UTRECHT UNIVERSITY

Using Information Extraction and Evolutionary Algorithms to Improve Matchmaking on the Labor Market

by

Koen Rodenburg

Supervision
Rogier van Eijk, Utrecht University
Harold Kasperink, NCIM-Groep

Master's Thesis
30 ECTS

Cognitive Artificial Intelligence
Faculty of Humanities

August 2014

UTRECHT UNIVERSITY

Cognitive Artificial Intelligence

Abstract

by Koen Rodenburg

This thesis identifies a significant lack of attention to using unprocessed documents as input for automated matchmakers on the labor market. To address this lack of attention, a job matching application is proposed that uses unprocessed Curriculum Vitae (CVs) and job descriptions as input, avoiding the great restrictions that many other approaches described in the literature pose on the data. This is achieved by parsing the documents using Information Extraction techniques. Afterwards, an Evolutionary Algorithm is used to optimize the weight (relative importance) of finding a match on each extracted field.

The obtained results show that there is a significant difference in performance between the application described in this thesis and statistical full-content matching. This indicates that the identified lack of attention to unprocessed CVs and job descriptions is unjust, as this approach still has unexplored potential. An interesting suggestion for future work is to combine the Information Extraction component from this thesis with the rule and ontology based approaches currently popular in the literature. This way, the latter approaches can elevate the previously imposed restrictions on the input data, and accept unprocessed CVs and job descriptions.

Acknowledgements

I would like to thank NCIM-Groep for giving me the opportunity to be with the company for ten months, first in the context of a preparatory internship, and then to build a job matching application, conduct an experiment, and write this thesis. I am grateful to the company's employees, who have made my stay a very enjoyable one. Specifically, I want to thank COO/CTO Harold Kasperink for being my supervisor from the end of NCIM-Groep. Although a very busy man, he always managed to make time for me when I needed it most.

From the end of Utrecht University, I am grateful to dr. Rogier van Eijk for supervising my thesis. He has given me valuable advice from a scientific point of view, which helped me shape this thesis into what it is now. I would also like to thank dr. ir. Jan Broersen, who has supervised my internship, but had to step down in order to fully focus on his project on Responsible Intelligent Systems[1].

Finally, I want to extend my gratitude to all of those, mainly friends and family, whom I have lectured about (and may have bothered with) my project, its progress, and the hurdles I encountered. I have gained valuable insights by discussing my thesis, sometimes already by simply trying to explain where I got stuck.

[1]REINS - `http://www.projects.science.uu.nl/reins/`

Artificial Intelligence is the attempt to get real machines to behave like the ones in the movies.

Russell Beale
Professor of Human-Computer Interaction
Univeristy of Birmingham

Contents

Chapter 1

Introduction

If you think it's expensive to hire a professional to do the job, wait until you hire an amateur.

Paul Neal 'Red' Adair
Founder of *Red Adair Co., Inc.*

At a certain point in life, people will be faced with the challenge of finding a job. The ideal job needs to fit one's skills, level of education and vocational interests, among other factors, and must also be fulfilling and challenging at the same time. Similarly, companies continuously face the challenge of selecting the best available candidate for any vacant job. In order to achieve a successful match between a candidate and a job, there must be agreement between supplied and demanded characteristics, which include skills, level of education, and previous experience.

Existing research in the field of automated matchmaking on the job market has almost exclusively focused on matching between manually structured candidate and job profiles [1]. Usually, these studies require profiles to be available in a relational database, inserted through a web form with separate fields for every aspect of a CV or vacancy. In reality, these structured profiles are often not available, and it is very impractical to acquire structured profiles directly from the source or to convert plain text to structured data manually. This gap between research and practice leads to software applications that only cover a subset of the available candidates, or it results in the decision to revert to manual matchmaking. The lack of attention to plain text CVs and job descriptions is significant, since it prevents automation of a tedious and error-prone task that is currently carried out manually. Filling this matchmaking void with a working program would increase efficiency in the recruitment process [2] and may even improve the financial performance of companies using the system [3].

In this Master's Thesis, which concludes my education in Cognitive Artificial Intelligence at Utrecht University, I will address the under-examination of matchmaking between unstructured CVs and vacancies. I will propose and build a software application that uses Artificial Intelligence techniques to automate parts of the process of selecting suitable candidates for available jobs, using unprocessed CVs and unprocessed vacancies as input. The research will be conducted in combination with an internship at NCIM-Groep[1], giving me access to an environment where the application is to be deployed, including resources such as CVs and job descriptions, and interaction with potential users of the application.

The rest of this chapter is structured as follows. It starts with defining the problem context into more detail. Then, the current standing of the literature concerning this particular issue will be examined. Having a solid understanding of the problem and the approaches that have been taken previously, I will propose an application that attempts to fill the identified void in automated matchmaking on the job market and a corresponding research question, therewith determining the scope of the current research.

1.1 Problem context

Mobility and flexibility on the labor market, respectively the percentage of the workforce that changes jobs and the percentage of the workforce that is on a temporary contract during a certain period, have been on the rise in The Netherlands, especially since the turn of the century [4, 5]. This is in accordance with a policy adopted over 40 years ago by the 34 Western nations that form the Organisation for Economic Cooperation and Development (OECD), which entails the promotion of mobility and flexibility in the labor market [6]. The advent of widespread access to the Internet has played a significant role in this development, as it has been shown that workers with more, faster and easier access to job advertisements and to the ability to apply to a job are more likely to change occupations than those who do not have access to the virtually unlimited web of data that is the Internet [7].

Since the onset of the worldwide financial and economic crisis in 2008, unemployment rates went through the roof throughout the European Union [8] and the United States of America [9]. As a result the labor market saw, and continues to see, a large influx of people searching for a suitable occupation. At the same time, as of 2013 in the European Union there is still a deficit of 5.9 million jobs, or 1.6%, versus the last full quarter of 2008 before the financial crisis broke out [10].

[1]`http://www.ncim-groep.nl`

Over the past few decades, the Internet has come to play a larger and larger role in job recruitment. As of 2008, 32% of hires in large, high-profile United States-based firms can be attributed to the Internet, of which about two-thirds comprises applications through the corporate website. In contrast, the number of hires that can be traced back to print media has dropped to a mere 3%. The largest sources with a combined 38% of all hires are internal transfers and promotions, and the remaining 27% can be attributed to referrals through employees, alumni and vendors [11]. In 2013, the largest German IT companies posted nearly all of their vacancies on their corporate websites, around 80% of their vacancies on on-line job boards such as Monster.de, and just over 37% of job openings were also posted on Social Media. Actual hires can be traced back to on-line job boards in 43% of the cases, while 21% is attributable to the corporate website and another 11% to Social Media [12]. In other words, three quarters of actual hires originate from an on-line source, making the Internet decidedly the largest source of successful applications.

As a result of a higher percentage of the workforce changing jobs each year, unemployment rates going up, and vacancies spreading faster and further through the Internet, the number of applicants per vacancy has risen [13, 14], and the process of selecting suitable candidates for vacant jobs has hence become more difficult during the past decades. When faced with a lot of applicants per vacant position, the number of applicants considered, or the extent of the search, increases. Consequently, recruiters have been proven to spend less time per applicant in such situations, decreasing the intensity of the search, which leads to less informed hiring decisions [15]. Conversely, it has been shown that in the current fast-moving, competitive labor market, companies that closely align their recruiting and selection policies with their corporate goals can positively influence their financial position [2], and that companies that are able to quickly make job offers have an advantage over companies that are not, since opportunities to hire qualified applicants are not squandered due to lengthy response times [3]. Thus, selecting suitable candidates for vacancies has in recent times become more difficult, but also more important for companies.

NCIM-Groep is a company based in Leidschendam, The Netherlands, that specializes in seconding[2] highly educated IT professionals, predominantly in the defense and security, energy and utilities, telecommunications, transportation and media, and technical automation sectors. Since the company's primary business model is seconding employees, freelancers, and new applicants, it is a crucial task for NCIM-Groep to select suitable candidates for available jobs and assignments. Currently, the Sales and Human Resources (HR) department at NCIM-Groep performs the matchmaking process

[2] "The temporary transfer of an official or worker to another position or employment" (Oxford Dictionaries [16])

completely manually, similar to the manner in which many other companies operate [3]. Based on the requested hard and soft skills, a candidate that fulfils the requirements is sought in a vast and ever expanding collection of available candidates, represented by a hard copy of their Curriculum Vitae (CV). With this in mind, it is clear how NCIM-Groep and many companies like it would benefit from the creation of an application that partially automates this repetitive and error-prone process by suggesting possible matches between a candidate and an available job. Not only would efficiency be enhanced, leaving more time for Sales and Human Resources staff to spend on candidates qualified for the job rather than sifting through many irrelevant CVs [3]; it was also shown that investment in HR automation can lead to a significant return of investment through improved financial performance [2].

1.2 Status quaestionis

There are two dimensions along which the compatibility between a candidate and a job can be determined. The first and foremost is person-job (P-J) fit; measuring to what extent an individual possesses the qualities, skills and experience required for carrying out the tasks associated with the job [17]. The second dimension is person-environment (P-E) fit, including sub-dimensions such as person-organization fit, person-group or person-team fit, and person-vocation fit; respectively attempting to measure an individual's fit relative to the policies of the company, prospective colleagues, and his or her vocational interests [18]. While P-J fit is the primary dimension, a better P-E fit can improve the performances of both the newly hired employee and the members of the team or department he or she started working in [1].

In recent literature, there exists a widespread belief that a good P-J fit often depends on underlying aspects that are generally hard to formalize and measure [1, 17, 19, i.a.], corroborating the accounts given by the Sales and Human Resources staff at NCIM-Groep. It is therefore difficult to pinpoint exactly which aspects of a person's CV determine whether he or she is compatible with the available job, apart from obvious factors such as a requirement that a candidate has a university-level education. Quantifying P-E fit is even more difficult, as it requires an extensive analysis of both the person and the environment [20], where further problems arise, including normalizing the subjective answers people give about their qualities and needs [21]. The several issues the measurement of P-E fit faces, make it nearly impossible to perform in a relatively short period of time or on a relatively large scale.

Although the use of on-line job portals for recruiting new employees has exponentially grown over the past few decades [11, 12], their true potential is still extremely underexploited. The majority of companies use on-line job boards exclusively in a similar way to traditional print media: to publish their job offerings. Only 44% of the companies that are active on these job portals also use them to actively search for candidates [22]. For those looking for a job and the companies that take the time to actively search for future employees, it is not a trivial task to find a dream job or a perfect candidate, respectively. In fact, the Boolean search capabilities that most on-line job portals offer are often inadequate for finding an ideal match [1, 22, 23], as they for instance do not take inflections into account (e.g. 'programmer' is not returned in a search for 'programming'). With the notable exception of location, which appears to currently be the single most important factor in on-line job selection, no prioritization can be observed for attributes or search options – they are all given the same importance by the job boards [24, 25]. Given the previously mentioned subtleties in matchmaking between potential employees and vacant jobs, it is clear that the current searching and matching capabilities of on-line job boards lead to suboptimal results and missed opportunities in terms of a good candidate or job being seized up by a competitor. The described limitations may be an explanation for the fact that on-line job boards are in practice mainly used to make a crude pre-selection of candidates based on standard queries, which define only concrete requirements such as a desired level of education [1].

Up until the turn of the century, both scientific and general interest in information systems in Human Resources Management was very low, and available systems were almost exclusively employed for administrative tasks, with only 21% of companies entering more data than just contact information in their applicant tracking systems [22]. Since then, the specific challenge of matching job searchers and vacant jobs has experienced mild scientific interest [1, 14, 26–28, i.a.]. All attempts readily available in the literature are exclusively aimed at finding matches with a good P-J fit, by determining in various manners the similarity between an available candidate's CV and a job description of a vacant job, rather than focusing on the much more volatile P-E fit. While CVs and job descriptions can be somewhat vague or even incomplete, they can still be deemed relatively reliable and are thus suitable for matching. Since it is likely that any discrepancies with reality will be revealed during a potential job interview, and will even put the candidate in a negative light, people have an incentive to provide rich and reasonably accurate information on their CVs [29]. Conversely, it is nearly impossible to describe, let alone formalize, one's personal characteristics that influence P-E fit, such as work ethics or humor (though admittedly, formalizing humor is an active field of research [30]). This makes it nearly impossible to automate matching along this dimension, and explains the apparent absence of scientific work in this area.

Roughly speaking, two approaches can be distinguished in relation to matching people to jobs in the Human Resources domain. The first approach is statistical text-based similarity search, which is closest to the modus operandi of big corporate players such as Google, Inc. Here, matchmaking is performed by solely determining the degree of textual similarity between candidates' CVs and available job descriptions. This is done by using the full content of a CV as input for a search through all available job descriptions, or vice versa [31]. This approach will be designated as statistical full-content search for the rest of this thesis. The strength of textual-similarity based techniques is that they are very flexible in the sense that they are able to easily discover non-exact matches and are not domain dependent. A downside to this first approach is that precision[3] and recall[4] are inferior compared to the second approach [32].

The second approach, which is more recent and more prominent than the first approach, is the logic and semantics-based similarity approach. This approach makes use of ontologies of the domain in order to derive semantic relationships between concepts in the document. This is done to abstract from irrelevant linguistic factors such as formulation and synonyms. Then, an attempt is made to find a job description with a semantic structure analogous to that of a certain CV [14, 26, 28]. This approach has much higher precision and recall than the first approach, but the resulting inflexibility is a steep trade-off. The use of an ontology firmly restricts an application to a single domain, and creating an ontology for a new domain is a complex and time consuming task [32]. A matchmaking attempt in the on-line dating domain [33], which is closely related to the human resources domain in that it comprises matching a set of qualities to a set of requirements, formalizes the relationships between fields from the request document (comparable with a job description) to the available profiles (comparable with CVs) in a very specific way. This enforces an additional inflexibility in terms of domain specificity and the addition or removal of certain fields. More importantly, defining these exact relationships and their relative importance for the Human Resources domain is considered to be infeasible in recent literature, as mentioned earlier [1, 17, 19, i.a.].

The vast majority of current human resources applications that attempt to find optimal matches between candidates and vacant jobs require the data to be presented in a neatly structured database [14, 26, 28, 32, 33]. In most cases, the researchers have developed a user interface, such as a web application, in which candidates are requested to fill out a form, asking about contact information, work history, enjoyed education, and other pieces of information usually present on a Curriculum Vitae [1]. By others,

[3]Measure of exactness: The number of relevant results divided by the total number of results.

[4]Measure of completeness: The number of relevant retrieved results divided by the total number of existing relevant results.

this ideal situation is judged currently unrealistic [27], while some even call it "an escape for constraining real data into an available technique, [rather] than a real solution" [33].

1.3 Current research

It is virtually impossible to formalize personal characteristics, such as work ethics and humor, that determine whether there is a Person-Environment fit [1, 18]. This in contrast to characteristics that determine Person-Job fit, including level of education and previous experience. While the latter characteristics can easily be listed on one's CV, it is very difficult to describe someone's work ethics beyond hollow phrases like "I am a team player who is also great at working alone" and "I do not have a 9-to-5 mentality". As a result, information about relevant personal characteristics is not readily available in a job application. Recruiters generally invite potential candidates based on P-J fit, using information from their CVs, and try to determine the P-E fit during the subsequent job interview [1, 17]. Due to the aforementioned difficulties and in line with previous research, this thesis will exclusively focus on matching along the P-J fit dimension, and leave the judgment of P-E fit to the recruiter.

Another practice adopted by the vast majority of previous attempts is working under the assumption that the information from CVs and job descriptions is available in a structured database [14, 26, 28, 32, 33]. In reality, CVs and job descriptions are not inherently structured, but rather only available in free-form text or sometimes, if supplied by the applicant, semi-structured using tables. In most previous research, candidates were required to fill out standardized forms with information from their CV [1, 33]. In practice, only 12% of applications for the typical job are received through standardized web-based forms [22]. Forcing candidates to retype their CV every single time they apply for a job is an additional hurdle in the application process, which may lead to good candidates applying elsewhere instead. This approach has therefore been criticized as being unrealistic and not a real solution to the problem [27, 33]. The envisioned software application is to be used by recruiters, and it is not realistic to require them to spend an extensive amount of time on mind-numbing data entry for every single candidate. That way, the application would further reduce the efficiency of the matchmaking process rather than improve the efficiency. A final alternative is to require applicants to provide a unique identifier of their account on an on-line job board, the information of which can then be imported by the application. Complicating factors here are the existence of a myriad of on-line job boards, every one of them slightly different from the others, and the fact that many people do not have a profile on any such website. One of the largest and most well-known job boards, LinkedIn[5], has a market penetration of only 32% in the United States, with most other countries having significantly smaller fractions of the population active on this board [34, 35]. Given the previous, it is clear that there currently exists no practical alternative to entering unprocessed CVs and job descriptions in their original unstructured or semi-structured form.

The proposed application will, conform to the aforementioned observations, be a matchmaking application using unprocessed CVs and job descriptions, matching them along the P-J dimension. Since pure content-based search has shown disappointing results, leading to the attention shift to ontology-based approaches [32], I propose to divide the matchmaking process into two stages. During the first stage, the document is to be parsed by the application. Using techniques from Information Extraction, a subfield of Natural Language Processing, the contents of the document will be divided into an array of fields, including name, location, education, and work experience. This is the stage that helps the application to be able to work with unstructured input documents such as raw CVs, voiding the need for manual pre-processing. The second stage is the actual matchmaking stage. During this stage, each of the fields extracted from a CV will be paired up with the corresponding extracted field of a job description. A search query will determine the degree of similarity between the two documents, based on the field-by-field similarities. Since the literature deems it infeasible to determine the relative importance of these fields beforehand [1, 17, 19], an Evolutionary Algorithm will be used to optimize the relative weights of the field pairs. The Evolutionary Algorithm will use previously generated matches, scored by a human expert, for its fitness function. The mentioned methods and techniques will be discussed into more detail in Chapter 2, and a detailed description of the architecture of the application will follow in Chapter 3.

Since NCIM-Groep, my internship provider, seconds IT-professionals, the data that will be provided to me for testing purposes will be in the IT domain. While this effectively constrains my research to this domain, the overall structure of CVs remains the same across domains, and it will therefore be relatively easy to generalize to other domains. Where domain specific elements are introduced, they will be made replaceable by elements specific for another domain, or a general element for matching across multiple domains.

The scope of the current research is confined by the aforementioned decisions, and can be captured by the following research question, which will be central to the rest of this thesis:

[5]`http://www.linkedin.com`

Can matching between unprocessed CVs and job descriptions using statistical full content search be improved upon, by structuring these documents into fields using Information Extraction, and optimizing the relative weights of these fields using an Evolutionary Algorithm?

1.4 Hypothesis

Given the comprehensive advantages that automated matchmaking on the job market has for companies [2, 3], the low adoption rate of Human Resources systems that perform such tasks [1, 22] is striking. While this may be partly attributable to businesses failing to commercialize such products, recent literature describing promising, but nowhere near perfect results [1, 14, 26–28] supports the conclusion that the capabilities of current systems are simply inadequate.

Early attempts to automated job matching were based on statistical methods [31]. While promising initial results were shown, recall and precision of these statistical full-content search systems remained at a suboptimal level [32]. Even though corporate mogul Google, Inc. continues to outperform current rule-based systems in all sorts of domains with its statistical approach [36, 37], current research into job matching has ventured into the logic and ontology based realm. This shift of attention has led to improving results [32], but at the same time has imposed great restrictions on the input data [14, 26, 28, 33].

The low adoption rate of applications that perform matchmaking on the job market may partly be caused by the relatively low success rates and, more importantly, the mentioned restrictions that are imposed. The proposed system tries to improve on the early statistical approaches, partly using the philosophy of the current approaches, but avoiding many of the restrictions that are inherent to that philosophy. More concretely, the current research tries to automatically structure the data, as structured data can be seen as an important factor in the relatively good results of the current rule and ontology based approaches.

The hypothesis for this thesis is that the current approach yields better results than the early statistical full-content search attempts to solve this problem. This approach will use the same paradigm of matching structured profiles as successfully applied in other recent approaches to the problem, as opposed to unstructured profiles. At the same time, it avoids many of the restrictions posed by most other approaches to matchmaking on the job market currently prominent in the literature, most importantly the requirement that data needs to be structured manually before being entered into the application.

The optimization of the weights of the different fields from the structured profiles using user-scored examples eliminates the problem of not being able to determine the optimal weights beforehand.

1.5 Outline

This chapter has identified a significant lack of attention in the domain of automated matchmaking on the labor market, given the potential improvement of efficiency and financial performance. More exactly, using unprocessed CVs and vacancies as input to job matchmaking applications has been neglected in recent years due to disappointing early results. In this thesis, I propose an alternative approach to job matching that uses unprocessed CVs and vacancies as input. The proposed architecture will be implemented in order to answer the research question and assess the validity of the hypothesis posed in the previous section. The performance of the application will be compared to both human performance and performance of the statistical full content search method, the latter of which has previously yielded slightly disappointing results, leading to the abandonment of the approach that uses unprocessed documents as input.

In Chapter 2, relevant methods and techniques from the field of Artificial Intelligence will be set forth, and the architecture that combines these techniques into an application will be laid out in Chapter 3. The experimental setup of the current research will be described in Chapter 4. Chapter 5 presents the results yielded by the experiment. A discussion of these results, the employed methods and its position in the field will be presented in Chapter 6, and this thesis will be concluded in Chapter 7.

Chapter 2

Methods

We are drowning in information,
but starving for wisdom.

Edward Osborne Wilson
American biologist and author

Chapter 2 describes the methods and techniques that will be used to build the proposed matchmaking application. Since the matching process is divided in two phases, this chapter is comprised of two parts. The first section explicates Information Extraction, a method from the domain of Natural Language Processing. The second section focuses on Evolutionary Algorithms, which are an optimization method.

Information Extraction focuses on identifying relevant pieces of information and their interrelationships in a text. A famous test bed for this technique is the detection of the central event in a short newspaper article, including key people and organizations. The current purpose of Information Extraction is to extract candidate profiles from CVs and job profiles from job descriptions, including name, role, and relevant skills. That way, the profiles can be stored in a standardized and structured way, paving the way for the actual matchmaking step. In essence, Information Extraction is the practice of cleverly combining other Natural Language Processing techniques, including pattern matching, tokenization, and Named Entity Detection, in such a way that the relevant pieces of text are identified. These techniques are described in the first half of this chapter.

The second phase of matchmaking starts with two databases of structured candidate and job profiles, respectively, and consists of finding the right job for the right candidate. This is done by executing a complex search query, with sub-queries for each field in a candidate profile paired up with the field in a job profile. As described in Chapter 1, it is impossible to accurately determine which of the field-pairs are more important and which are less important [1, 17, 19]. An Evolutionary Algorithm will therefore be used as an optimization technique for the relative importance of the field-pairs. Since Evolutionary Algorithms are thus a main component of the second part of the matchmaking application, the second half of this chapter will describe them in detail.

This chapter will give, as described, a thorough overview of the relevant methods and techniques from the field of Artificial Intelligence. Chapter 3 will subsequently describe in further detail how these methods and techniques will work together to achieve the ultimate goal; matchmaking between CVs and job descriptions.

2.1 Information Extraction

Information Extraction (IE) is an application of Natural Language Processing, a part of the broader process of Text Mining, and focuses on distilling information including names, dates and locations from naturally occurring text. As Jurafsky and Martin put it, the "process of Information Extraction ... turns the unstructured information embedded in texts into structured data. More concretely, Information Extraction is an effective way to populate the contents of a relational database." [38, p.759] This is exactly the purpose of the first phase of the current matchmaking process. Unless denoted otherwise, the Information Extraction techniques described in this section are adapted from one of the leading textbooks on Natural Language Processing [38].

Given the relatively narrow domain of matchmaking on the labor market, relevant documents correspond to fairly common, stereotypical situations: profiles of candidates and jobs. These abstract situations can be captured in a script, which is a prototypical representation of the sequences of events, participants and roles in a situation [39, 40]. Scripts can be represented as a template consisting of a fixed set of slots that for each new text need to be filled with the appropriate values from that text. Maintaining the parallel with relational databases, a template can be seen as consisting of the column headers of the table. For each new document, a new row is inserted, of which the cells need to be filled with appropriate pieces of text from the document. As an example, a simplified version of one of the scripts used in the current application is given in Table 2.1.

Table 2.1: Simplified version of the candidate script used in the application.

CANDIDATE	Name	Donald Duck
	Roles	Coin cleaner, Factory worker, Driver
	Location	Ducktown

The explicit representation of scripts or templates imposes strong expectations about the contents of the text, which can facilitate the assignment of entities into roles and relationships, and help draw inferences about implicit aspects of the text. For instance, when filling the template in Table 2.1, an application should expect to find job titles and names of people and cities in the text at hand, which helps disambiguation where multiple meanings are plausible. Once the elements of interest have been identified, they may be directly transported to the template or go through an additional processing step first. This additional processing can for instance consist of standardizing dates and times or amounts, or the comparison of the identified elements with gazetteers or other types of ontologies, as will be described later.

Statistical sequence labeling approaches to template filling are surprisingly effective [41, 42]. These approaches use statistical classifiers (often Hidden Markov Models) that determine the probability that a certain piece of text fits in a cell in the template, based on its textual and contextual similarity to previously identified slot-fillers. However, the constrained nature of the tasks at hand is a great factor in the impressive results. The documents used for these approaches were all small, relevant and homogeneous. They always contained fillers for the slots of interest, which were also limited in number, and the document size left little room for distracting elements [38]. Situations where dealing with considerably more complex documents is required, such as the problem currently at hand, need an approach more grounded in the structure of the text and language [38, 43].

The structure of a document often gives a first indication of the relations between different pieces of the text. CVs and job descriptions inherently provide a lot of structural information, as most of these documents consist of small pieces of text, usually laid out in tables or short paragraphs denoted by specific headings. This makes the process of Information Extraction both easier and more complex. Structural information can be extremely useful in determining which parts of a text belong together, and what they are about. However, most IE methods determine the function and relevance of pieces of text by looking at the context. In tables and other similar structures, most of the context and even functional words have been stripped out. This leaves only bare nouns and fragments of phrases, which makes it harder to recognize the function and relevance of this piece of text.

Extracting semantic notions such as the aforementioned names and locations from natural language is a non-trivial task. In order to achieve results nonetheless, Information Extraction solutions are often clever combinations of more elementary methods commonly applied in Natural Language Processing. To the extent they are used in the application created for this thesis, these methods are introduced in the following sections. The first method discussed is the relatively rigid, but nevertheless effective, Pattern Matching. Next, Tokenization is the preparatory step for more advanced methods including Named Entity Detection and the use of ontologies. How these techniques are fused into a full-fledged Information Extraction system will be explicated in Chapter 3, where the architecture of the application and the combination of methods are described.

2.1.1 Pattern matching

Most CVs and job descriptions have a table at the outset of the document, containing basic information such as the candidate's name, date of birth and residence, or the title, contact details and location of the job. The availability of this information in a semi-structured fashion can be exploited by detecting the table headers using pattern matching. In case of a horizontally oriented table, with its headings in the first column, the cells in the same row as a certain heading can, with high confidence, be taken as the value corresponding to that heading. Indeed, if the first column of a table in a CV contains cells with terms like 'Name' and 'Location', the cells adjacent to those cells most likely contain the name and location of the candidate.

Pattern matching is a technique aimed at finding an exact pattern in a string of text. In computer science, these patterns are usually defined as Regular Expressions, a language used for specifying search strings. Regular Expressions were designed and proven equivalent to Finite State Automata (FSA) by Kleene [44], and inspired by the Turing machine [45] and the McCulloch-Pitts Neuron [46]. Formally, Regular Expressions are a means to characterize sets of strings in an algebraic notation, and they are used to recognize any string from the defined set in a corpus, the search space. In other words, Regular Expressions can be seen as a search string, where sets of keywords or key phrases can be compounded into a single query. The building blocks of Regular Expressions are standard alphanumeric characters that are to be recognized, and special characters to perform operations on the alphanumeric characters. These operations include negation ($\hat{}$), disjunction ($[AB]$ or $A|B$) and repetition ($+$ or $*$). Additionally, there are special sequences that help identify special positions in the text, such as word boundaries (any position between a letter and a non-letter, $\backslash b$), digits (0-9, $\backslash d$) or 'word-characters' (mostly letters, $\backslash w$). Table 2.2 gives several examples of Regular Expressions. More specific examples of Regular Expressions as they are used in the application described in this thesis will be given in section 3.2.1.

Table 2.2: Regular Expression examples.

Regular Expression	Results
$abaa$	abaa, aabaa, babaa, abaaa, abaab, aabaaa, babaab, ...
$\backslash b\ abaa\ \backslash b$	abaa, -abaa-, ...
$abaa\|b$	abaa, b, ...
$aba(a\|b)$	abaa, abab, ...
$aba*$	aba, abaa, abaaa, abaaaa, ...
$(aba)*$	aba, abaaba, abaabaaba, ...

2.1.2 Tokenization

Tokenization is the task of segmenting running text into tokens; words or sometimes groups of closely related words. In English, and other languages using the Latin alphabet, this is a simple task at first glance, since words are separated by a special character: the whitespace character. Other languages, such as Chinese and Japanese, do not have a specific character to denote word separations, making tokenization more difficult. However, the whitespace character separator is not a sufficient tokenizer for English either, as this would separate the words 'San Francisco', while they arguably form a single token, as they have no meaning separately. Similarly, from the perspective of tokenization, punctuation could be seen as forming word boundaries, were it not for special cases such as hyphens ('the 28-year-old woman'), abbreviations ('i.e.'), dates ('01/01/2014'), numbers ('123,456.78'), and websites ('http://www.google.com'). Finally, depending on the intended use of the tokenized text, a tokenizer may be required to resolve clitics and contractions such as "we're" in English or "j'ai" in French.

Tokenization is used as the first processing step in many Natural Language Processing applications, as most techniques that are more advanced require the input text to be tokenized. For that reason, tokenization plays an important role as first step for pieces of running text in the current application. Albeit not always the case, tabular text often consists of single words or very short phrases, making it ineligible for tokenization. However, the previously described Regular Expressions (Section 2.1.1) already cover most of the relevant information that is presented in a tabular way.

Contemporary implementations of tokenizers are rule-based and language specific, often supplemented by a dictionary containing multi-word expressions that should not be separated, such as the aforementioned 'San Francisco' and 'Natural Language Processing'. Often, these dictionaries are partially constructed using Named Entity Detection (described in Section 2.1.3) because most entries in such dictionaries are in fact named entities. At the same time, Named Entitiy Detection requires tokenized text as input. Tokenization and Named Entity Detection thus have a very close relationship.

2.1.3 Named Entity Detection

The term *named entity* can be applied to anything that can be referred to with a proper name. Examples of this are people, organizations, and geographical entities such as countries, cities, streets and other named locations. Also commonly designated as named entities are temporal expressions such as dates and times, and numerical expressions such as amounts of money. As is often the case in natural language, named entities can be ambiguous. When 'Washington' is mentioned in a text, this can refer to a myriad of things; one of the Founding Fathers of the United States, the capital of the United States, a metonymy for the U.S. government, the U.S. state, or one of many streets, hospitals and other places named after the aforementioned.

Named Entity Detection is the task of detecting these named entities in a text and identify the category of named entities they belong in. Hereby, disambiguation needs to be performed often, such as for the aforementioned named entity 'Washington'. The context of named entities is crucial in the disambiguation process, as the words around named entities often give important clues about the category of that named entity. For instance, if a text contains the phrase "I talked to Washington", it becomes immediately clear that this "Washington" cannot be a state, town or other geographical entity, but rather that it is most likely a person.

The problem faced in Named Entity Detection (NED), also known as Named Entity Recognition, is similar to that of Part Of Speech (POS) taggers. Where Named Entity Detectors assign (sequences of) words to several categories of named entities, POS taggers assign words to their corresponding lexical classes. Since parts of speech and categories of named entities form relatively small closed sets, POS tagging and NED essentially boil down to classification problems. With this in mind, statistical classifiers such as Hidden Markov Models (HMMs) are often applied to these problems. HMMs (using the Viterbi algorithm) maximize the likelihood of words being of a certain class, and the probability of certain tag sequences occurring together, and assign the word(s) to the category with the highest probability. In order to train the HMMs, large annotated training texts are necessary, not uncommonly consisting of up to 15,000 example sentences. Tailoring these training sets to specific use cases is extremely labor intensive, leading to efforts to extract annotated data from readily available sources [47].

In the current application, NED is used to extract several types of named entities. If the name of the candidate has not been found using Regular Expressions as described in Section 2.1.1, an attempt is made to find the candidate's name using NED. This works especially well for common names that are represented in the example sentences. Names that are underrepresented in the example sentences, for instance foreign names, are less likely to be detected. In a similar way, the names of cities and towns are detected to determine the place of residency of the candidate, and the location of the job.

2.1.4 Ontologies

An ontology is an hierarchical representation of knowledge of a certain domain, denoting relevant concepts in that domain and their interrelationships. Since the previously mentioned methods are all based on either assumptions or probabilities, they are often not able to reach an accuracy of 100%. In certain situations, it can be practical to use ontologies as additional means of detecting certain entities or concepts related to the subject of the ontology.

The most widely accepted ontologies to be used in Natural Language Processing in general, and Named Entity Detection in particular, are gazetteers; geographical directories that alphabetically list the names of towns, regions, and landmarks that appear on a map of a certain region. Lists of persons and organizations are deemed less effective, since these would require a lot more maintenance because of their more volatile nature. Additionally, long lists containing more than the most well-known named entities only slow the detection process down, but do not significantly increase the performance [48].

Once a suitable ontology has been selected or created, its application is fairly straightforward. When concerning Named Entities, the gazetteer or ontology can be used before or after the Named Entity Detector. When applied first, the speed of the system can be decreased depending on the size of the ontology, since all terms need to be searched for in the text. Since only exact matches are found by utilizing the ontology, Named Entity Detection will afterwards be employed to detect false negatives. When the ontology is applied last, the detected Named Entities are compared to those defined by the ontology, increasing accuracy by removing any false positives and thus confirming true positives. Depending on the type of problem, a suitable order of the two methods is determined.

Ontologies are used by the current application in the form of gazetteers to find place names, in addition to those found by Named Entity Detection. Furthermore, they are used to detect names of programming languages and other software applications, as well as educational institutions. How this is achieved exactly will be described in Section 3.2.3.

2.1.5 Post-processing

After the completion of Information Extraction, the values assigned to certain fields require post-processing. Hereby, two techniques from Natural Language Processing are utilized. The simplest of the two is stop word removal, which is the practice of removing words from a text that are not central to the topic. These words are usually the most common words, which appear in many texts without being indicative of its subject, and therefore are not instrumental to the searching process. Examples of stop words are 'the', 'and', 'of', and 'with'. Removing stop words from the sentence "Peter is a software dseveloper with experience" might, depending on the exact implementation, yield "Peter software developer experience".

A somewhat more complex technique is stemming. This technique reduces inflected or derived words to their stem, in such a way that the meaning of the word is not significantly destroyed. An example of stemming is reducing the words 'programming' and 'programmer' to the stem 'program'. The most elementary stemmers work using a look-up table, of which the size would need to be gigantic for use in a real-life application, whereas more sophisticated stemmers use a compound of rules to reach the correct stem, including the default removal of the '-ed', '-ing', and '-ly' suffixes. The de-facto standard for English stemming is the Porter stemming algorithm [49], whereas the Snowball stemming framework, which is also devised by Porter [50], is most commonly used for implementations of stemming in other languages.

2.2 Evolutionary Algorithms

The second part of this chapter describes Evolutionary Algorithms, which are a central part of the second phase of the matchmaking application: the actual matching step. Matchmaking occurs by executing a complex search query on the database of jobs, with sub-queries for each field in a candidate profile. The exact structure of these queries will be shown in Section 3.3. As described, the relative importance of the fields, and the corresponding sub-queries, cannot be determined beforehand. Evolutionary Algorithms are therefore used in the application to optimize the relative importance parameters for the search query, using manually scored matches.

2.2.1 Fundamentals

Evolutionary Algorithms are population-based stochastic metaheuristic optimization algorithms based on the Darwinian principles of evolution [51]. In his famous work *On The Origin Of Species* [52], Darwin laid out his theory of natural selection and survival of the fittest, which forms the basis of contemporary evolutionary biology. In a population of stable size, and faced with limited resources such as food or space, a struggle for life ensues. Due to variation within the population, some individuals are better adapted to their surroundings than others; they are fitter. Given their heightened ability to negotiate the environment, the fitter individuals have a higher chance of reproducing. The varying traits are hereditary through DNA, meaning that the advantageous traits will likely persist over generations, while disadvantageous traits exhibited by less fit individuals will slowly die out. Sexual reproduction combines relatively fit parents to form potentially fitter offspring with DNA, and thus traits, from both parents. Occasional mutations in the DNA help maintain variation in the gene pool by introducing new genes.

The principles of evolution as described here form the inspiration for Evolutionary Algorithms. A fixed-size population of candidate solutions (individuals) is maintained for many generations. Within each generation, candidate solutions compete with each other for limited resources; a finite number of positions in the subsequent generation. New candidate solutions are generated by sexual reproduction. Two parent solutions, selected randomly, but usually with a bias for a higher fitness, recombine their genes to form offspring solutions. The now enlarged population is trimmed down to the fixed size, based on the relative performance of each candidate solution [53]. Hereby, there is a bias towards the best performing individual, while often some suboptimal solutions are carried over as well, to preserve a heterogeneous population. Additionally, variance in the population is promoted by a mutation operator, which randomly alters genes between generations.

Evolutionary Algorithms offer a lot of freedom, leaving it up to the user to choose how candidate solutions are represented, and which genetic operators are implemented and in what way exactly. There is no need for examples labeled with their ideal output, as only relative performance is taken into account by Evolutionary Algorithms, expressed as an individual's fitness. Offspring solutions are generated by recycling parts of parent solutions and the search space is expanded by random mutations. No restrictions are posed on the means of determining the fitness of a candidate solution, allowing for indirect evaluation by an external algorithm [53].

Additionally, there is a vast number of internal parameters that can be adjusted, such as the size of the population, the number of offspring, and probabilities of being selected as parent or surviving to the next generation [51, 53]. Almost all of these options are important to the functioning of the algorithm, with the means of representation and fitness determination arguably being the most crucial ones [54]. Careful consideration of the available options is thus necessary to implement an effective Evolutionary Algorithm.

According to Holland's Schema Theorem [55], schemata[1] that contribute more than averagely to the overall fitness increase exponentially in subsequent generations. Once

[1]Templates identifying subsets of candidate solutions with similarities at certain positions

the Evolutionary Algorithm has ran through a sufficient number of generations, the candidate solutions with the highest fitness can be said to be *probably approximately correct* [56], a term that stems from computational learning theory [57]. Computational learning theory assumes that a solution that is consistent with a large set of examples is a solution that, with a high probability, is approximately correct. This is based on the hypothesis that a candidate solution that is seriously wrong will almost certainly be proven faulty already after a small amount of evaluations [54].

2.2.2 Representation

In Evolutionary Algorithms, the representation of individuals, their 'DNA', is one of the most crucial aspects [54], as it needs to be able to represent all possible solutions to the problem. The most elementary representation is a string of bits, but other alphabets such as letters, integers or even real-valued numbers can be used.

Depending on the type of problem, additional constraints can be applicable to the individuals. In the classical example of the Travelling Salesman Problem (TSP), the task is to optimize the route of a salesman travelling through several cities. The goal is to find the shortest route starting and finishing in the same city, while visiting every other city exactly one time. Thus, candidate solutions for the TSP are represented as arrays of names of cities that can contain every city exactly once, except for the starting point.

2.2.3 Fitness

As mentioned, the fitness of a candidate solution represents its performance on the task at hand. The architecture of Evolutionary Algorithm leaves a lot of freedom regarding the implementation of the fitness function, the only constraints being that the output is numerical, and consistent over multiple evaluations of the same individual. Unlike many other methods, including Artificial Neural Networks, Evolutionary Algorithms therefore do not require examples labeled with the corresponding ideal output. This paves the way for external evaluations, where not the actual parameters determine the fitness, but the performance of an algorithm executed with those parameters.

Fitness functions range from elementary operations, such as addition or multiplication of integer genes, to very complex external functions executed with the values of the individual's genes as parameters. The outcome of the fitness function can be defined as a reward or a punishment; in the former case a higher fitness score is better and in the latter case a lower fitness score is better. For the TSP, the fitness function is a cumulative punishment, as its goal is to minimize the total distance travelled, which is equal to the sum of distances.

2.2.4 Genetic operators

Genetic operators are the means of creating candidate solutions that differ from the currently available candidate solutions. Due to the fact that fit individuals have a higher probability of being selected as parents, offspring candidate solutions likely express advantageous traits inherited from their parents, possibly leading to combinations that perform even better.

The most basic genetic operator is mutation. It takes a single parent and, depending on the implementation of the individual, randomly alters an elementary part of the genome. In bit string representations, mutation can entail randomly flipping a bit. In letter, integer or real values, mutation generally consists of replacing a value at a random position with a random other accepted value. In the case of the TSP, this is impossible since this would always result in an invalid solution. Mutation is therefore often implemented for the TSP as swapping two random cities. As this example indicates, careful consideration of the genetic operators in relation to the representation is necessary to avoid the creation of invalid solutions.

The other genetic operator is crossover, the recombination operator. Here, two parents are selected to mate, resulting in offspring with combined traits of both parents. There are several types of the crossover operator, the effectiveness of each of them depending on the representation of the individuals. One-point crossover most closely resembles biological recombination, as it divides the parents' genomes in two halves and swaps the halves on one side. Two-point and N-point crossover work in a similar fashion, with multiple places where the genomes are sliced. These forms of crossover are especially useful when adjacent genes are in some way related to each other. In cases where adjacent genes are independent of each other, uniform crossover may be a more appropriate operator. This method treats each gene individually, swapping the genes from the two parents at that position with a certain probability.

More complex restrictions on the representation of individuals require more complex crossover operators, as the standard crossover operations result in invalid sequences. Examples of more complex crossover operators are order crossover, partially mapped crossover, position crossover and maximal preservative crossover. These forms of crossover are suitable for recombination of individuals where, for instance, each gene can only appear once, such as in the TSP.

Depending on the desired ratio between exploitation, sticking with the best solution so far, and exploration, trying out new solutions, genetic operators can be applied on their own or in combination. Concretely, this means that an exploratory implementation of an Evolutionary Algorithm can choose to generate the entire new population through crossover, after which the mutation operator is applied to all offspring individuals. This leads to an offspring generation that differs relatively much from the parent generation, focusing on the exploration aspect of the search for the optimal solution. Alternatively, letting individuals participate only in either crossover or mutation lies the emphasis more on exploitation, decreasing the probability of destroying high quality schemata and thus focusing on exploitation. Many Evolutionary Algorithms implement a combination of exploration and exploitation, decreasing the probability of crossover and mutation occurring as the generations progress.

2.2.5 Selection

Once parents have been selected by means of their fitness and genetic operators have been applied to create offspring, the size of the population has greatly increased. In order to implement the principle of limited resources, a selection algorithm will each generation trim the population down to its original size. As stipulated by the theory of survival of the fittest, the relatively unfit individuals have a smaller chance of making the cut.

The harshest selection method is truncation selection, whereby simply the fittest N (N being the original population size) individuals are carried over and the rest is discarded. To maintain the heterogeneity of the population, more moderate algorithms have been devised. In tournament selection, random small fixed-size subsets of the population are taken, of which the individual with the highest fitness is carried over to the next generation. This is repeated until the next generation's population is full. A third algorithm is fitness proportionate selection, also known as roulette wheel selection, where the probability of an individual i being selected is equal to its normalized fitness (Equation 2.1).

$$\frac{f_i}{\sum_{j=1}^{N} f_j} \tag{2.1}$$

An adjustment that can be made to any selection operator is elitist selection, where the highest-scoring individuals of the parent generation are carried over as-is, and the rest of the individuals are selected by means of one of the other described selection operators.

2.2.6 Generations

Calculating the fitness, applying genetic operators and trimming down the population using selection are processes that are repeated every generation that the algorithm runs. In order to reach optimal results, this compound of operations needs to be repeated many times. Depending on the complexity of the problem, the number of generations needed before the algorithm converges on an optimum can vary from several dozens to hundreds or even thousands of generations. The feasibility of such vast numbers of generations primarily depends on the size of the population and the implementation of the fitness function. With the processing power of modern computers, evolving a relatively small population with a relatively simple fitness function over a thousand generations falls perfectly within the range of possibilities. However, if calculating the fitness of a single individual is more complex, and takes for instance one second, calculating 100 fitnesses over 1000 generations takes 100.000 seconds, or almost 30 hours.

There are several possible methods for specifying the stop condition of the Evolutionary Algorithm. The first and most obvious option is limiting the number of generations to a fixed amount. The individual that has the best fitness at that point is taken as the final solution. Another option is to determine a target fitness, and to let the algorithm run until the best individual reaches that target fitness. Choosing the second option can be dangerous, since it is not always certain that the target fitness will ever be reached. To avoid this problem, a combination of the two is sometimes made; the algorithm runs until it reaches a target fitness, unless a predetermined maximum number of generations is reached. The final option is to not limit the maximum number of generations, or the desired fitness, but to let the algorithm run until no improvement is seen over a number of generations.

2.3 Conclusion

Chapter 2 has described the two main methods that will be used in the proposed application for matchmaking on the labor market; Information Extraction and Evolutionary Algorithms. Information Extraction is a sub-field of Natural Language Processing, and focuses on identifying relevant words and phrases in a text, by using lower-order techniques such as pattern matching, Named Entity Detection and applying ontologies. Evolutionary Algorithms come from the field of Computational Intelligence and are population-based stochastic optimization algorithms based on the principles of natural evolution as first identified by Charles Darwin [52].

With these techniques in mind, the next chapter will describe how the proposed application is built. All components of the application will be described, including how they cooperate in selecting suitable matches between candidates and jobs.

Chapter 3

Application

Any sufficiently advanced bug is indistinguishable from a feature.

Rich Kulawiec

This chapter contains a detailed description of the architecture of the application, including the roles of the different techniques as laid out in Chapter 2, their position in the software application, and exactly how they cooperate towards reaching the ultimate goal: suggesting matches between CVs and job descriptions.

Due to the various tasks that need to be performed by the application, it consists of several subroutines. From an Artificial Intelligence point of view, the two main components are the document parsers, which use Information Extraction (as described in Section 2.1), and the parameter optimizer, which consists of an Evolutionary Algorithm (see Section 2.2). Other important components are the data storage and matcher, and included as auxiliary components are a web crawler and a web application that functions as a user interface.

Figure 3.1 gives a schematic overview of the architecture of the application, highlighting the positions and interrelationships of the different components. The main horizontal axis represents the actual matchmaking process: A CV is taken from the data storage, a query is constructed by the matcher and executed on the stored job profiles, resulting in a number of suggested matches. The two smaller vertical axes depict the information extraction steps that fill the relational databases with the appropriate pieces of text from the CVs and job descriptions. Finally, the circular path is the feedback loop that provides the parameters that determine the relative importance of the different fields in the match query. This feedback loop is carried out by an Evolutionary Algorithm that learns over time through suggested matches that are scored by the user.

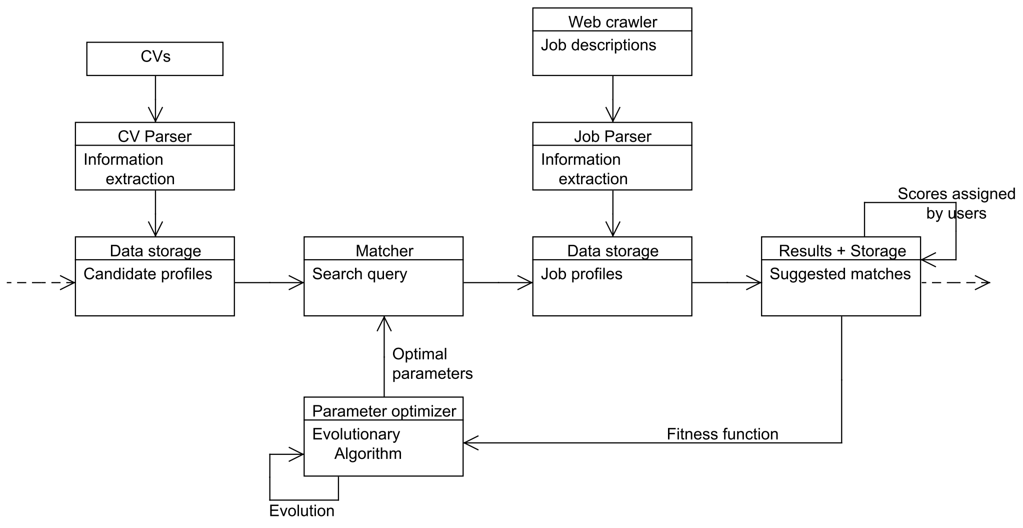


FIGURE 3.1: Schematic overview of the architecture of the application.

3.1 Data storage

The data storage forms the core of the matchmaking application, since this is the place where crawled and parsed CVs and job descriptions are stored, until they are picked up to be matched to each other. It also stores the matches that result from the query created by the matcher, as well as the scores assigned to them by the user. These scores can then be used by the Evolutionary Algorithm from the feedback loop to optimize the search parameters. The data storage as used in the application is represented in Figure 3.2.

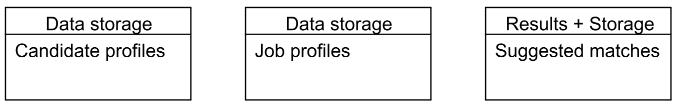


FIGURE 3.2: Data storage as used in the application.

Although at first glance the obvious choice for the data storage in the application is a traditional Relational DataBase Management System (RDBMS) such as MySQL, Big Data search solution ElasticSearch[1] is elected as the data storage in this thesis. ElasticSearch is based on search library Apache Lucene[2], and offers storage capabilities that are functionally equivalent to an RDBMS. ElasticSearch is especially apt at dealing with large quantities of data. On large data sets, ElasticSearch outperforms RDBMSs at data retrieval and analysis, including pattern discovery and search. The sequential

[1]`http://www.elasticsearch.org`
[2]`http://lucene.apache.org/core`

nature of these processes in traditional systems results in unrealistically long processing times [58], whereas ElasticSearch takes a more distributed approach, which yields a faster system. An additional advantage of ElasticSearch is that it offers a wide range of processing capabilities, including an advanced query language, right out of the box. While this is not especially relevant in terms of data storage, it makes building the matcher (as presented in Section 3.3) a lot less complex.

The data storage consists of four indices, which are ElasticSearch's equivalent to tables; one for candidate profiles, one for job profiles, one for generated matches, and one for the Evolutionary Algorithm's population. Candidate profiles consist of the fields ID, name, date of birth, location, education, roles, programming languages, software, and full content of the CV. Job profiles contain fields for the ID, job title, location, keywords, and the full content of the job description. Matches contain fields for the names and IDs of the candidate and job that have been suggested as a good pairing, the similarity score from the match algorithm, and the score assigned by the user. Finally, the population, which will be described into more detail later in this chapter, contains numerical weights for each of the fields in a candidate profile, plus the calculated fitness for that individual.

3.2 Document parsers

Although internally complex components, the role of the document parsers, one for CVs and one for job descriptions, in the application is relatively simple. As previously mentioned, and depicted in Figure 3.3, their role is to extract the relevant pieces of content from CVs and job descriptions in order to store candidate and job profiles in a relational fashion.

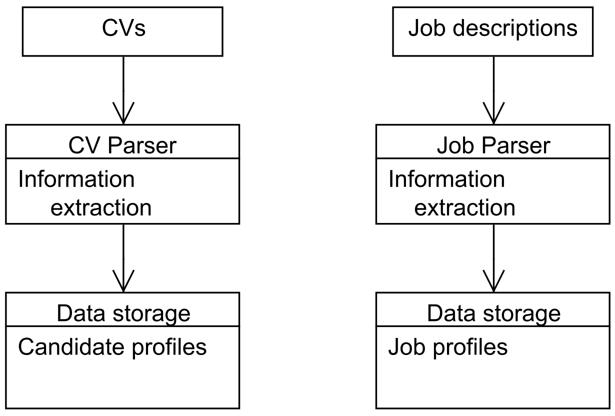
The document parsers are written in the programming language Java, and use content analysis toolkit Apache Tika[3] to extract the contents from the uploaded documents, including Microsoft Word (.doc), Adobe Portable Document Format (.pdf), and plain text (.txt) files, in the form of XHTML. Subsequently, HTML Document Object Model (DOM) parser JSoup[4] is employed to traverse the XHTML.

3.2.1 Pattern matching

As described in Section 2.1.1, pattern matching is used to extract information from tables that are usually found at the top of CVs and job descriptions. First, it needs to

[3]`http://tika.apache.org`

[4]`http://www.jsoup.org`



(A) CV parser (B) Job parser

FIGURE 3.3: Architectural view of the CV parser and Job parser.

be determined whether certain keywords are present in a table. These keywords include name, date of birth and location, as this is the information that is most often presented in a tabular fashion at the top of a CV. If it has been established that some of the selected keywords are present in a table, the orientation of the table needs to be determined, as it can be oriented horizontally or vertically. If all keywords occur in the same column, the table and its key-value pairs are oriented horizontally, whereas when all keywords occur in the same row, the table and its key-value pairs are oriented vertically. In case of a horizontally oriented table, the cell to the right of the cell containing the name keyword is assumed to contain the candidate's name, the cell to the right of the cell containing the location keyword is assumed to contain the location of the candidate, etcetera. In case of a vertically oriented table, the cells below the cells containing the keywords are taken to contain the corresponding value.

TABLE 3.1: Examples of horizontal and vertical tables.

(A) Horizontal table

Name	Donald Duck
Residence	Ducktown

(B) Vertical table

Name	**Residence**
Donald Duck	Ducktown

Below, the Regular Expression to detect keywords associated with candidates' names in CVs is displayed. [] and | indicate disjunction, ? indicates optionality, $\backslash s$ indicates a space character, and $*$ denotes 'zero or more'. The entire Regular Expression thus searches for a string optionally starting with either 'voor', 'roep', 'volledige' or 'achter', all optionally starting with a capital. After that, an optional space follows, and then either 'naam' or 'namen', optionally with a capital. This entire string may be followed by zero or more spaces, colons, semicolons, commas or periods.

$$([Vv]oor|[Rr]oep|[Vv]olledige|[Aa]chter)?\ \backslash s?\ [Nn]a(am|men)\ (\backslash s|:|;|,|.)* \tag{3.1}$$

3.2.2 Tokenization and Named Entiy Detection

Pieces of text that are not structured into tables, and text in tables wherein no keywords can be recognized, need to be tokenized in order to apply any further processing methods. For this purpose, Apache OpenNLP[5] with training models[6] in the Dutch language are used.

After completion of the tokenizer, the same Natural Language Processing software package is used for Named Entity Detection in the tokenized text. Particularly, the types of named entities that is searched for are names of persons, names of places, and dates. In cases where the pattern matching approach has not been able to isolate the candidate's name, residence or date of birth, the detected names, locations and dates are used to fill these slots. Thereby, it is taken into consideration that the values should be plausible; a date that is less than twenty years ago, or more than seventy years ago, is not likely to be a candidate's date of birth.

3.2.3 Ontologies

As laid out in Section 2.1.4, ontologies can be used as an additional means of detecting concepts related to the subject of an ontology. They can either serve as extra check after pattern matching or named entity detection, to detect false positives, or to find results when the other two methods fail, to detect false negatives. In the former case, the detected pieces of text are compared to entities in the ontology. In the latter case, the entire text is compared to the ontology in order to find the words in the text that also occur in the ontology.

The most widely accepted ontologies used in Natural Language Processing are gazetteers, or lists of geographical entities. Here, a gazetteer is used to check detected locations, and if none were detected, to find suitable locations in the text. Ontologies are also used to detect keywords specific to the IT domain: programming language and software applications, since those are important characteristics on the CV of an IT professional.

[5]`http://opennlp.apache.org`

[6]`http://opennlp.sourceforge.net/models-1.5/`

Since it is deemed infeasible to maintain most large ontologies manually [47], DBPedia Spotlight[7] is used as means for detecting entities from an ontology in the text. This application uses DBPedia[8], structured information extracted automatically from Wikipedia[9], as its main ontology. Due to the massively collaborative nature of Wikipedia, one can expect this ontology to be relatively up-to-date, especially on well-known and popular subjects. Since programming languages and software applications are standard categories on Wikipedia and thus in DBPedia, it is a trivial task to extract entities from those categories from text using DBPedia Spotlight.

3.2.4 Post-processing

After all the information has been extracted, some of the fields require post-processing. Examples of such fields include fields that contain dates and fields that contain relatively long texts.

In natural language, there is a myriad ways of denoting a certain date, some of which require disambiguation. For instance, the date 'June 5th, 2014' can also be referred to as 'the 5th of June 2014', '06/05/14' in the United States, or '05-06-14' in most other parts of the world. For post-processing dates, natural language date parser Natty[10] is used. This is a small application that accepts dates in any of the aforementioned formats, and many more, and parses those dates into an abstract, object-oriented representation. Afterwards, one can specify a format wherein the parsed date should be output.

For textual fields, it can be useful to apply stemming and stop-word removal. As described in Section 2.1.5, stemming is the practice of removing inflections, leaving only the bare stem of the words: 'programmer' and 'programming' are both stripped to 'program' and will thus lead to a match. Stop-word removal is, as indicated by the name, the process of stripping the text of all irrelevant stop-words, mostly functional words, such as 'the', 'this', and 'for'. Both techniques are built-in in ElasticSearch, and they are used with the default parameters as provided by the search library.

3.3 Matcher

The matcher is the component where, from a functional point of view, the magic of the application happens; this is the component that finds relevant job descriptions for

[7]`https://github.com/dbpedia-spotlight/dbpedia-spotlight/wiki`
[8]`http://dbpedia.org`
[9]`http://www.wikipedia.org`
[10]`http://natty.joestelmach.com/`

available candidates and presents them as suggested matches. However, because the other components of the application carry out so many supporting tasks, all that is left to do for the matcher is combine all of the information into a search query and execute this query on the database of job profiles. This process is represented in Figure 3.4.

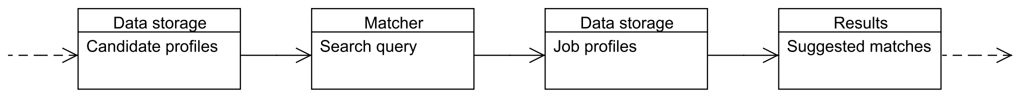


FIGURE 3.4: Architectural view of the matcher.

The starting point for the matcher is the database of candidate profiles. First, candidates for which matches need to be suggested are selected. The search query is subsequently built using the content of the candidate profile and the weight values provided by the parameter optimizer (for the latter, see Section 3.4). The search query is schematically represented in Figure 3.5.

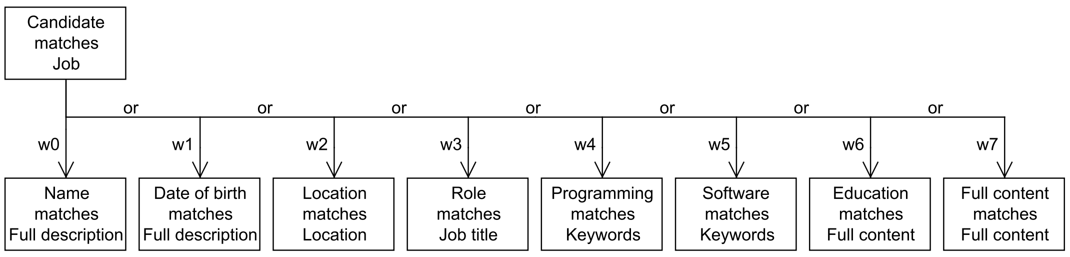

Candidate matches Job
or
or
or
or
or
or
or
w0
w1
w2
w3
w4
w5
w6
w7
Name matches Full description
Date of birth matches Full description
Location matches Location
Role matches Job title
Programming matches Keywords
Software matches Keywords
Education matches Full content
Full content matches Full content

FIGURE 3.5: Schematic representation of the search query.

The top left square in Figure 3.5 represents the main query: the compatibility of the candidate with jobs. The main query is disjunctively divided in eight sub-queries, represented by the eight squares below it. In this case, a disjunction is preferred over a conjunction, because if a field is absent in the candidate profile or if the term is not present in the job profile, then this does not necessarily mean that the candidate and job as a whole are incompatible. Each of the sub-queries takes the content of one of the fields in the candidate profile, and searches the corresponding field of the current job profile to determine the measure of similarity. The labels $w0$ through $w7$ are the weights assigned to the different fields from the candidate profile and their corresponding sub-query by the parameter optimizer. The overall measure of compatibility between a candidate and a job is the sum of measures of similarity from each sub-query multiplied by their corresponding weights. Potential matches are then ordered by compatibility score, whereby the highest scoring candidate-job matches are suggested first.

3.4 Parameter optimizer

Optimizing the search parameters, the weights of the different fields and their corresponding sub-queries as laid out in the previous section, is an important task in the application. Without this subroutine, it is unclear which of the fields are more important and which are less important for determining the agreement between candidates and jobs. The parameter optimizer can be seen as a learning agent; an Artificial Intelligence technique will form the learning element, while the matcher from Section 3.3 forms the performance element used to assess the quality of potential solutions [54].

There are two leading sub-fields in Artificial Intelligence; symbolic AI and sub-symbolic AI. Symbolic AI, also dubbed Good Old-Fashioned AI (GOFAI), encodes aspects of intelligence in the form of symbols and rules, using logic and reasoning to make inferences; all notions that are central to the field of Artificial Intelligence [54]. The symbols are representations themselves, such as words, which stand for concepts in the real world, or have a straightforward mapping to a set of words, making them interpretable for humans. Sub-symbolic AI, on the other hand, often encodes knowledge in a for human beings less meaningful way; in sets of numerical patterns [51]. As a result, sub-symbolic AI algorithms often work as a black box, with no way for a human to trace the system's steps and find out why or how the system came to the given output. Both areas have their respective merits and their faults, and excel at solving different types of problems.

In the literature it is deemed infeasible to accurately determine a priori what the optimal distribution of weights should look like. Although intuitions can be somewhat helpful within a certain domain (for IT professionals, it is important to have computer and programming skills, but for construction workers, these skills are irrelevant), they are not sufficient to weigh all fields and determine their relative importance [1, 17, 19]. Therefore, it is impossible to provide the AI component with facts and rules that allow it to deduce potential solutions. This problem thus does not seem to have the characteristics that make GOFAI a suitable approach to solving it.

Many prominent sub-symbolic AI techniques are of the Computational Intelligence (CI) kind; probabilistic methods based on biological processes, such as Artificial Neural Networks (ANNs) and Evolutionary Algorithms. Other CI approaches include swarm intelligence and artificial immune systems. ANNs are networks of vastly interconnected nodes, inspired by the human brain [51]. These models are especially apt at pattern recognition tasks, solving classification problems; assigning inputs to one of a finite set of classes. These networks are trained by an updating algorithm, most commonly back-propagation, that calculates the error between the actual output and the desired output.

Using this error, the internal parameters are adjusted in such a way that when a similar input is encountered later on, the output will be more like the desired output. When using an ANN as the parameter optimizer, direct feedback about the performance of the network cannot be given, since there is no ideal output to compare the actual output to. Therefore, it is impossible to determine which part of the solution is responsible for settling on an incorrect output.

The task currently at hand is not one of classification, but of optimizing numerical parameters, something Evolutionary Algorithms (EAs) are well-suited for [51]. Based on the Darwinian principles of survival of the fittest, candidate solutions compete for a limited number of positions in the subsequent generations, where better solutions have a higher chance of being carried over. The crucial advantage that Evolutionary Algorithms have over Artificial Neural Networks is that a candidate solution does not need to be compared to some optimal outcome. Rather, EAs only take relative performance into account, posing no restrictions on the means of measuring the performance, which allows for indirect evaluation by an external algorithm [53]. While in the task at hand it is impossible to assess the quality of a parameter distribution, it is possible to determine whether this particular distribution yields the right matches, as manually judged by a human expert.

From the previous, it has become clear that an Evolutionary Algorithm is the most suitable technique to implement the parameter optimizer. In the current application this EA is implemented in the Java programming language. This optimization step constitutes the feedback loop in the system architecture, schematically depicted in Figure 3.6.

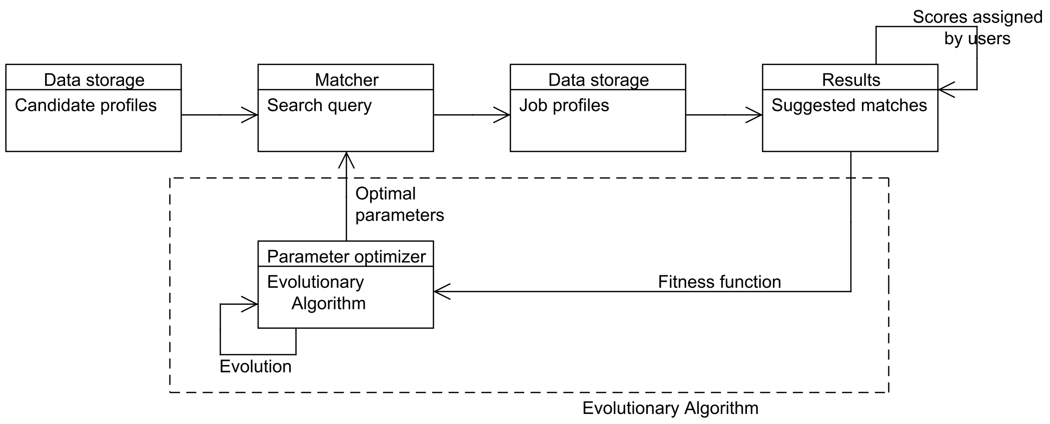


FIGURE 3.6: Architectural view of the parameter optimzer.

The horizontal axis in the figure is once again the matcher, from which the parameter optimizer cannot be seen independently. In the square labeled 'Results', matches are stored and suggested to the user, who then gets the opportunity to evaluate the correctness of the match by awarding a score to the match, ranging from a 5 for a very good match to a 1 for an unsuitable match.

The feedback from the user in the form of scored matches is used by the Evolutionary Algorithm to optimize the search parameters, in order to achieve results that are more in line with the user's evaluations. As explained in Section 2.2, Evolutionary Algorithms are population-based stochastic optimization engines. In the current application, the Evolutionary Algorithm's population consists of individuals that are vectors of real-valued weight factors. In other words, and individual is an array of real-valued weights $w0 \ldots w7$, as used in the search query in Figure 3.5. The population will evolve over a number of generations, during which the fitness of the individuals is determined by their performance on the scored matches. At any point in time, the matcher can request the currently most optimal individual and use these parameters to generate new matches.

Genetic operators that are used to generate new candidate solutions, or individuals, during evolution are mutation and uniform crossover. Mutation is the operator that ensures variety in the gene pool of the population; in other words, it ensures that new weight values are introduced, which can possibly be beneficial in terms of achieving more optimal solutions. Maintaining variety in the gene pool is necessary for evolution because otherwise, the algorithm would quickly converge to a population filled with clones of a single relatively well performing individual, after which no improvement will occur. During each generation, several individuals are selected for mutation. Of the selected individuals, two genes are randomly reinitialized. This is a relatively drastic mutation, but given the vast size of the search space, an emphasis on exploring many different solutions is necessary.

Crossover is the recombination operator, that combines two parents that are selected with a bias towards more optimal solutions. This way, partial solutions that are relatively fit are propagated over generations, and possibly combined with other relatively fit partial solutions to form even better overall solutions. In this case, the uniform crossover operator is used, since adjacent weights are independent of each other. Two parent individuals are selected, and for each field the weights are swapped between parents with a probability of 50%. This means that the two offspring individuals contain around half of the weights from one parent and the rest from the other parent.

The most complex aspect of this implementation of the Evolutionary Algorithm is the fitness function. As mentioned, the fitness of individuals is determined by their performance on scored matches, since a direct evaluation is impossible. This is effectuated by executing the search query from the matcher, and comparing the top results from this query with the top results indicated by the scores assigned by the user. This comparison is done by calculating the Mean Absolute Error (MAE) of the outcome of the query in relation to the user-scored matches. The MAE is a means of measuring how close predictions are to the actual outcome. The Mean Absolute Error is, as the name indicates, the mean of the absolute errors. It is calculated by dividing the sum of the absolute distances ($|e_{ij}|$) between the predicted positions (f_{ij}) and the actual positions (y_{ij}) by the number of predictions (m), and represented in Formula 3.2. Since the MAE is also used to evaluate the performance of the application in the experiment that will be conducted in light of this thesis, it will be discussed into further detail, including an example calculation, in Section 4.2.2.

$$MAE_i = \frac{1}{m}\sum_{j=1}^{m}|e_{ij}| = \frac{1}{m}\sum_{j=1}^{m}|f_{ij} - y_{ij}| \tag{3.2}$$

During each generation, the population needs to be trimmed down after offspring individuals have been generated. As a method of selection, this application uses elitist selection, carrying over the five individuals with the highest fitness, in combination with fitness proportionate selection. Both methods have been explained in Section 2.2.5. Proportionate selection ensures that more fit solutions have a higher probability of surviving, but also that less fit solution have a chance of being carried over to the next generation. This is again a means to emphasize exploration of the search space, by not only sticking with solutions that have already been proven to be relatively good.

3.5 Web crawler

The auxiliary component represented in the overall architecture in Figure 3.1 is the web crawler, singularly depicted in Figure 3.7. This crawler, built primarily in the programming language Python, scrapes job descriptions from several job portals and staffing websites generally used by NCIM-Groep.

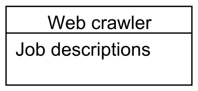


Figure 3.7: Architectural view of the web crawler.

The functionality of the web crawler is relatively simple. It monitors the RSS-feeds, or similar content listings, of relevant websites. After a fixed interval of time, the crawler determines whether new job descriptions of available jobs have been published. For each of the newly found job description, the relevant part of the page, excluding among other things the website header, is sent to the job parser, and the resulting job profile is added to the data storage.

3.6 Web application

Even though not represented in the architecture of the application as depicted in Figure 3.1, there is one other auxiliary component: a web application. This web application, essentially a graphical user interface (GUI) for the entire software application, enables the user to view and manipulate candidate profiles and job profiles. Additionally, suggested matches can be shown, after which the user is encouraged to assign a quality score to that match. This is important because the application will function optimally when new scores keep being added to suggested matches, so that the Evolutionary Algorithm can remain adjusting the search parameters to the user's preference.

3.7 Conclusion

Chapter 3 has described all the subroutines that are used to find suitable matches between CVs and job descriptions. This includes the data storage embedded in ElasticSearch, the document parsers that use Information Extraction, the matcher that implements the search query, and the parameter optimizer implemented as an Evolutionary Algorithm. This chapter has also given an architectural view of the application, highlighting the position and role of the various subroutines in the application.

Having created an automated matcher for the human resources domain, an experiment will be conducted to determine whether this application outperforms statistical full content search. The experiment that will be conducted is outlined in Chapter 4, after which the results are presented in Chapter 5.

Chapter 4

Experimental Setup

It's not an experiment if you know
it's going to work.

Jeffrey Preston "Jeff" Bezos
Founder of *Amazon.com*

In order to address the research question as posed in the Introduction, an experiment will be conducted that is designed to test the hypothesis as outlined in section 1.4. This chapter describes the experimental setup in detail, ensuring the reproducibility of the results that will be obtained. For the reader's convenience, the research question is repeated below:

> *Can matching between unprocessed CVs and job descriptions using statistical full content search be improved upon, by structuring these documents into fields using Information Extraction, and optimizing the relative weights of these fields using an Evolutionary Algorithm?*

To recapitulate, the hypothesis for this experiment is that the proposed method can indeed improve upon the performance by statistical full-content job search. Unlike rule- and ontology-based systems as described in section 1.2 however, which have also shown better results than the statistical full-content search approach, the current method will not impose strong restrictions on the input documents. As a result, if the hypothesis turns out to be correct, the application devised in this thesis has much more practical usability than the methods that do impose these restrictions.

4.1 Data sets

The data that will be used in the experiment will be representative sets of real-life CVs and job descriptions, as they are available to the Human Resourcing staff at NCIM-Groep. The documents will be selected from current requests for personnel, and from applicants and available employees of NCIM-Groep. These documents are usually Microsoft Word, Adobe Portable Document Format (PDF) or plain text files. The data sets will be entered into the different conditions of the experiment, as described in subsequent sections, in their original form, without any manual pre-processing such as mentioned in section 1.2 and used in many similar experiments.

For this experiment, the web crawler (Section 3.5) will be restricted to crawling a single one of the websites it is built for. The website IT-Staffing[1] was selected to form the source for the job descriptions in the experimental data set. This selection was made based on the fact that this website offers relatively rich information about current vacancies. The rationalization behind this selection criterion is that one cannot expect an application to extract information and suggest suitable matches when very little information is available, in the same way that it is implausible to expect this from a human expert matcher.

The size of the data sets will be kept relatively small, due to the fact that potential combinations need to be judged by a human expert in Condition 0 (see Section 4.2.1). The data set of job descriptions will be limited at 50, whereas 10 CVs will be randomly selected. This results in 500 potential combinations that need to be assessed by the expert, which is a very time-consuming task. Given the current circumstances, these sizes for the data sets are both small enough to remain feasible to score by the expert, and sufficiently large to yield informative results.

4.2 Conditions

The experiment that will be conducted has three conditions. The baseline condition, condition 0, will consist of manually ordering potential matches by a human expert. Since a gold standard of good and bad matches between candidates and jobs does not exist, the manual order from this condition will be taken as the actual ordering of relevance, that the automated systems try to replicate. Condition 1 is the condition wherein the matchmaking is performed by statistical full-content search. In condition 2, the Artificial Intelligence application as laid out in Chapter 3 will perform the matchmaking between available CVs and job descriptions.

[1]http://www.it-staffing.nl

4.2.1 Condition 0: Manual baseline

Condition 0 functions as a baseline condition, to create the gold standard that automated matchers should try to replicate. In this stage of the experiment a human expert, a member of the Human Resources staff at NCIM-Groep, manually orders a predetermined set of m job descriptions for each individual from a set of n CVs. More concretely, for each CV, a top five is made of the most suitable job descriptions. This is an important step in the experiment, as these shortlists per candidate are used to evaluate the accuracy reached in the other two conditions. Additionally, they are used in the Evolutionary Algorithm in Condition 2.

Having created these shortlists, the potential matches will be scored according to their position in the top 5 of the relevant candidate. The highest candidate-job combinations will get a score of 5, the second-highest a score of 4, and so forth. Matches that are not present on the shortlist of the relevant candidate receive a score of 0.

4.2.2 Condition 1: Statistical full-content search

The first experimental condition is the statistical full-content search condition. This will be executed in ElasticSearch, which is the search library that also forms the data storage and the basis for the Artificial Intelligence approach as used in Condition 2 and described in Chapter 3. The approach in Condition 1 is very simple, as the unprocessed full content of a CV is taken as search query, and the unprocessed full contents of the available job descriptions form the search space. This means that the job descriptions with the highest overall textual similarity to the CV will surface as suggested matches.

The search will be executed for each candidate, resulting in n shortlists of job descriptions, ordered by their presumed relevance to the corresponding CV. As this is the same format as has been manually created in Condition 0, the performance of the statistical full-content search can now, separately for each candidate, be compared to the standard that has been set by the human expert.

As a measure of comparison, the Mean Absolute Error will be used, which is a means of measuring how close predictions are to the actual outcomes. The error e of any candidate-job combination ij will be defined as the difference in score between the standard created in Condition 0 (actual outcome y_{ij}) and the results found in Condition 1 (predicted outcome f_{ij}). The Mean Absolute Error (MAE) for candidate i will then be the mean of the absolute errors of all candidate-job combinations ij associated with candidate i. The calculation of the MAEs is captured by Formula 4.1, where m is the number of job descriptions in the data set:

$$MAE_i = \frac{1}{m}\sum_{j=1}^{m}|e_{ij}| = \frac{1}{m}\sum_{j=1}^{m}|f_{ij} - y_{ij}| \tag{4.1}$$

Using the MAE to determine the correctness of a certain outcome is, in this case, more righteous than simply using a Boolean right or wrong measure (counting the number of correct predictions). This is emphasized in the example given in table 4.1. The second column gives the correct order of the letters (in the current experiment: jobs), while the third and fourth column give possible outcomes of a matchmaking attempt.

Table 4.1: Mean Absolute Error example:
Two possible orderings compared to the actual order.

Position	Actual order	Outcome 1	Outcome 2
1	A	B	E
2	B	C	D
3	C	D	C
4	D	E	B
5	E	A	A

Outcome 1 has reached the correct order, with the exception that A is a miss. In terms of a Boolean decision of right or wrong, this outcome has the lowest possible score of 0, since none of the letters are in the correct position. The absolute error, on the other hand, of A is $5 - 1 = 4$, since it is four rows away from its actual position. The absolute error for each other letter is 1, since their predicted positions are one off from their actual positions. This results in a MAE of:

$$MAE_{Outcome1} = \frac{4+1+1+1+1}{5} = \frac{8}{5} = 1.6 \tag{4.2}$$

Outcome 2 has completely reversed the correct order, but for putting C in the correct position, resulting in 1 correct prediction, it still gets a Boolean right or wrong score of 1. In terms of calculating the MAE for Outcome 2, however, the absolute errors for both A ($|5-1|$) and D ($|1-5|$) are 4. The absolute errors for B (—$4-2|$) and D ($|2-4|$) are 2, and C is in the correct position. The MAE is thus:

$$MAE_{Outcome2} = \frac{4+2+0+2+4}{5} = \frac{12}{5} = 2.4 \tag{4.3}$$

In terms of Boolean score, counting the correct predictions, Outcome 2 is better since it gets one letter in the correct position as opposed to none in Outcome 1. However, the Mean Absolute Error of Outcome 2 is much higher than that of Outcome 1, indicating that it is a worse solution in terms of ordering. As the latter conclusion is the intended one, since Outcome 1 is closer to the intended outcome, the Mean Absolute Error is in this situation the superior comparison method.

In the current experimental setup it is possible that jobs from the top five generated by the automated matcher do not appear in the top five as created by a human exert in Condition 0. Therefore, if such a situation occurs, the error for that particular job will be fixed at a value of 8. The error in these situations is fixed because otherwise, a job put at the fifth position by the automated matcher that does not appear in the gold standard top five would have an error of just 1. The number 8 results in a large penalization of gold standard top five jobs not showing up in the automated matcher top five, whereas a wrong position within the top five receives a smaller penalty.

The final outcome of Condition 1, the statistical full-content search condition, will thus be a list of n MAEs, as the Mean Absolute Error of matched jobs will be calculated for each CV in the data set individually. This list of MAEs will be compared with the list of MAEs produced by Condition 2, in order to test whether there is a significant difference and thus to answer the research question.

4.2.3 Condition 2: Artificial Intelligence

In Condition 2, the method of cross-validation will be used to create a top 5 of jobs for each CV in the data set. Afterwards, the Mean Absolute Error with respect to Condition 0 will be calculated in the same fashion as it was in Condition 1.

Cross-validation is a validation technique used in situations where a predictive model is trained on known data (the training set), after which it is tested or validated on novel or first seen data (the test set). The technique is used to assess the generalizability of the model to new, unseen data. In cross-validation, the data set of labeled examples is repetitively partitioned in a training set and a test set in a way that notorious problems such as overfitting[2] are avoided. These problems arise when choosing a fixed training set and test set, which used to be the accepted method. Cross-validation dynamically rotates examples from the training set to the test set and back, diminishing the chance of overfitting. Thus, cross-validation gives the most accurate idea of how the trained model would perform on independent data.

The power of cross-validation lies in repetition. Every iteration, the complete set of labeled examples is partitioned in a different way, resulting in any data point being part of the training set and of the test set at least once. During such an iteration, the model is trained with the currently selected training data, after which the test set will be used

[2]Training and testing the model on the same data, resulting in a model that has adapted so much to the training set that it is not generalizable to new data outside of this set.

to asses the performance of the model during that iteration. The generalizability is thus tested by determining the performance on an example that is not part of the training set. While other numbers can be chosen, k-fold cross-validation (k iterations) will be used here with $k = n = 10$. This is because of the relatively small set of examples, as it maximizes the size of the training set during each iteration. As a result of this choice for k, every CV will form the test set of size one exactly one time, which is why this method is also known as leave-one-out cross-validation.

During every iteration of the n-fold cross-validation, the Evolutionary Algorithm will optimize the search parameters using the training data selected for that iteration. A population with 50 individuals will be initialized, after it will evolve over 50 generations. These numbers are chosen in relation to the fact that a single fitness evaluation costs several seconds. As a result, this entire experimental condition will last an estimated $10 \cdot 50 \cdot 50 \cdot \sim 4s \approx 100,000s \approx 30h$. The remaining candidate will be used for testing, which once again entails drafting a top 5 of suggested matches for that candidate. The positions of the jobs in the top five for each candidate will be compared to the position of the job in the top five for that candidate from Condition 0, again using the Mean Absolute Error as described in Equation 4.1. Similar to Condition 1, the result of Condition 2 is a list of n MAEs.

4.3 Comparison

The hypothesis about this experiment, as presented in Section 1.4 of this thesis, is that the devised and created application as used in Condition 2 is a better matchmaker in the Human Resourcing domain than the statistical full-content search approach as used in Condition 1. In order to choose a method of comparison, normality of the results need to be tested. The Shapiro-Wilk Test [59] will be used to determine if the results from both conditions are normally distributed.

If the results are confirmed to be normally distributed, a t-test can be employed to detect a significant difference. Since significantly lower Mean Absolute Errors are expected in Condition 2 than in Condition 1, a one-tailed unpaired t-test will then be used to analyze the results. In relation to the small sample sizes, a significance level of $\alpha = 0.1$ will be chosen, indicating that there is a 10% chance of a Type 1 error.

The null hypothesis for the t-test is that there is no significant difference between the observed MAEs in the two conditions. The alternative hypothesis, which is in accordance with the expectations about the experiment from Section 1.4, is that the MAEs in Condition 2 are significantly lower than the MAEs in Condition 1.

$$\begin{aligned} H_0 &: \mu_1 = \mu_2 \\ H_1 &: \mu_1 > \mu_2 \end{aligned} \tag{4.4}$$

If the null hypothesis is rejected, a significant difference between the results from both conditions is found. If this is the case, the experiment can be deemed a success, since the Artificial Intelligence matcher proposed in this thesis has then outperformed the statistical full-content search approach. If however the null hypothesis should be retained, it must be concluded that there is no significant difference between the performances of the statistical full-content search matcher and the Artificial Intelligence matcher, at least under the circumstances as described in this thesis.

Chapter 5

Results

However beautiful the strategy,
you should occasionally look at
the results.

Sir Winston Churchill
British Prime Minister,
1940-45 and 1951-55

A software application using Artificial Intelligence techniques and methods has been proposed and built in this thesis in order to improve on statistical full-content search during matchmaking on the job market. The methods and techniques that are embedded in this application are laid out in Chapter 2, after which the architecture that combines these methods into a working application has been presented in Chapter 3. To test the performance of the devised application, an experiment has been conducted, as described in Chapter 4. This chapter presents the results obtained from this experiment.

The first section describes the results from Condition 0, the manual baseline condition. In this condition, a human expert job matcher creates the gold standard to which the two experimental conditions will be compared. The top five of one CV will be given as an illustration of the process, and the same CV will be used throughout this chapter as an example. In Section 5.2, the results of Condition 1, the statistical full-content search condition, are presented. The top fives for candidates are generated, and will be compared to the expert top fives as created in Condition 0. This will be done by calculating the Mean Absolute Error of the top fives generated by the matcher. In Section 5.3, the results of the Artificial Intelligence condition, Condition 2, will be presented. Again, top fives will be generated, and their results will be assessed by calculating their Mean Absolute Errors.

After completing all three conditions, the results from Condition 1 and 2 will need to be compared in order to determine whether a significant difference exists. First, normality will be tested using the Shapiro-Wilk Test. If normal distribution of the results is confirmed, a t-test can be utilized to determine if there is a significant difference between the outcomes of the two conditions.

5.1 Condition 0: Manual baseline

For the experiment, a data set of 10 CVs and 50 job descriptions has been selected. For each CV separately, an expert has ordered the job descriptions by relevance to the candidate described by the CV. For each of the CVs, this has yielded a top five of jobs that are most compatible with the candidate and his or her education and skills. An example of such a top five is shown in Table 5.1, where jobs are designated by their ID number, and the candidate is anonymized as 'Candidate 1'.

TABLE 5.1: The baseline top five of jobs for Candidate 1 as created by a human expert job matcher.

Candidate 1	Expert top five
1	27
2	37
3	47
4	43
5	12

5.2 Condition 1: Statistical full-content search

In Condition 1, the matchmaking process is executed by the statistical full-content search matcher. This matcher uses the complete text of the CV as a search query to be run on the database of job descriptions. Job descriptions with the highest textual similarities to the current CV will be among the top results. For each CV, a shortlist of five most similar job descriptions are returned by this matcher. The top five for Candidate 1 as generated by the statistical full-content search approach is shown in Table 5.2.

For the other nine candidates, top fives have been created in the exact same manner. With top fives for every candidate generated by the automated statistical full-content search matcher, the performance of this automated matcher can be assessed by comparing these top fives with the expert top fives that have been created in Condition 0. As shown in Table 5.3, the absolute errors of the jobs in the top five of Candidate 1 are

TABLE 5.2: The top five for Candidate 1 as created by the statistical full-content search matcher.

Candidate 1	Full-content search top five
1	24
2	34
3	37
4	33
5	25

calculated. If one of the jobs in the automated matcher's top five does not appear in the expert's top five, the absolute error is taken to be 8.

TABLE 5.3: Calculation of the absolute errors of the top five generated by the statistical full-content search matcher for Candidate 1

Candidate 1	Full-content search top five	Expert top five	Absolute error
1	24	27	not present = 8
2	34	37	not present = 8
3	37	47	$\|3-2\| = 1$
4	33	43	not present = 8
5	25	12	not present = 8

Having calculated the absolute errors for each job matched to Candidate 1 in the same fashion, the MAE is found by simply taking the mean of these numbers, as shown in Equation 5.1. Here, m is the number of predictions, in this experiment always 5, and $|e_{ij}|$ is the absolute error of prediction j for candidate i.

$$MAE_i = \frac{1}{m}\sum_{j=1}^{m}|e_{ij}| = \frac{\sum_{j=1}^{m}|e_{ij}|}{5} \tag{5.1}$$

The Mean Absolute Error for Candidate 1 is thus:

$$MAE_1 = \frac{8+8+1+8+8}{5} = \frac{33}{5} = 6.6 \tag{5.2}$$

In a similar fashion, MAEs have been calculated for the automatically generated top fives for all ten candidates. The final result of Condition 1, the list of MAEs, is presented in Table 5.4.

TABLE 5.4: The results of Condition 1: Mean Absolute Errors of job top fives created by the statistical full-content matcher.

Candidate i	1	2	3	4	5	6	7	8	9	10
MAE_i	6.6	6.6	5.0	5.6	6.4	8.0	4.6	8.0	5.0	6.6

5.3 Condition 2: Artificial Intelligence

In Condition 2, the application employing Artificial Intelligence that has been outlined in previous chapters of this thesis is responsible for the matchmaking between candidates and jobs. The application first processes the CVs and vacancies in order to extract candidate and job profiles that are stored in a relational fashion. In the second step, the application builds a complex query comprised of the values of the extracted fields, using optimized weight parameters that denote the relative importance of a match on that field. In Table 5.5, the top five as generated by the Artificial Intelligence matcher for Candidate 1 is shown.

Table 5.5: The top five for Candidate 1 as created by the Artificial Intelligence matcher.

Candidate 1	Artificial Intelligence top five
1	50
2	27
3	34
4	22
5	37

Similar to Condition 1, top fives for the other nine candidates are generated as well, following the same procedure. With these top fives, the performance of the Artificial Intelligence matcher can also be assessed by calculating the Mean Absolute Errors of the generated top fives, by comparing them to the top fives as created in Condition 0. The absolute errors of the jobs matched to Candidate 1 are calculated in Table 5.6. Once again, if a job from the automated matcher's top five does not appear in the expert's top five, the absolute error is taken to be 8.

Table 5.6: Calculation of the absolute errors of the top five generated by the Artificial Intelligence matcher for Candidate 1

Candidate 1	Artificial Intelligence top five	Expert top five	Absolute error
1	50	27	not present = 8
2	27	37	$\|2-1\| = 1$
3	34	47	not present = 8
4	22	43	not present = 8
5	37	12	$\|5-2\| = 3$

The Mean Absolute Error of the top five for each candidate can be subsequently calculated using the formula in Equation 5.1. For Candidate 1, this becomes:

$$MAE_1 = \frac{8+1+8+8+2}{5} = \frac{28}{5} = 5.6 \tag{5.3}$$

MAEs have been calculated for the other candidates as well, using the same methodology. This yields a list of MAEs, one for each candidate, which is the final result of Condition 2 and presented in Table 5.7.

TABLE 5.7: The results of Condition 1: Mean Absolute Errors of job top fives created by the Artificial Intelligence matcher.

Candidate i	1	2	3	4	5	6	7	8	9	10
MAE_i	5.6	7.0	4.6	5.6	5.6	6.0	5.0	6.6	4.6	5.6

5.4 Comparison

To answer the research question posed in Section 1.3, it is necessary to determine if there exists a significant difference between the MAE values found in Condition 1 and the values found in Condition 2. For illustration, the results from both conditions are lined up side by side in Table 5.8.

TABLE 5.8: The results of Condition 1: Mean Absolute Errors of job top fives created by the Artificial Intelligence matcher.

Candidate i	1	2	3	4	5	6	7	8	9	10
Condition 1: MAE_i	6.6	6.6	5.0	5.6	6.4	8.0	4.6	8.0	5.0	6.6
Condition 2: MAE_i	5.6	7.0	4.6	5.6	5.6	6.0	5.0	6.6	4.6	5.6

For most candidates, Condition 2 has reached a MAE that is lower than or equal to the MAE reached in Condition 1. Candidates 2 and 7 are the exceptions to this observation. To determine whether the data from both samples is normally distributed, and can thus be compared using a t-test, the Shapiro-Wilk Test [59] is utilized. The results of this test are outlined in Table 5.9.

TABLE 5.9: The Shapiro-Wilk Test indicates that the results of both conditions are normally distributed.

(A) Condition 1

Mean (μ)	6.240
Standard deviation (σ)	1.192
Sample size (n)	10
W-value	0.910
Threshold	0.781($\alpha = 0.01$)
Normally distributed	Yes

(B) Condition 2

Mean (μ)	5.620
Standard deviation (σ)	0.780
Sample size (n)	10
W-value	0.924
Threshold	0.781($\alpha = 0.01$)
Normally distributed	Yes

Now that it has been determined that the results of both conditions are normally distributed, the question is whether the results from Condition 2 are significantly lower than the results from Condition 1. To that end, a one-tailed unpaired t-test is used. The t-value is calculated as follows:

$$t = \frac{\mu_1 - \mu_2}{\sqrt{\frac{var_1}{n_1} + \frac{var_2}{n_2}}} = \frac{\mu_1 - \mu_2}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}} \tag{5.4}$$

Filling in the appropriate values in Formula 5.4 yields:

$$t = \frac{6.240 - 5.620}{\sqrt{\frac{1.192^2}{10} + \frac{0.780^2}{10}}} = \frac{0.620}{\sqrt{0.142 + 0.061}} = 1.376 \tag{5.5}$$

The alpha level for this t-test has been set at $\alpha = 0.1$. The degrees of freedom is equal to the sum of sample sizes minus two: $df = 10+10-2 = 18$. The corresponding threshold t-value is $t_{threshold} = 1.330$. Since t exceeds $t_{threshold}$, there is sufficient evidence to reject the null hypothesis $H_0 : \mu_1 = \mu 1$ at the $\alpha = 0.1$ significance level. Instead, the alternative hypothesis $H_1 : \mu 1 > \mu 2$ is accepted, indicating that the Mean Absolute Errors of Condition 2 are significantly smaller than those of Condition 1.

Chapter 6

Discussion

There are no facts, only
interpretations.

Friedrich Nietzsche
German philosopher

In this chapter, a discussion of the results of the experiment executed in light of this thesis will be presented. First, the implications of the results will be discussed. Afterwards, the limitations of the devised application and the experiment will be touched upon. Finally, recommendations for future work will be given.

6.1 Implications

The results presented in Chapter 5 indicate that there exists a significant difference between the performance of the early statistical full-content search approach to matchmaking on the job market, and the Artificial Intelligence approach as proposed in this thesis. More precisely, it has been shown that the application employing Artificial Intelligence techniques reaches significantly lower error rates than the statistical full-content search method. This means that it has been proven that the application described in this thesis reaches better results than the old statistical full-content method, the latter of which has been abandoned in the literature because of disappointing early results. These findings are in accordance with the hypothesis that was laid out in Section 1.4, indicating that this study can be deemed a success.

Although not measurable, the experiment has yielded an additional interesting result. Conform the expectations composed at the onset of this thesis, optimal solutions generated by the Evolutionary Algorithm generally assign lower relative importance parameters to fields that can beforehand be qualified as less important in the matching process. These fields include the 'name' field and, to a lesser extent, the 'date of birth' field. Higher relative importance, on the other hand, is assigned to fields that seem more important. These fields including 'role' and 'programming languages'.

At this point, it is impossible to tell whether the current approach would perform better than the rule and ontology based approach, but it does outperform the classical statistical full-content search approach. The relatively small improvement of the application laid out in this thesis over the classical approach indicate that the former may not, in fact, yield better results than the rule and ontology based method. However, although it is not reflected in the results, the lack of great restrictions on the input data is an important improvement as well. Additionally, there is quite some room for refinement of the devised application, set forth in Sections 6.2 and 6.3, which may improve the performance of this approach relative to the rule and ontology based approach.

6.2 Limitations

While proven to perform significantly better than statistical full-content search, the current approach did not yield overwhelmingly good results. This section looks at the limitations of the application as devised by this thesis, and the experiment that was conducted with it.

A limitation nearly always present in experiments that require examples to be labeled or scored manually, is that too little data is used. It is a possibility that the CVs and jobs in the used data set together are not completely representative for the average CV and job description within the domain. Related to this is the fact that only scored top fives are used, and not rankings of all jobs for each candidate. It is possible that the application would yield better results if full rankings would be provided, since positive developments would be rewarded not only if they result in suitable jobs entering the top five, but already if those would move to any position closer to the top.

A similar problem is that since running the optimization takes relatively much time, the size of the population and the number of generations had to be limited. The search space of this problem, however, is very large; seven parameters need to be optimized, all of which can take any real value between zero and ten. Usually, larger search spaces take larger populations and more generations to reach an optimal solution. To diminish this problem, parameters are limited to having only three decimals, since any more than that is unlikely to influence the result.

While better than simply counting the number of correct jobs, the Mean Absolute Error may be improved as a measure of determining performance. Since it is more crucial that the top job is returned by the application than the fifth most suitable job, it would be more appropriate if the error was calculated proportionately to the position, where mistakes at higher positions have a higher weight factor than mistakes at lower positions.

The information extraction step of the application as described, identifies seven specific characteristics of the candidate as listed on his or her CV. Although these are the most important features of the candidates, there is still more information available in most CVs. For instance, from the start and end dates of previous occupations can be deduced how much experience a candidate has in executing the tasks associated to specific roles or skills. A similar reasoning can be held for the fact that not all relevant information is extracted from job descriptions. It is therefore possible that important bits of information that include the reason why some candidates or jobs are preferred over others are still missing from the created candidate and job profiles, making the subsequent matching process more difficult than it should be.

In comparison to most other research concerning matchmaking on the labor market, the problem presented in this thesis is more difficult. Not only does it focus on using unprocessed CVs and job descriptions as input, it is also restricted to the domain of IT professionals and jobs. While a term such as 'software development' is a distinguishing term among terms taken from all professions, it is present on the CV of nearly every IT professional. It is therefore plausible that the experiment would have yielded more outspoken results if CVs and job descriptions from multiple domains had been included.

6.3 Recommendations

The process of writing this thesis and conducting the experiment described in it has lead to several recommendations for future work. These recommendations can be divided in two categories. First, recommendations about extensions of the current application are discussed. Afterwards, recommendations for the advancement of the field in relation to the current findings are suggested.

6.3.1 Extensions

Part of what makes the rule and ontology based approach good at matching candidate profiles to job profiles is the fact that it abstracts from linguistic factors such as formulation and choice of words. To mimic the latter, a sensible extension of the application devised in this thesis would be to resolve synonyms. Support for this functionality is offered by ElasticSearch, as it is able to replace all words denoting the same concept by a single one of those words (although the originals are returned as search results). A list of words and all their synonyms needs to be available for this functionality, and while such lists are available for general purposes, profession-specific lists probably do not exist.

Another potential improvement is in modifying the matcher. At this point, search for matches is unidirectional; candidate profiles are transformed into a search query, which is executed on job profiles. However, implementing bidirectional search might yield better results. In the current application, jobs are selected that are most in accordance with the skills candidates exhibit on their CV, but the suitability of the candidate for the job, in relation to other candidates, is not taken into account. Combining the results from two queries, one selecting jobs for candidates, and another selecting candidates for jobs, might give more realistic results.

6.3.2 Future work

For the advancement of the field of automated matchmaking on the job market, the next step is to compare the application devised in this thesis to current rule and ontology based matchers as described in the literature. Although it is likely that the rule and ontology based approach outperforms the Artificial Intelligence approach, it needs to be determined how much the performance of the devised application needs to improve to be able to compete with leading approaches. After all, the ease of using unprocessed documents as opposed to manually crafting candidate and job profiles may outweigh a small loss of precision in the matchmaking process.

An alternative, and potentially even better, next step would be to combine the best of two worlds. Maintain the rule and ontology based approach for matching candidate profiles and job profiles, but at the same time use information extraction in a manner as described in this thesis to create these profiles. Not only would such an approach retain all the benefits of using rules and ontologies in searching for matches, but it would also void the need to impose the aforementioned restrictions on the input data. If such an approach would prove to be viable, that would mean a great leap forward for the field, as restricting the input data can be seen as one of the most important reasons for the low adoption rates of applications that perform matchmaking on the labor market.

Chapter 7

Conclusion

A conclusion is simply the place where one got tired of thinking.

Martin Henry Fisher
American physician and author

This thesis has identified a profound lack of attention in scientific literature to using unprocessed CVs and job descriptions as input for applications that perform matchmaking on the labor market. After disappointing early results from statistical full-content matchers using unprocessed input, attention has shifted to rule and ontology based matching, requiring input documents to be preprocessed and entered into relational databases. This poses great restrictions on the input data, and has been characterized as being an escape rather than a real solution. In order to address this lack of attention, the purpose of this thesis has been to devise and build an matchmaking application that improves on the early statistical full-content approach, all the while refraining from imposing great restrictions on the input data.

In order to improve matchmaking between unprocessed CVs and job descriptions, matchmaking has been divided into two stages. During the first stage, Information Extraction methods are applied in order to identify relevant information in the documents, and store that information in a relational fashion in a database. The second stage is the matchmaking stage, wherein a complex search query is executed, with the fields of a candidate profile as constituents, and the database of job descriptions as search space. The different parts of the query are weighted, in order to make similarity on some fields more important for determining the quality of candidate-job combinations. These weights are optimized by an Evolutionary Algorithm, based on previously generated matches that receive quality scores from a user.

The research question that was posed in order to scientifically examine the problem at hand is the following:

> *Can matching between unprocessed CVs and job descriptions using statistical full content search be improved upon, by structuring these documents into fields using Information Extraction, and optimizing the relative weights of these fields using an Evolutionary Algorithm?*

As has been made abundantly clear, a significant improvement was found over statistical full-content matching using the application described in this thesis, which structures documents into fields using Information Extraction, and utilizes an Evolutionary Algorithm to optimize the relative weights. The answer to the research question is thus that matchmaking can indeed be improved by the proposed combination of techniques, as proven by the results obtained in this thesis.

An interesting suggestion for the future of the field is to attempt to conjoin the best of two approaches. First, the promising stage of Information Extraction to fill relational databases is taken from the approach described in this thesis. Next, matchmaking is executed by the rule and ontology based approaches as prominent in the literature, which require data to be preprocessed and stored in a relational database.

The results collected in light of this thesis indicate that the lack of attention to using unprocessed documents for matchmaking on the job market is unjust. The current experiment has shown that there are alternatives to imposing great restrictions on the input data, that yield promising results. This suggests that the shift of attention may have come too early or too easily, and warrants reconsidering the direction the field as a whole will venture into next.

Bibliography

[1] J. Malinowski, T. Keim, O. Wendt, and T. Weitzel. Matching people and jobs: A bilateral recommendation approach. In *Proceedings of the 39th Hawaii International Conference on System Sciences*, 2006.

[2] M. A. Huselid. The impact of human resource managment on turnover, productivity and corporate financial performance. *Academy of Management Journal*, 38(3):635–672, 1995.

[3] P. Buckley, K. Minette, D. Joy, and J. Michaels. The use of an automated employment recruiting and screening system for temporary professional employees: A case study. *Human Resource Management*, 43(2&3):233–241, 2004.

[4] A. H. Borghans. *Mobiliteit op de Nederlandse arbeidsmarkt*. 1996. ISBN 90-5321-183-7.

[5] R. W. Euwals and R. A. L. De Groot. Flexibilisering over generaties. *Economisch Statistische Berichten*, 97(4629):109, 2012.

[6] M. M. Brodsky. Labor market flexibility: a changing international perspective. *Monthly labor review*, pages 53–60, 1994.

[7] Betsey Stevenson. The internet and job search. Technical report, National Bureau of Economic Research, 2008.

[8] Eurostat (European Commission). Unemployment rate by sex and age groups, March 2014. URL `http://appsso.eurostat.ec.europa.eu/nui/show.do?dataset=une_rt_a&lang=en`.

[9] Bureau of Labor Statistics (United States Department of Labor). Unemployment rate, March 2014. URL `http://data.bls.gov/timeseries/LNS14000000`.

[10] International Labour Organisation (United Nations). Snapshot of the labour market in the european union - 2013, March 2014. URL `http://www.ilo.org/global/about-the-ilo/media-centre/issue-briefs/WCMS_209596/lang--nl/index.htm`.

[11] G. Crispin. Careerxroads 8th annual source of hire study: What happened in 2008 and what it means for 2009, March 2014. URL `http://www.careerxroads.com/news/SourcesOfHire09.pdf`.

[12] T. Weitzel, A. Eckhardt, S. Laumer, A. Von Stetten, C. Maier, and C. Weinert. Recruiting trends 2014. Centre of Human Resources Information Systems (CHRIS), Otto-Friedrich Universität Bamberg, Goethe Universität Frankfurt am Main, 2014. URL `https://www.uni-bamberg.de/fileadmin/uni/fakultaeten/wiai_lehrstuehle/isdl/RecruitingTrends_2014.pdf`.

[13] D. H. Autor. Wiring the labor market. *Journal of Economic Perspectives*, pages 25–40, 2001.

[14] C. Bizer, R. Heese, M. Mochol, R. Oldakowski, R. Tolksdorf, and R. Eckstein. The impact of semantic web technologies on job recruitment processes. In *Wirtschaftsinformatik 2005*, pages 1367–1381. Springer, 2005.

[15] J. M. Barron and J. Bishop. Extensive search, intensive search, and hiring costs: new evidence on employer hiring activity. *Economic Inquiry*, 23(3):363–382, 1985.

[16] Oxford dictionaries, April 2014. `http://www.oxforddictionaries.com/definition/english/secondment`.

[17] S. E. Jackson. The consequences of diversity in multidisciplinary work teams. In M. A. West, editor, *Handbook of work group psychology*, pages 53–75. John Wiley & Sons, Sussex, 1996.

[18] A. L. Kristof. Person-organization fit: An integrative review of its conceptualizations, measurement, and implications. *Personnel psychology*, 49(1):1–49, 1996.

[19] M. A. West. *Effective Teamwork*. BPS Books, Leicester, UK, 1994.

[20] J. Rounds, R. Dawis, and L. Lofquist. Measurement of person-environment fit and prediction of satisfaction in the theory of work adjustment. *Journal of Vocational Behavior*, 31:297–318, 1987.

[21] J. Edwards, R. Caplan, and R. Van Harrison. Person-environment fit theory: Conceptual foundations, empirical evidence, and directions for future research. In C. Cooper, editor, *Theories of organizational stress*. Oxford University Press, 1998.

[22] F. Färber, T. Weitzel, and T. Keim. An automated recommendation approach to selection in personnel recruitment. In *AMCIS*, pages 302–313. Citeseer, 2003.

[23] A. Drigas, S. Kouremenos, S. Vrettos, J. Vrettaros, and D. Kouremenos. An expert system for job matching of the unemployed. *Expert Systems with Applications*, 26 (2):217–224, 2004.

[24] B. J. Jansen, K. J. Jansen, and A. Spink. Using the web to look for work: implications for online job seeking and recruiting. *Internet Research*, 15(1):49–66, 2005.

[25] E. Herder and P. Kärger. Competence matching tool-explanations and implementation. Technical report, Open Universiteit, 2010. `http://hdl.handle.net/1820/2282`.

[26] S. Colucci, T. Di Noia, E. Di Sciascio, F. M. Donini, M. Mongiello, and M. Mottola. A formal approach to ontology-based semantic match of skills descriptions. *Journal of Universal Computer Science*, 9(12):1437–1454, 2003.

[27] G. Maniu and I. Maniu. A human resource ontology for recruitment process. *Review of General Management*, (2):12–18, 2009.

[28] M. Fazel-Zarandi and M. S. Fox. Semantic matchmaking for job recruitment: an ontology-based hybrid approach. In *Proceedings of the 3rd International Workshop on Service Matchmaking and Resource Retrieval in the Semantic Web at the 8th International Semantic Web Conference, Washington DC, USA*, 2010.

[29] L. Pizzato, T. Rej, J. Akehurst, I. Koprinska, K. Yacef, and J. Kay. Recommending people to people: the nature of reciprocal recommenders with a case study in online dating. *User Modeling and User-Adapted Interaction*, 23(5):447–488, 2013.

[30] O. Stock, C. Strapparava, and A. Nijholt, editors. *The April Fools' Day Workshop on Computational Humour*. ITC-IRST, 2002.

[31] D. Veit, J. P. Müller, M. Schneider, and B. Fiehn. Matchmaking for autonomous agents in electronic marketplaces. In *Proceedings of the fifth international conference on Autonomous agents*, pages 65–66. ACM, 2001.

[32] D. Bianchini, V. De Antonellis, and M. Melchiori. Flexible semantic-based service matchmaking and discovery. *World Wide Web Journal*, 11(2):227–251, 2008.

[33] A. Cali, D. Calvanese, S. Colucci, T.o Di Noia, and F. Donini. A logic-based approach for matching user profiles. In *Knowledge-Based Intelligent Information and Engineering Systems*, pages 187–195. Springer, 2004.

[34] Deep Nishar. 200 million members! LinkedIn Official Blog, April 2014. `http://blog.linkedin.com/2013/01/09/linkedin-200-million/`.

[35] Linkedin statistics. Socialbakers, April 2014. `http://www.socialbakers.com/linkedin-statistics/`.

[36] R. Cilibrasi and P. Vitanyi. The google similarity distance. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):370–383, 2007.

[37] H. Choi and H. Varian. Predicting the present with google trends. *Economic Record*, 88(s1):2–9, 2012.

[38] D. Jurafsky and J. Martin. *Speech And Language Processing - An introduction to Natual Language Processing, Computational Linguistics, and Speech Recognition.* Pearson Education, Upper Saddle River, NJ, second edition, 2009.

[39] R. Schank and R. Abelson. *Scripts, Plans, Goals, and Understanding: An Inquiry Into Human Knowledge Structures (Artificial Intelligence Series).* Erlbaum, Hillsdale, NJ, 1977.

[40] R. Abelson. Psychological status of the script concept. *American Psychologist*, 36 (7):715–729, July 1981.

[41] D. Roth and W. Yih. Probabilistic reasoning for entity & relation recognition. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics, 2002.

[42] L. Peshkin and A. Pfeffer. Bayesian information extraction network. IJCAI, 2003.

[43] J. Hobbs, D. Appelt, J. Bear, D. Israel, M. Kameyama, M. Stickel, and M. Tyson. FASTUS: A cascaded finite-state transducer for extracting information from natural-language text. *Finite-State Language Processing*, 1997.

[44] S. Kleene. Representation of events in nerve nets and finite automata. Rand Corporation, 1951.

[45] A. Turing. On computable numbers, with an application to the entscheidungsproblem. In *Proceedings of the London Mathematical Society*, volume 42, pages 230–265, 1936. Correction in volume 43, pages 544-546.

[46] W. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[47] J. Nothman, N. Ringland, W. Radford, T. Murphy, and J. Curran. Learning multilingual named entity recognition from wikipedia. *Artificial Intelligence*, 194:151–175, 2013.

[48] A. Mikheev, M. Moens, and C. Grover. Named entity recognition without gazetteers. In *Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics*, pages 1–8. Association for Computational Linguistics, 1999.

[49] M. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[50] M. Porter. Snowball: A language for stemming algorithms. `http://snowball.tartarus.org/texts/introduction.html`, 2001.

[51] A. Engelbrecht. *Computational Intelligence: An Introduction.* John Wiley & Sons, Hoboken, NJ, USA, 2007.

[52] C. Darwin. *On the Origin of Species by Means of Natural Selection, or Preservation of Favoured Races in the Struggle for Life.* John Murray, London, 1859.

[53] T. Bäck and H. Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, 1(1), 1993.

[54] S. Russel and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, Upper Saddle River, NJ, USA, second edition, 2003.

[55] J. Holland. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.* University of Michigan Press, 1975.

[56] K. Burjorjee. The fundamental learning problem that genetic algorithms with uniform crossover solve efficiently and repeatedly as evolution proceeds. ArXiv:1307.3824 [cs.NE], 2013.

[57] M. Kearns and U. Vazirani. *An Introduction To Computational Learning Theory.* MIT Press, Cambridge, MA, USA, 1994.

[58] A. Jacobs. The pathologies of big data. *Communications of the ACM*, 52(8), August 2009.

[59] S. Shapiro and M. Wilk. Analysis of variance test for normality (complete samples). *Biometrika*, pages 591–611, 1965.