should not discriminate its platform users and items to the extent that it will harm the platform provider's business goals. A way of circumventing

the dilemma between, optimal accuracy or coverage, would be to implement parallel recommender systems that direct recommendations towards users based on their "data profile maturity levels". A low maturity level would mean that a specific recommender system is triggered in the platform for a user that has a limited amount of data in it profile. For the high-level maturity case, a threshold can be set, like in the case of System 3, that specifies the minimum data requirements for a user profile for it to become a target to the recommender system. Applying this approach, the system as a whole would imply a higher coverage of its users and items while keeping its ability to produce more accurate recommendations for users with high maturity levels.

Another noteworthy result is that the diversity of recommendations seems to vary only slightly between different configurations for each system, but a clear difference between the systems prominent. It is not surprising that System 2 seems to produce least diverse recommendations as the recommendations are produced based on item similarity. However, more unexpected is that System 1, which takes item similarity into considerations when creating clusters, seems to produce more diverse recommendations than System 3 which recommends items solely based on users' interactions with the items. This indicates that if diversity of recommendations is desirable from a business perspective, higher diversity can be achieved through a combination of a user-based classifiers and item-based clustering than with a purely item-based system or a pure collaborative filtering system. Comparing diversity to accuracy and coverage, it is harder to define what systems performed well in terms of diversity as good diversity is more subjective. Some companies might prefer more diverse recommendations while others less so. Therefore, further research is needed in order to establish what levels of diversity that are desirable in this context.

Comparing the results of the offline experiments and the user study, the systems showed better accuracy in the user test. A reasonable explanation is that the number of preferred candidates in the offline experiments were underestimated. When using historical data as criteria for validating a good recommendation, an underlying assumption is that all recommendations for which there are no earlier recorded expressions of preference for, are bad recommendations. The weakness in this assumption has been identified by Gunawardana & Shani (2015) and the results in this study seems to confirm that. However, with the knowledge in mind that the number of preferred candidates might be underestimated, the result indicate that offline evaluation does provide some insight. The ranking of the systems remained the same with System 3 having the best accuracy in performance both in the offline experiments and in the user study. Therefore, companies in the studied domain who wishes to improve and evaluate recommender systems can consider offline experiments a reliable option for conducting non-expensive comparisons between systems.

When optimizing recommender systems, attention also needs to be paid to the ethical aspects of the system. Awareness of the importance of personal integrity in this matter has caused governments and regulatory bodies to take legislative actions to make sure that data is collected and processed in a controlled fashion. In terms of recommender systems, such regulations could potentially affect the performance negatively due to less availability of data. However, in the studied domain candidates agree to submit data in order for employers to find them on the platform. In this case, the purpose for which non-sensitive data is processed in the recommender system is aligned with the motive for which the users agreed to submit the data, namely expose the candidate profile to employers. Therefore, no further caution needs to apply when using data in a recommender system compared to other processing of the data done in the system. Regarding sensitive data however, as described by Datainspektionen (1998), more strict regulations apply. Most of such sensitive data, such as political opinions or philosophical convictions, could be discarded anyways as irrelevant for a candidate recommender system, thereby not limiting the quality of recommendations. Sensitive data that could possibly be of interest in recommender systems in this context is trade union memberships and health information prohibiting the candidate from working in certain environments. Without access to data that could potentially filter out candidates, the accuracy could suffer and the users of the system might require a manual check-up to ensure that recommended candidates are able to work for them. However, the importance of personal integrity is high and regulations and guidelines regarding data security should be treated as preconditions that must be considered when designing recommender systems.

The threats to validity in this study are mainly limited to the sample size of data, in the offline experiments, and test subjects, in the user study. The limited availability of recorded interactions with the system caused low coverage, which in turn meant that the systems could only be evaluated for a limited number of users, especially in the case of System 3. Furthermore, as users in general had only explored a small number of the available items on the platform, the number of preferred items could have been underestimated in the offline experiments, leading to reduced precision. Due to time constraints only four test subjects were included in the user study and, therefore, those results should be seen as indications rather than proof, or ground truth, of system performance. Furthermore, the users were asked to rate 10 negative samples for each algorithm, which only represent a small portion of the actual negative samples. Therefore, the proportion of false negatives and true negatives runs a risk of not reflecting the real values. Also, just as in the case of offline experiments, the user study relies on an assumption, the assumption that the users' ratings of the candidates indicate the true quality of the recommendations. This assumes that users, with great certainty, know which candidates are relevant to them and which are not. To strengthen the assessment of the systems, further studies evaluating the systems in an online environment should be conducted.

This study has identified a number of guidelines for any software engineer who attempts to design and implement a recommender system. Firstly, it has highlighted the importance of combining different approaches and system variants in order to circumvent disadvantages of using any single approach. Secondly, it has shown that when evaluating the performance of different systems, applying offline experiments to measure the Matthews correlation coefficient can provide a good indication of performance in terms of accuracy. However, one should be aware that the metric can be underestimated, as users might have preferences for items that are not included by the criteria of preferred items in the experiments. Finally, by also taking coverage and diversity into account software engineers can get a more comprehensive understanding of the systems overall performance.

8 Conclusions

A recommender system is a viable option for dealing with information overload in several contexts, including identifying appropriate candidates for employers on a recruitment platform. Through implementation of three prototype recommender systems, this study has shown that by combining different approaches, such as content-based or collaborative filtering, when implementing candidate recommender systems, companies in the studied domain can overcome the limitations of using any single approach. Furthermore, broadening the scope of evaluation to include other properties than accuracy, such as coverage and diversity, increases the ability to build and select a recommender system that is better aligned with common business goals in the domain. A recommender system designer must, however, be aware of the possible trade-offs between these properties when optimizing the recommender system to perform with respect to certain business goals.

9 Bibliography

Adomavicius, G., Kamireddy, S. & Kwon, Y., 2007. *Towards more confident recommendations: Improving recommender systems using filtering approach based on rating variance.* s.l., Proc. of the 17th Workshop on Information Technology and Systems.

Adomavicius, G. & Tuzhilin, A., 2015. Context-Aware Recommender Systems. In: *Recommender Systems Handbook.* s.l.:Springer, pp. 191-226.

Al-Otaibi, T. S. & Mourad, Y., 2012. A Survey of Job Recommender Systems. *International Journal of Physical Sciences,* 7(29), pp. 5127-5142.

Amatriain, X. & Pujol, J., 2015. Data Mining Methods for Recommender Systems. In: *Recommender Systems Handbook.* s.l.:Springer US, pp. 227-262.

Ansari, A., Essegaier, S. & Kohli, R., 2000. Internet recommendation systems.. *Journal of Marketing,* 37(2000), pp. 363-375.

Attenberg, J. et al., 2009. *Collaborative email-spam filtering with the hashing trick.* s.l., s.n.

Attenberg, J. et al., 2009. *Feature hashing for large scale multitask learning.* s.l., ACM.

Billsus, D. & Pazzani, M., 1997. *Learning probabilistic user models.* [Online] Available at: http://www.dfki.de/~bauer/um-ws/

Breese, J., Heckerman, D. & Kadie, C., 1998. *Empirical analysis of predictive algorithms for collaborative filtering.* s.l., Proceedings of the Fourteenth on Uncertainty in Artificial Intelligence., pp. 43-52.

Buettner, R., 2014. *A Framework for Recommender Systems in Online Social Network Recruiting: An Interdisciplinary Call to Arms.* s.l., System Sciences (HICSS), 2014 47th Hawaii International Conference, pp. 1415-1424.

Burke, R., 2007. Hybrid Web Recommender Systems. In: *The Adoptive Web.* s.l.:Springer Berlin Heidelberg, pp. 377-408.

Burke, R., O'Mahony, M. P. & Hurley, N. J., 2011. Robust Collaborative Recommendation. In: *Recommender systems handbook.* s.l.:Springer US, pp. 805-835.

Datainspektionen, 1998. *The Personal Data Act.* [Online] Available at: http://www.datainspektionen.se/in-english/legislation/the-personal-data-act/
[Accessed 17 February 2017].

Davis, J. & Goadrich, M., 2006. *The relationship between Precision-Recall and ROC curves.*. s.l., ACM.

de Gemmis, M. et al., 2015. Semantics-Aware Content-Based Recommender Systems. In: *Recommender Systems Handbook.* s.l.:Springer US, pp. 119-159.

Delen, D. & Demirkan, H., 2013. Data, information and analytics as services. *Decision Support Systems,* Volume 55, pp. 359-363.

Eckerson, W. W., 2007. *PREDICTIVE ANALYTICS: Extending the Value of Your Data Warehousing Investment,* s.l.: The Data Warehouse Institute (TDW).

Felfernig, A., Friedrich, G., Jannach, D. & Zanker, M., 2015. Constraint-Based Recommender Systems. In: *Recommender Systems Handbook.* s.l.:Springer US, pp. 161-190.

Fisher, G., 2001. User modeling in human-computer interaction. *User Modeling and User-Adapted,* 11(1-2), pp. 65-86.

Ganchev, K. & Dredze, M., 2008. *Small statistical models by random feature mixing.* s.l., s.n., pp. 19-20.

Ge, M., Delgado-Battenfeld, C. & Jannach, D., 2010. *Beyond Accuracy: Evaluating Recommender Systems by Coverage and Serendipity.* s.l., ACM, pp. 257-260.

Gunawardana, A. & Shani, G., 2009. A Survey of Accuracy Evaluation Metrics of Recommendation Tasks. *Journal of Machine Learning Research,* pp. 2935-2962.

Gunawardana, A. & Shani, G., 2015. Evaluating Recommneder Systems. In: *Recommender Systems Handbook.* s.l.:Springer US, pp. 265-308.

Gupta, A. & Garg, D., 2014. *Applying data mining techniques in job recommender system for considering candidate job preferences.* s.l., Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on. IEEE.

Hahsler, M., 2011. *recommenderlab: A Framework for Developing and Testing Recommendation Algorithms,* s.l.: Southern Methodist University.

Har-Peled, S., Roth, D. & Zimak, D., 2002. Constraint Classification for Multiclass Classification and Rankning. *Urbana,* 51(2002), p. 61801.

Heap, B., Krzywicki, A., Wobcke, W. & Compton, P., 2014. *Combining Career Progression and Progression and Profile Matching in a Job Recommender System.* s.l.,

Pham DN., Park SB. (eds) PRICAI 2014: Trends in Artificial Intelligence. PRICAI 2014. Lecture Notes in Computer Science,.

Herlocker, j. L., Konstan, J. A., Terveen, L. G. & Riedl, J. T., 2004. Evaluating Collaborative Filtering Recommender Systems. *ACM Transactions on Information Systems,* 22(1), pp. 5-53.

Hevner, A. R., March, S. T., Park, J. & Ram, S., 2004. Design Science in Information Systems Research. *MIS Querterly,* March, 28(1), pp. 75-105.

Hong, W., Zheng, S. & Wang, H., 2013. A Job Recommender System Based on User Clustering. *Journal of Computers,* 8(8), pp. 1960-1967.

IBM Software, 2013. *Descriptive, predictive, prescriptive: Transforming asset and facilities,* Somers, NY: IBM Corporation.

ISACA, 2011. *Data Analytics - A Practical Approach,* Rolling Meadows, IL: ISACA.

Isinkaye, F. O., Folayami, Y. O. & Ojokoh, B. A., 2015. Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal,* 16(3), pp. 261-273.

Isson, P. J. & Hariott, S. J., 2015. *People Analytics in the Era of Big Data: Changing the Way You Attract, Acquire, Develop, and Retain Talent.* s.l.:Wiley.

Jannach, D., Zanker, M., Ge, M. & Gröning, M., 2012. *Recommender Systems in Computer Science and Information Systems - a Lanscape of Research.* Vienna, Austria, 13th International Conference on Electronic Commerce and Web Technologies, Springer.

Kim, K.-j. & Ahn, H., 2008. A recommender system using GA K-means clustering in an online shopping market. *Expert Systems with Applications,* Volume 34, p. 1200–1209.

Kotthoff, L., Gent, I. & Miquel, I., 2012. An evaluation of machine learning in algorithm slecetion for search problems. *AI Communications,* 25(2012), pp. 257-270.

Linas, B., Bernd, L., Peer, S. & Ricci, F., 2012. Context Relevance Assessment and Exploitation on Mobile Recommender Systems. *Personal and Ubiquitous Computing,* 16(5), pp. 507-5266.

Linden, G., Smith, B. & York, J., 2003. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing,* 7(1), pp. 76-80.

Lu, Y., El Helou, S. & Gillet, D., 2013. *A Recommender System for Job Seeking and Recruiting Website.* s.l., ACM, pp. 963-966.

Mitchell, T., 1997. *Machine Learning.* Burr Ridge, IL: McGraw Hill.

Mujawar, S. & Joshi, A., 2015. Data Analytics Types, Tools and their Comparison. *International Journal of Advanced Research in Computer and Communication Engineering,* pp. 488-491.

Nilashi, M., Haniza Sarmin, N. & bin Ibrahim, O., 2015. A multi-criteria collaborative filtering recommender system for the tourism domain using Expectation Maximization (EM) and PCA–ANFIS. *Electronic Commerce Research and Applications,* Volume 14, pp. 542-562.

O'Connor, M. & Herlocker, J., 1999. *"Clustering items for collaborative filtering.*. Berkeley, UC Berkeley.

Ouyang, D., Li, D. & Li, Q., 2006. Cross-validation and non-parametric k nearest-neighbour estimation. *Econometrics Journal,* 9(2006), pp. 448-471.

Pazzani, M. J. & Billsus, D., 2007. Content-based recommendation systems. In: *The adaptive web.* s.l.:Springer Berlin Heidelberg, pp. 325-341.

Portugal, I., Alencar, P. & Cowan, D., 2015. *The Use of Machine Learning Algorithms in Recommender Systems: A Systematic Review,* s.l.: arXiv preprint arXiv:1511.05263.

Powers, D., 2011. Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation. *Journal of Machine Learning Technologies,* 2(1), pp. 37-63.

Pu, P., Chen, L. & Hu, R., 2011. *A user-centric evaluation framework for recommender systems.*. s.l., ACM, pp. 157-164.

Qamar, A. & Gaussier, E., 2012. RELIEF Algorithm and Similarity Learning for k-NN. *International Journal of Computer Information Systems and Industrial Management Applications,* 4(2012), pp. 445-458.

Qamar, A.-M., Gaussier, E., Chevallet, J.-P. & Lim, J., 2008. *Similarity Learning for Nearest Neighbor Classification.* s.l., Data Mining, 2008. ICDM'08. Eighth IEEE International Conference, IEEE, pp. 983-988.

Ricci, F., Rokach, L. & Shapira, R., 2015. Recommender systems: Introduction and challenges.. In: *Recommender Systems Handbook.* s.l.:Springer US, pp. 1-34.

Ries, E., 2011. *The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses.* s.l.:Crown Business.

Rokach, L. & Maimon, O., 2007. Introduction to Decision Trees. In: *Data Mining with Decision Trees: Theory and Applications:*. s.l.:World scientific, p. 264.

Sarwar, B., Karyois, G., Konstan, J. & Riedl, J., 2000. *Analysis of Recommendation Algorithms for E-Commerce.* s.l., Proceedings of the 2nd ACM conference on Electronic commerce.

Sarwar, B., Karypis, G., Konstan, J. & Riedl, J., 2001. *Item-based collaborative filtering recommendation algorithms.* Hong Kong, Proceedings of the 10th international conference on World Wide Web, pp. 285-295.

Schröder, G., Thiele, M. & Lehner, W., 2011. *Setting Goals and Choosing Metrics for Recommender System Evaluations.* Chicago, ACM.

Sharda, R., Asamoah, D. & Ponna, N., 2013. Business Analytics: Research and Teaching Perspectives. *Int. Conf. on Information Technology Interfaces,* Volume 35.

Srivastava, J., Desikan, P. & Kumar, V., 2005. Web Mining - Concepts, Applications, and Research. In: *Foundations and advances in data mining.* s.l.:Springer Berlin Heidelberg, pp. 275-307.

Tripathi, P., Agarwal, R. & Vashishtha, T., 2016. *Review of job recommender system using big data analytics.* s.l., Computing for Sustainable Global Development (INDIACom), 2016 3rd International Conference on.

Watson, H. J., 2014. Tutorial: Big Data Analytics: Concepts,Technologies, and Applications. *Communications of the Association for Information Systems,* 34(1), pp. 1247-1268.

Xue, G.-R.et al., 2005. *Scalable collaborative filtering using cluster-based smoothing.* Salvador, ACM.

Yu, H., Liu, C. & Zhang, F., 2011. Reciprocal Recommendation Algorithm for the Field of of Recruitment. *Journal of Information & Computational Science 8,* 16(2011), pp. 4061-4068.

Zhang, M. & Hurley, N., 2008. *Avoiding monotony: improving the diversity of recommendation lists.*. s.l., ACM, pp. 123-130.

Zhang, T. & Iyengar, V. S., 2002. Recommender Systems Using Linear Classifiers. *Journal of Machine Learning Research,* 2(2002), pp. 313-334.

Appendix A – Candidate Features

Feature	Data Category	Date Type	Collection Method [Manual/Automatic]	Is collected today	Entity	Type
Driver license	Categorical	String	Automatic	Yes	Candidate	Attribute
Truck License	Boolean	Boolean	Automatic	Yes	Candidate	Attribute
Certificate	Categorical	String	Automatic	Yes	Candidate	Attribute
Current Employment status	Categorical	String	Automatic	Yes	Candidate	Attribute
Student	Boolean	Boolean	Automatic	Yes	Candidate	Attribute
Skills/Competencies	Categorical	String[]	Automatic	Yes	Candidate	Attribute
Education	Categorical	String[]	Automatic	Yes	Candidate	Attribute
Description (Keywords)	Text	String	Automatic	Yes	Candidate	Attribute
Age	Numerical	Integer	Automatic	Yes	Candidate	Attribute
Gender	Categorical	String	Automatic	Yes	Candidate	Attribute
School grades	Numerical		Automatic	No	Candidate	Attribute
Work experience/Roles	Categorical	String[]	Automatic	Yes	Candidate	Attribute
Location	Numerical	Float[2]	Automatic	Yes	Candidate	Attribute
Languages	Categorical	String[]	Automatic	Yes	Candidate	Attribute
Psykometrika	Numerical		Automatic	No	Candidate	Attribute
Available days	Categorical	Numerical[][]	Automatic	Yes	Candidate	Attribute
Degree of availability (nr of hours per week)	Numerical	Float	Automatic	Yes	Candidate	Attribute
Jappa contract status	Boolean	Boolean	Automatic	Yes	Candidate	Attribute
Current School	Categorical	String	Automatic	Yes	Candidate	Attribute
Nr of votes	Numerical	Integer	Automatic	Yes	Candidate	Attribute
Rating Score	Numerical	Float	Automatic	Yes	Candidate	Attribute
Tags	Categorical	String[]	Automatic	Yes	Candidate	Attribute
Profile Quality	Numerical		Automatic	No	Candidate	Attribute
Reviews (Keywords)	Text		Automatic	No	Candidate	Attribute
Flexible	Boolean		Automatic	No	Candidate	Attribute
Allergies	Categorical		Automatic	No	Candidate	Attribute
Access to own car	Boolean		Automatic	No	Candidate	Attribute
Job/Title	Categorical	String	Automatic	Yes	Candidate	Preference
Industry Field	Categorical	String[]	Automatic	Yes	Candidate	Preference
Salary	Numerical	Integer	Automatic	Yes	Candidate	Preference

Work hours	Numerical	Integer	Automatic	Yes	Candidate	Preference
Time to time report	Numerical	Float	Automatic	Yes	Candidate	Transaction
Occupancy	Numerical		Automatic	No	Candidate	Transaction
Name of pools	Categorical	String[]	Automatic	Yes	Candidate	Transaction
Number of pools	Numerical	Integer	Automatic	Yes	Candidate	Transaction
Chat Behavior	Categorical		Automatic	No	Candidate	Transaction
Flexibility (Short notice)	Numerical		Automatic	No	Candidate	Transaction
Platform	Categorical	String	Automatic	Yes	Candidate	Transaction
Ratings	Numerical		Automatic	No	Candidate	Transaction
Previous Assignments	Categorical	String[]	Automatic	Yes	Candidate	Transaction
Employment Length (Per Job)	Numerical	Float[]	Automatic	Yes	Candidate	Transaction
Nr of worked hours (per week)	Numerical	Float	Automatic	Yes	Candidate	Transaction
Average Salary	Numerical	Float	Automatic	Yes	Candidate	Transaction
Visited company page	Categorical		Automatic	No	Candidate	Transaction
Response time	Numerical	Float	Automatic	Yes	Candidate	Transaction

Appendix B – Employer Features

Feature	Data Category	Data Type	Collection Method [Manual/Automatic]	Is collected today	Entity	Type
Description (Keywords)	Text	String	Automatic	Yes	Employer	Attribute
Industry Field	Categorical	String[]	Automatic	Yes	Employer	Attribute
Location (City)	Categorical	String	Automatic	Yes	Employer	Attribute
Pay Scale	Numerical	Integer	Automatic	Yes	Employer	Attribute
Company Structure	Categorical	String	Automatic	Yes	Employer	Attribute
Nr of votes	Numerical	Integer	Automatic	Yes	Employer	Attribute
Rating Score	Numerical		Automatic	No	Employer	Attribute
Special Needs	Text		Automatic	No	Employer	Attribute
Unsocial Hours	Numerical		Automatic	No	Employer	Attribute
Branding	Categorical		Automatic	No	Employer	Attribute
Position (Title)	Categorical		Automatic	No	Employer	Attribute
Reviews (Keywords)	Text		Automatic	No	Employer	Attribute
Allergies	Categorical		Automatic	No	Employer	Attribute
Age	Numerical		Automatic	No	Employer	Preferences
Gender	Numerical		Automatic	No	Employer	Preferences
Work Experience	Categorical		Automatic	No	Employer	Preferences
Current employer status	Boolean		Automatic	No	Employer	Preferences
Availability	Categorical		Automatic	No	Employer	Preferences
Skills	Categorical		Automatic	No	Employer	Preferences
Driver license	Categorical		Automatic	No	Employer	Preferences
Certificate	Categorical		Automatic	No	Employer	Preferences
Personality	Categorical		Automatic	No	Employer	Preferences
Title/Role	Categorical		Automatic	No	Employer	Preferences
Alerts	Categorical	String	Automatic	Yes	Employer	Transaction
Candidate profile view	Categorical		Automatic	No	Employer	Transaction
Contract (time)	Numerical	Float	Automatic	Yes	Employer	Transaction
Rating	Numerical	Float	Automatic	Yes	Employer	Transaction
Previous Offers	Categorical	String[]	Automatic	Yes	Employer	Transaction
Candidate Hires	Categorical	String[]	Automatic	Yes	Employer	Transaction
Candidate Bookmarks	Categorical	String[]	Automatic	Yes	Employer	Transaction
Search History	Categorical	String[]	Automatic	Yes	Employer	Transaction

Item Tags	Categorical	String[]	Automatic	Yes	Employer	Transaction
Add to Pool	Categorical	String[]	Automatic	Yes	Employer	Transaction
Meeting	Categorical	String[]	Automatic	Yes	Employer	Transaction
Tags	Categorical	String[]	Automatic	Yes	Employer	Transaction
Recommend candidate to other branch	Categorical		Automatic	No	Employer	Transaction
Non-Occupied Work Shifts	Categorical		Automatic	No	Employer	Transaction
Response to recommendation	Categorical		Automatic	No	Employer	Transaction
Denied Time Report	Numerical	Float	Automatic	Yes	Employer	Transaction
Platform	Categorical	String	Automatic	Yes	Employer	Transaction

Appendix C – System Configurations

System 1 Configuration

Algorithm	Parameter	Setting
K-means Clustering	Variant	Auto
	Max Iterations	300
	n_init	10
	Tolerance	0.0001
Nearest Neighbor Classifier	Variant	Brute
	n_neighbors	5
	Similarity Function (Metric)	Standardized Euclidian Distance
	Weighted	False
Decision Tree Classifier	Max depth	none
	Max features	none
	Min_impurity_split	10-6
	Min_samples split	2
	Min_weight_fraction_leaf	0.0
	Presort	false
	Splitter	best
Multinomial Naïve Bayes	Alpha	1
	Class_prior	none
	Fit_prior	true

System 2 Configuration

Algorithm	Parameter	Setting
K-Nearest Neighbor	Brute	Auto
	Similarity Function (Metric)	Standardized Euclidian Distance
	Weighting	False

System 3 Configuration

Algorithm	Parameter	Setting
K-Nearest Neighbor	Variant	Brute
	Similarity Function (Metric)	Cosine Similarity
	Weighting	True