Personalised Talent Search on LinkedIn

by

Niall Hughes, BA (Mod.)

Dissertation

Presented to the

University of Dublin, Trinity College

in fulfillment

of the requirements

for the Degree of

Master of Computer Science

University of Dublin, Trinity College

May 2017

Declaration

Permission to Lend and/or Copy

Acknowledgments

Firstly, I would like to thank Dr. Seamus Lawless and Yu Xu for all of the guidance that I have received from them over the past number of months.

In addition I would also like to thank the many lecturers from the School of Computer Science and Statistics whom I had the pleasure of learning from throughout my studies in Trinity College Dublin.

Also, I would like to acknowledge my fellow students and friends whom I have enjoyed spending the past number of years learning beside.

Finally, I would like to express my gratitude to my family, whom without their support none of this would be possible.

Niall Hughes

University of Dublin, Trinity College
May 2017

...

Dedicated to my parents, Niall and Margaret Hughes.

Personalised Talent Search on LinkedIn

Niall Hughes, M.A.
University of Dublin, Trinity College, 2017

Supervisor: Dr. Séamus Lawless

Talent management is a critical part of every organization. One of the many facets of talent management is employee recruitment and retention. It is important that organizations have the ability to identify suitable professionals to fill their vacant job positions. LinkedIn is a business- and employment-oriented social networking service that is widely used by both employees and employers. The platform provides a wealth of information regarding the skills, education and experiences of professionals in today's world. Using the mentioned information obtained from LinkedIn, this dissertation explores the use of machine learning in a quest to better classify what employees would be suitable for an organization. The machine learning techniques which are implemented are support vector machine, random forest and k-means. The three techniques were trained and tested under the same circumstances. Evaluation of the tests showed positive results especially in the case of support vector machine.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 An Introduction to Personalized Information Retrieval

Ever since us humans have had questions, we've also had information retrieval (IR). To give anything more than a broad overview of the topic of IR would be far beyond the scope of this project. IR comes in countless forms - from a person checking whats in their pocket to a scientist querying the mysteries of the universe - any quest for knowledge can be classified as a form of information retrieval. However in the field of academics, the topic can be summarized as:

"Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers)." - Manning, Raghavan, and Schütze, 2008

1.2 An Introduction to Machine Learning

According to Alpaydin, 2014 "Machine learning is programming computers to optimize a performance criterion using example data or past experience." In other words, machine learning is a sub field of computer science that aims to generate algorithms that allow computers to learn intuitively from a dataset.

1.3 Context of Use

The volume of information that is generated in the world is growing constantly. As this volume increases, so does the amount of information that is being collected and stored by various organizations. These organizations must answer many business decisions. *"Are our employees happy?" "Who needs to get a promotion?" "Who is going to quit soon?" "Who should we hire?" etc.* These questions and how an organization answers them can determine whether the organization will thrive or die.

One method that organizations can utilize to help assist them in making these business decisions is to process the information which they can gather. Through using information retrieval techniques on 'raw' data they can infer higher order data that in turn can help with the decision making process. A simple example of how an organization could use this process in a real world example could be to record the number of keystrokes, a form of 'raw' data, an employees performs on a computer. A declining number of keystrokes by the employee could correlate to a decrease in work productivity. This scenario could infer that the employee is possibly unhappy in their job or that they are not qualified to perform work they are undertaking.

The increasing popularity of data analytics can be seen in Figure **??** below. Across the three recruitment websites ItJobsWatch, LinkedIn and Indeed.com we can see that there has been a stark increase in the number of data scientists using these websites in recent times. This is no doubt due to organizations attempting to exploit the many capabilities of data analytics.

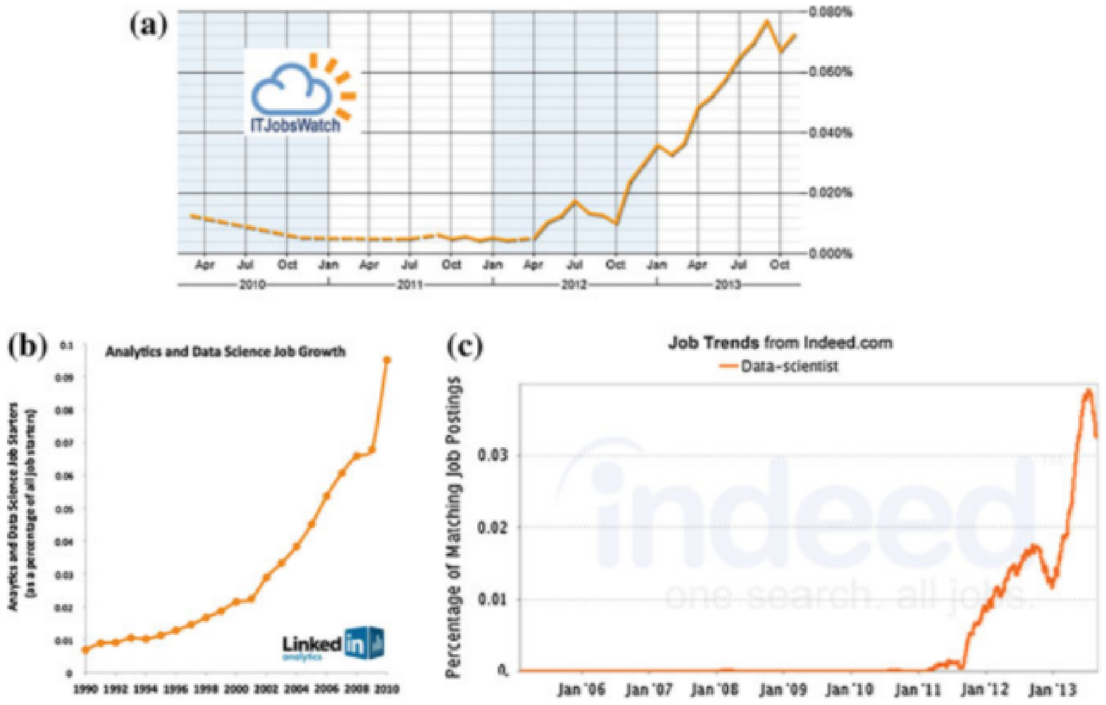


FIGURE 1.1: Data Scientist Job Trend

The scenario of converting raw data to higher order data that I will be focusing this paper on is the one of helping employers find suitable employees. The strength of an organization is strongly linked to the employees who work there. Thus, the recruitment process of an organization has a significant impact on it's strength. This results in organizations placing a lot of emphasis on creating a robust hiring procedure. This recruitment process is part of a larger field of human resource work known as *talent management*.

By having an efficient recruitment system, a company can ensure that it has a better chance of hiring the appropriate people for their job positions. It can also result in the organization utilizing its resources more efficiently in the search for talent i.e. lower expenditure spent on recruitment. Due to these stated reasons, it is quite clear that their is a constant need to continually improve the procedure of hiring new talent. This is what has motivated this project.

1.4 Research Question

The research project was put forward by my supervisor Dr. Seamus Lawless. From meeting with him and discussing what objectives the project would be setting out to reach, a research question was formulated. When formulating this research question inspiration was taken from the motivation behind the project and also taken from a number of current approaches o the topic. From analyzing these, I arrived at the following research question:

Can I use machine learning to develop an accurate personalized talent search using information from LinkedIn user profiles?

1.5 Objectives

The main goal of this project was to develop a method of identifying suitable employees for companies to hire. This would involve working closely with raw data that has been scraped from the public profile of LinkedIn users in order to build a system that would be capable of classifying users into one of two groups; suitable or unsuitable candidates for employment by that company. In order to achieve this functionality I would utilize the benefits of machine learning in order to generate a model that would be capable of making this recruitment decision. Here are the milestones that I needed to complete in order to answer the research question:

1. **Parse raw information from Linked In profiles**
   This initial step is crucial to the success of the project. For it, the aim is to extract as much information about a user as possible. Failure to parse a sufficient amount of information could lead to the system being unable to accurately differentiate between what users would be suitable for a job and what ones wouldn't. The more information that is available to the system, the higher likelihood there is that the system will produce useful findings.

2. **Create various machine learning models using this raw data**
   When the raw information has been parsed, the next step is to use this data to build a number of machine learning models and evaluate which ones work best at classifying users. I decided to implement three separate machine learning techniques, one unsupervised technique and two supervised techniques. The decision to build multiple machine learning models was taken as it would make it possible to generate a wider variety of results.

3. **Use the machine learning models to classify what users would be suitable employees for a given company**
   Using the models created in the previous objective, users will be classified into whether they are suitable candidates for a job or not. A wide selection of users, job types and companies will be tested and evaluated during this project. The results of these classifications will be measured and accuracy scores will be derived for all of the various machine learning models to evaluate what method is the most successful at the process.

1.6 Report Structure Outline

Chapter 2 - State of the Art

In this chapter I will give an overview of some of the existing research in the field of talent management specifically related to employee recruitment. Broadly speaking, the content of the literature that was researched can be grouped into either technical or non-technical approaches.

Chapter 3 - Design & Methodology

This chapter will describe the steps I undertook in order to implement the system that I built. I will discuss in great detail the different methods I used and why they were chosen. In general, these methods will be presented in the paper in the order that the work was done chronologically.

Chapter 4 - Results

Here I will detail the various statistics that I generated from training and testing the machine learning algorithms. I will use a number of different metrics to reveal what the different machine learning algorithms are good for. These metrics will be described at the beginning of the chapter. I will also discuss the different findings that my research provided. I will speak about what factors influenced the results and why they influenced them. I will make judgments on whether the project has been a success, how the machine learning algorithms performed and how the results could possibly be improved.

Chapter 5 - Conclusion

In the final chapter of this paper I will provide my thoughts on how I feel the project has turned out. I will evaluate whether I was able to successfully answer the research question that I detailed in this introductory chapter. In addition I will also speak about the possible future work that could be undertaken by researchers in this field.

Chapter 2

State of the Art

2.1 Introduction

Talent is considered as the capability of any individual to make a significant difference to the current and future performance of the organization Lynne, 2005. The term '*talent management*' was coined by a group of Mckinsey associates in the late 1990's, Michaels, Handfield-Jones, and Axelrod, 2001. There is some ambiguity about what exactly is encompassed in the realm of talent management. Ashton and Morton, 2005 stated that there *"isn't a single consistent or concise definition"* to summarize the topic. Despite this ambiguity, it is broadly recognized that the term talent management represents the field of science that encompasses everything that involves recruiting, retaining, developing, rewarding and encouraging employees. Of these, this research project is primarily concerned with the recruitment of talent. This chapter will give an overview of some of the current approaches that are present in the realm of talent management being employed by organizations in order to recruit what they deem to be suitable employees. To provide a general overview of the industry that is talent recruitment, the current approaches that have been examined range from abstract non-technical methods to extremely technical solutions that are quiet similar to this very project. During this research of talent management, and specifically of talent recruitment, I discovered that there is a lack of technical solutions to the problems faced. Although there is a vast amount of literature available on talent management, this literature is mainly limited to the non-technical side of the process. These examples will be presented in this chapter in an order beginning with non technical approaches and progressing to discuss some of the technical solutions in existence today.

2.2 Non-technical Talent Management Solutions

As explained, this section I will outline some of the non technical literature that has been written on the topic of talent management. This literature that I have reviewed discusses various approaches to the field. These encompass both theoretical and existing real-world solutions.

2.2.1 A Theoretical Model Of Strategic Talent Management

A theoretical model of strategic talent management was proposed by Collings and Mellahi, 2009. In this model, the authors separate the process of talent management into two steps - identifying pivotal talent positions and creating a talent pool of professionals which can be used to fill these pivotal positions.

Step One - Identifying Pivotal Talent Positions

Step one involves identifying "pivotal talent" positions. A pivotal talent position can be identified through a number of methods as mentioned by Collings ad Melhali. According to Becker and Huselid, 2006, as referenced in the the paper, one of which could be “When employees are able to contribute to a firm’s strategic objectives they have (strategic) value” thus deeming their position to be a pivotal one. Another method of identifying a pivotal talent position which is considered by the paper in question is one that was presented by Huselid, Beatty, and Becker, 2005, “disproportionate importance to a company's ability to execute some parts of its strategy and second. . . the wide variability in the quality of the work displayed among the employees in these positions". It is also noted that although a job position may have a significant strategic role in the company, regulation and training may result in the performance of different employees undertaking that role being standardized i.e the role may be strategic but it is not deemed as pivotal by the talent management model. With this in mind, roles that have the potential for greater differentiation between the performance of separate employees undertaking that role must be given a greater importance in the talent management system.

Step Two - Developing a Talent Pool

The term talent pool is described as "the pool of high potential and high performing incumbents that the organisation can draw upon to fill pivotal talent positions". This technique is summarized as "recruiting ahead of the curve (Sparrow, 2007). In order for this talent management model to succeed, the talent pool must be pro actively populated with suitable talent. This is so employees can be recruited to pivotal positions as the need for them arises. This process is likened to how professional sports teams scout for talent.

Combining the Two

When a model has been created that includes both the identified pivotal talent positions and the developed talent pool then it theoretically should have the ability to efficiently and successfully recruit suitable employees to the important roles within an organization.

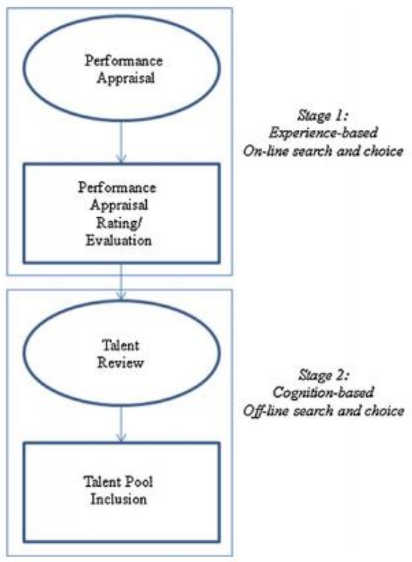
I can see how my thesis project could assist during this two stage process. An accurate machine learning model can be used to classify suitable employees into the talent pool that is generated in the first step of this process.

2.2.2 How Multinational Corporations Establish Internal Talent Pools

A paper by Mäkelä, Björkman, and Ehrnrooth, 2010 discusses the manner in which multinational corporations (MNCs) identify and generate internal talent pools. 'Internal talent' is defined as talented employees who already work for the MNC. The paper aims to understand the decision processes involved in the identification of this MNC-internal talent. In the paper Björkman and Ehrnrooth build a framework that describes this process of internal talent identification as being a two stage decision process. This process can be broadly separated into (1) performance based appraisal evaluations and (2) cognitive managerial decision making.

The first of these two steps, i.e. performance appraisal, is primarily influenced by annual and bi-annual performance reviews of employees by their respective direct superior manager. Occasionally, this review process can be supplemented with additional reviews by an employee's peers and/or subordinates.

FIGURE 2.1: Björkman and Ehrnrooth's Two Stage Internal Talent Identification Process (Mäkelä, Björkman, and Ehrnrooth, 2010)


Performance Appraisal
Performance Appraisal Rating/ Evaluation
Stage 1: Experience-based On-line search and choice
Talent Review
Talent Pool Inclusion
Stage 2: Cognition-based Off-line search and choice

The second step of the decision process, i.e. managerial decision making, is where the talent pool in constructed via exploiting the data collected in the first step. The research presented here shows that there are a number of factors that can negatively impact the ability of the manager to accurately classify employees into the internal talent pool at this step. In general, MNCs aim to implement corporate wide standardized methods for measuring performance however in reality the methods that are implemented often differ slightly to the ones that were intended (Nishii, Lepak, and Schneider, 2008). In addition, rater bias can impact the result of an appraisal. For example, this bias can be as a result of gender or race.

In this process there is once again an opportunity to utilize the system created from my project. This time it comes during the concluding phase as the managerial decision making can be replaced with machine learning classification to identify the suitable employees.

2.2.3 Talent Management for 21st Century

The book Talent Management for the Twenty-first Century (Cappelli, 2008) outlines the supply chain model for talent management. As the name suggests, the supply chain model deals with the flow of talent through a company. According to Capelli, the process relies on four principles regarding both risk and uncertainty. These four principles are:

1. Make and buy to manage risk. Having more talent in a company than is needed is expensive. Companies should undershoot the estimates of their talent needs. If a company experiences a shortage of talent then it is at this point where they will hire from outside the company.

2. Adapt to the uncertainty in talent demand.By its nature, demand is an uncertain thing. Companies should strive to ensure that they are capable of managing the varying levels of talent demand that can be placed upon themselves. There are a number of suggested methods of achieved this, for example by developing an organization wide talent pool that can be allocated amongst different sectors of the business depending on demand.

3. Improve the return on investment in developing employees. This might include asking employees to take on additional assignments as part of their work. Another method of achieving this would be by maintaining a relationship with an employee following their departure from an organization in the hopes that they may return someday, bringing back the investment that was made in their skills.

4. Preserve the investment by balancing employer-employee interests. The primary

reason employees leave a company is due to them finding better opportunities elsewhere. To preserve the investment in employees it is important that they have a share in advancement decisions.

A company that adheres to these principles successfully, place themselves in an advantageous position to exploit their investments in employees. Again, there is a need to identify suitable employees and due to this the machine learning classification model from my thesis project can be used to automate and possibly improve the process.

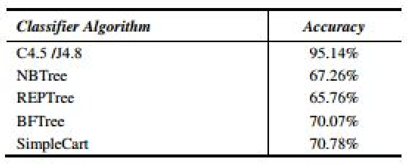
2.3 Technical Talent Management Solutions

Here, I will discuss examples of literature where technical solutions were created in order to classify talent.

2.3.1 Classification for Talent Management Using Decision Tree Induction Techniques

A study was conducted by Jantan, Hamdan, and Othman, 2009 regarding data mining and how it can be used in conjunction with decision tree machine learning technologies to classify talent in a dataset. It found that this method of classification can result in extremely accurate predictions regarding the dataset when a correct technique is applied. In this case, the C4.5 classification technique proved to be by far the most successful compared to its rivals in this research see ??. This experiment has shown the potential of using decision trees as a successful solution for data mining classification scenarios.

FIGURE 2.2: Accuracy results from the study Jantan, Hamdan, and Othman, 2009

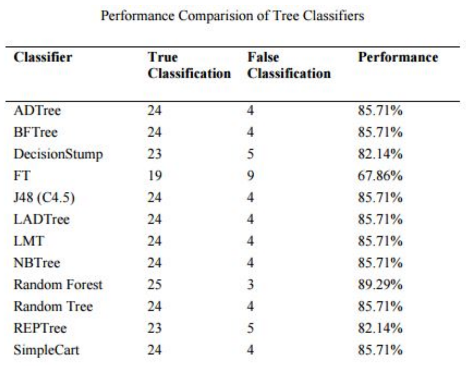


Classifier Algorithm	*Accuracy*
C4.5 /J4.8	95.14%
NBTree	67.26%
REPTree	65.76%
BFTree	70.07%
SimpleCart	70.78%

Jantan's paper also suggests that there is a need to further investigate this realm of science in an effort to "broaden our horizon of academic and practice work on Data mining in HR". He suggests that other techniques, namely Support Vector Machine (SVM) and Artificial Immune System (AIS), should be investigated during future work in this scenario.

Another example of research proving this to be true was conducted by Özsoy, Gümüş, and KHALILOV, 2015. In their paper they also contrasted the use of C4.5 against its compatriot decision tree based machine learning techniques. Once again, C4.5 achieved very successful results however even these results were out eclipsed by the machine learning technique Random Forest (see figure **??**). Each of the algorithms shown in this table were all trained and tested using the same data which was comprised of multiple features. This has shown me that using a decision tree based machine learning technique could be a possible solution for classifying the Linked In dataset, as it is also a dataset which contains items with multiple features. In addition, it has shown me that of all the decision tree based techniques, Random Forest is the most likely technique to achieve successful results.

FIGURE 2.3: Accuracy results from the study Özsoy, Gümüş, and KHALILOV, 2015



Performance Comparision of Tree Classifiers

Classifier	True Classification	False Classification	Performance
ADTree	24	4	85.71%
BFTree	24	4	85.71%
DecisionStump	23	5	82.14%
FT	19	9	67.86%
J48 (C4.5)	24	4	85.71%
LADTree	24	4	85.71%
LMT	24	4	85.71%
NBTree	24	4	85.71%
Random Forest	25	3	89.29%
Random Tree	24	4	85.71%
REPTree	23	5	82.14%
SimpleCart	24	4	85.71%

2.3.2 Applying Support Vector Machine Algorithm in Employee Achievement Classification

The paper "Towards Applying Support Vector Machine Algorithm in Employee Achievement Classification" explores the use of support vector machine as a means of classifying which employees would be suitable for a given task based on their previous performance achievements. This is an extremely similar use case to the one that I have with my dissertation project. The context behind the paper is also similar to my own as both papers aim to investigate using machine learning in order to make companies more competitive.

Like all classification experiments, it was set up in two phases - training and testing. The training phase analyzed the training data to construct a model and the testing phase evaluated the constructed model through using a separate testing set of data.

As can be seen from the results in figure 2.4, the accuracy scores achieved by the experiment were quite erratic. However, the model trained using 80% of the dataset achieved an accuracy score of 95% which was very encouraging. According to the paper, the average accuracy score of 56% across all ten models is "slightly low".

Despite this, the authors of the paper reach the conclusion that the accuracy of the generated model is acceptable. The paper does suggest that future work on the topic should strive to improve the accuracy figures of the model.

FIGURE 2.4: Accuracy results from the study “Towards Applying Support Vector Machine Algorithm in Employee Achievement Classification”

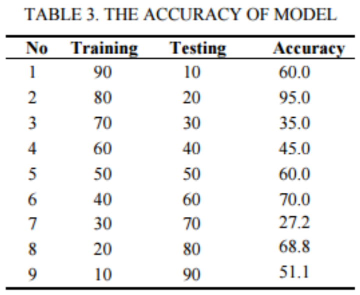


TABLE 3. THE ACCURACY OF MODEL

No	Training	Testing	Accuracy
1	90	10	60.0
2	80	20	95.0
3	70	30	35.0
4	60	40	45.0
5	50	50	60.0
6	40	60	70.0
7	30	70	27.2
8	20	80	68.8
9	10	90	51.1

2.3.3 Leveraging the LinkedIn Social Network Data for Extracting Content-based User Profiles

An interesting paper was published by Lops et al., 2011 where the authors had a very similar use case to the one that I dealt with in undertaking this research proposal question. Similarly to my project, this paper in question had the objective of extracting useful features from Linked In user profiles with the intension of using the extracted information in order to classify users. Where objectives of this paper differed to my own was in the purpose of why the users were being classified. The authors built a system that would use Linked In information to generate appropriate recommendations of research papers which would be of interest to particular users. This is in contrast where I built a system that had the end goal of recommending appropriate users to companies based on the available LinkedIn information.

In order to build the system described in Lops' paper, the authors developed a system known as the 'LinkedIn Extractor' (see figure 2.5). This 'LinkedIn Extractor' operates extremely similarly to how the system I built as part of my project operates. Both systems use user profiles as input, extract relevant information to create profiles that represent users and then use these profiles to provide recommendations as the output.

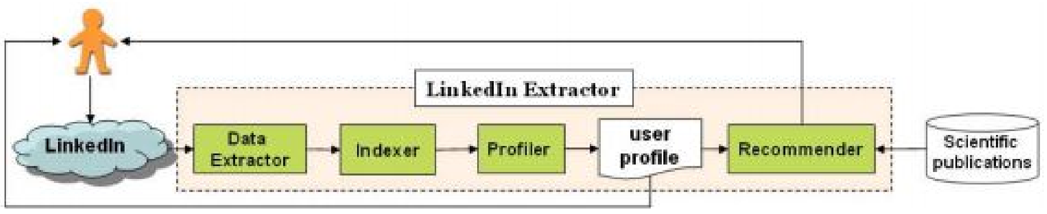
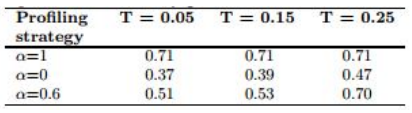


FIGURE 2.5: Overview of Lops' 'LinkedIn Extractor'

One advantage that Lops' data had over my own was that he also had access to the social graph information of users, i.e. who they were connected to. This gave the ability to use feature extraction to derive additional features via exploiting the information associated with the connections of a user.

Figure 2.6 shows the results that Lops' paper achieved. There were two variables that were modified to help generate results in order to evaluate what methods worked best. ***T***, represents the similarity threshold for selecting connections to contribute information to the recommendation process. Only connections with a similarity score greater than the threshold will contribute information to the process. In general a high ***T*** value would result in a lower number of connections being included and a low ***T*** score would result in the opposite. The second parameter is α, and it represents the threshold for using the data from the connections that satisfy the similarity threshold. A value of $\alpha = 1$ would mean

FIGURE 2.6: Accuracy results from the study Lops et al., 2011



Profiling strategy	T = 0.05	T = 0.15	T = 0.25
α=1	0.71	0.71	0.71
α=0	0.37	0.39	0.47
α=0.6	0.51	0.53	0.70

that the recommendation process relies entirely on the user's own profile features. A value of $\alpha = 0$ would mean that the recommendation process relies entirely on the connections of a user to generate its recommendations. A score of $0 < \alpha < 1$ means that the recommendation process includes an appropriate proportional amount of both the users own profile features and the features from its suitable connections.

This results presented in figure 2.6 shows that accurate results can be obtained when exclusively using only the users own profile information ($\alpha = 0$) or by only including information from a users' connection only if the connection has a high similarity score. Using only the information from connections ($\alpha = 1$) results in a poor performance. The fact that the use of connections did not improve the performance of the system suggests that the results obtained from my project will not suffer as a result of not including connections in the generation of a model.

What I Learned

From researching the topic of talent management I have discovered a number of ideas that have been put forward as solutions to the problem of talent identification and recruitment. Non technical solutions revolve around identifying potential employees and adding them to '*talent pools*. These *talent pools* can be exploited on an ad-hoc basis by making managerial judgments regarding who who would a suitable professional for a given role. A machine learning classification process could replace a lot the manual work of these non-technical solutions.

In regards to the technical solutions that I researched, I also discovered a number of interesting points. Of the decision tree techniques tested in Jantan, Hamdan, and Othman, 2009, random forest was the most successful. "Towards Applying Support Vector Machine Algorithm in Employee Achievement Classification" showed that SVM is a capable solution for machine learning classification. Also, thanks to the results from Lops et al., 2011, I know that not having access to the LinkedIn social graph information is not a big concern.

Chapter 3

Design and Methodology

3.1 Introduction

As mentioned in the introductory chapter, this project requires interacting with a large amount of raw data. A number of technologies were utilized in order to perform the various experiments required to achieve the meaningful results that were targeted at the beginning of the project. In this chapter I will outline the methods which I employed to build my system.

These methods will be summarized in the following sections:

- Use cases. I will discuss what scenarios my project tested. These scenarios encompassed a number of use cases.

- The data. This section comprises of the information regarding the data which the project operated on. The original form that the data was in as well as the necessary changes that I performed in order to translate the raw information into a manner which could be more easily worked with.

- Training and Testing. This will introduce the procedures which I followed to create the training a testing data that I used as part of the result generation process. I will speak about how I selected what data would be suitable and also how I structured this data.

- Machine Learning. This section will give an overview of machine learning and the two genres, supervised and unsupervised, that I implemented as part of the project. It will also go into more detail about the specific machine learning techniques within these two genres that I used, i.e. k-means, random forest and support vector machine.

- The Linked In Search User Interface. This section will summarize how I developed the *Linked In Search User Interface* that I used to generate and visualize my results. It will provide information on the steps that I undertook to build this system. It will also give an overview of the various software technologies that I incorporated into this system. Their purposes and how they operate in general and also how they operate specifically in my system. I will also speak about why I chose them instead of their counterparts and also any problems which I encountered with using them. Also, I will go into detail about the Lucene search engine and how I implemented it into my project to help better evaluate how accurate my results are.

- Generating Results. I will speak about the procedure that I followed to generate the results that this paper presents.

3.2 Use Cases

As part of this project, I aimed to test a number of different use cases to see how the system I build performs. I selected these use cases as I felt they either represented scenarios that could exist in the search for talent, would be a good method of evaluating the system or I was interested in seeing how the system would respond to their parameters.

The use cases which I decided to test were:

- Classifying by job role.
- Classifying by job and company pairs.

Each use case represents a high level of abstraction from the actual data that is being used to test it. This means that a single use case can be tested using different sets of data i.e. the '*classifying by job role*' use case can be tested using different sets of data comprised of employees with different job roles. An example of this would be testing this use case using the job role '*Software Engineer*' and then re-testing using the job role '*Lecturer*'.

3.3 The Data

The data that I was presented with at the beginning of this project came in the form of approximately 300,000 raw HTML files. These files were scraped from the Linked In website a number of months ago by a PHD student who worked closely with my supervisor, Dr Seamus Lawless, and who was a member of Trinity's SCSS department. These HTML files acted as the the basis thesis project.

The 300,000 raw text files were unintelligible in the original form that they were in. In order to perform efficient queries on the data which is contained in them it is necessary to structure the information within the files in a more suitable manner.

3.3.1 Creating a Data Type to Represent a Person

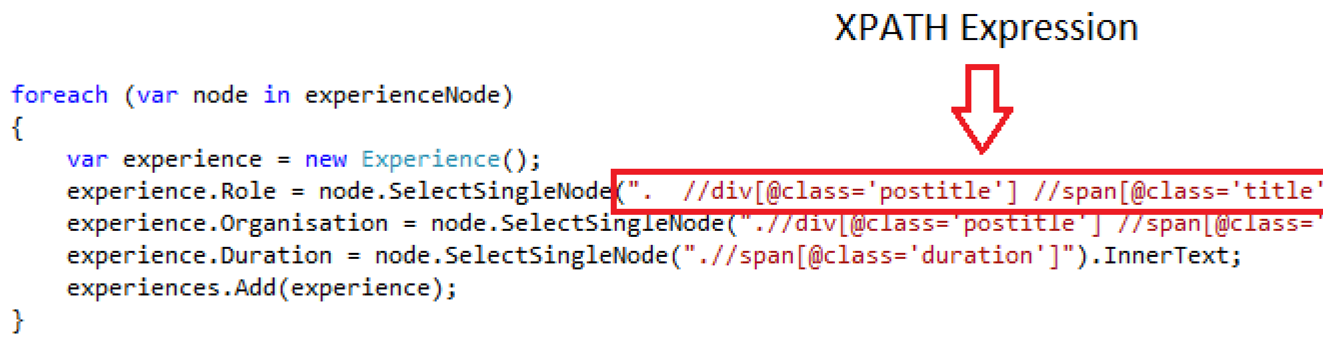
The first step that I took to achieve this was to parse each file in order to index it into a data type that will represent the critical information from a profile. This new *Person* data type was important for the system which I would build. It would allow me to interact with the information in an easier and faster manner.

In order to do this I needed to first investigate the raw HTML files to discover what meta data about the person that they contained. I decided that information such as education, job experience, skills etc. would be relevant to my search.

3.3.2 Parsing the Raw Files

Following this first step, I then developed a parser that would extract the relevant information from the files into the data type for a person. Similarly to the other development work I undertook as part of this project, I used the C# programming language to build the software. In addition, I also made use of HTML Agility Pack which is an open source .NET library.

HTML Agility Pack is a robust library for parsing HTML files. It makes use of XPATH. XPATH is is a syntax for defining parts of an XML document and allowed me to query the raw HTML documents using *'path expressions'* to navigate the documents. Through using these *'path expressions'* I was capable of selecting the desired node from the document.



```
foreach (var node in experienceNode)
{
    var experience = new Experience();
    experience.Role = node.SelectSingleNode(".  //div[@class='postitle'] //span[@class='title'
    experience.Organisation = node.SelectSingleNode(".//div[@class='postitle'] //span[@class='
    experience.Duration = node.SelectSingleNode(".//span[@class='duration']").InnerText;
    experiences.Add(experience);
}
```

FIGURE 3.1: Code snippet from the parser showing an example of XPATH in use.

3.3.3 Storing the Information

Parsing the information from the raw files was a complicated and time consuming task. Due to this I did not want to have to perform this resource intensive operation every time I needed to query the information. To resolve this problem, I used XML serialization to store the parsed information into a large XML file(see 3.2. Parsing the information from this one structured file was orders of magnitude faster than parsing the approximately 300,000 raw HTML files.

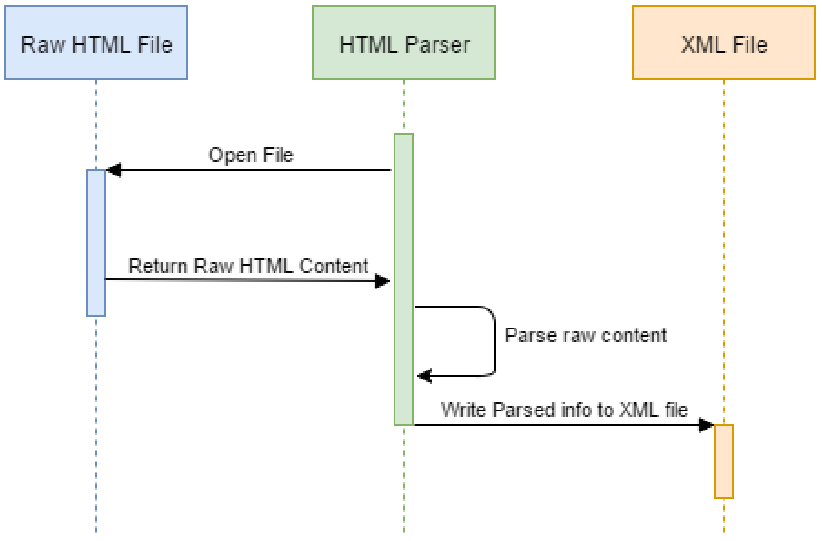
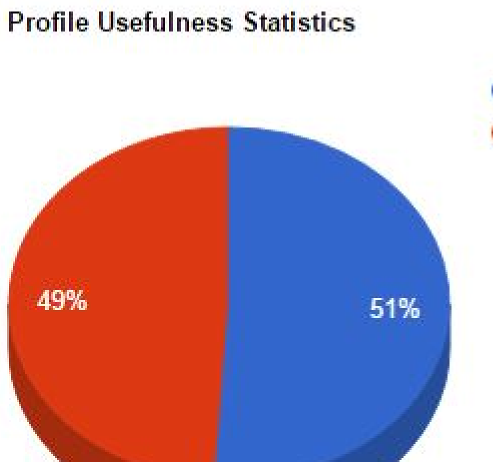


FIGURE 3.2: A Sequence Diagram Depicting how the raw HTML files are parsed.

Removing 'Unuseful' Data

When analyzing the parsed data from the raw HTML files I quickly discovered that a sizable number of the profiles did not contain information that would be useful to my project. These profiles had extremely limited meta data about the owner of the profile. I decided to remove any Linked In profiles that did not provide what I deemed to be adequate information. I reached

FIGURE 3.3: Profile usefulness breakdown



this decision as I felt that these 'unuseful' profiles did not offer any information that could help my project. I decided to classify each profile into one of two groups - 'useful' and 'unuseful'. I determined that a useful profile is one that contained work experience for the user and an 'unuseful' profile would be one that does not. Using this classification scheme, I discovered that 51% of profiles were 'uselful' and 49% were 'unuseful'.

3.3.4 Ethical Considerations Regarding the Information

In today's world, information privacy is becoming a more and more prevalent and important subject. With this in mind, I persevered to treat the information that I used throughout the project with care and integrity. In order to achieve this a number of considerations were made.

1. Public information. My project only utilizes information that is available on the public profile of a user. This information is freely accessible to anyone connected to the Internet.

2. Data anonymization. As part of indexing the information, the users name was removed from the profile.

3. Representing users as numerical feature vectors. During the experimentation process, users were translated into numerical feature vectors to be represented. This process further anonymized the information being used.

3.4 Creating the Training and Testing Sets

Using training and testing sets is a common practice in machine learning and information retrieval to create and evaluate retrieval methods. In short, the training set comprises of a group of data that will be used to 'train' the IR technique. The testing set comprises of a set of data which holds similar but differing data to that in the training set.

Job Role Only

Due to the primary goal of my project being creating a better method of identifying what employees would be suitable for a certain job - the training and testing sets needed to be constructed in a manner that would make the evaluating of this goal possible. The first step of this would be to identify suitable job roles to base the creation of the training and test sets on. To do this I generated a list of the most frequently occurring job roles throughout the dataset (see figure 3.4).

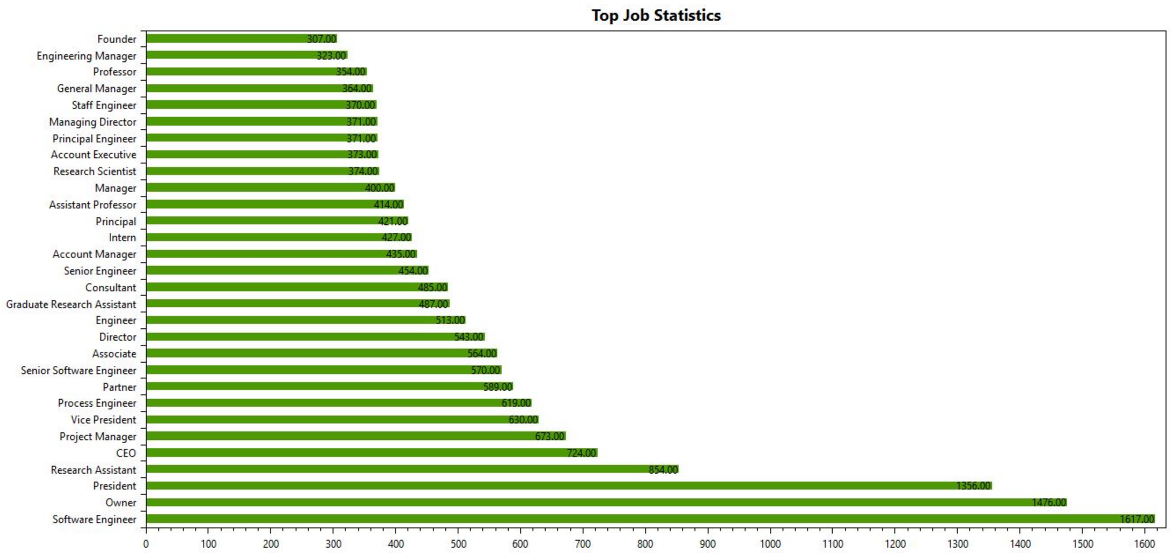


FIGURE 3.4: The top 30 Most Frequent Jobs

I decided to use the most frequently occurring job role to create my initial testing and training sets before repeating the process with less popular job roles. As can be seen from figure 3.4, the most popular job in the dataset is Software Developer with 1,616 people with that profession. Taking these 1,616 employees, I separating them evenly into two groups of 808 employees. I then added an additional 808 employees with different professions who were chosen at random from the dataset. Following this step I now had created my training and testing sets, each of which comprised of 50% software developers and 50% non-software developers. Figure 3.5 shows how this process is performed.

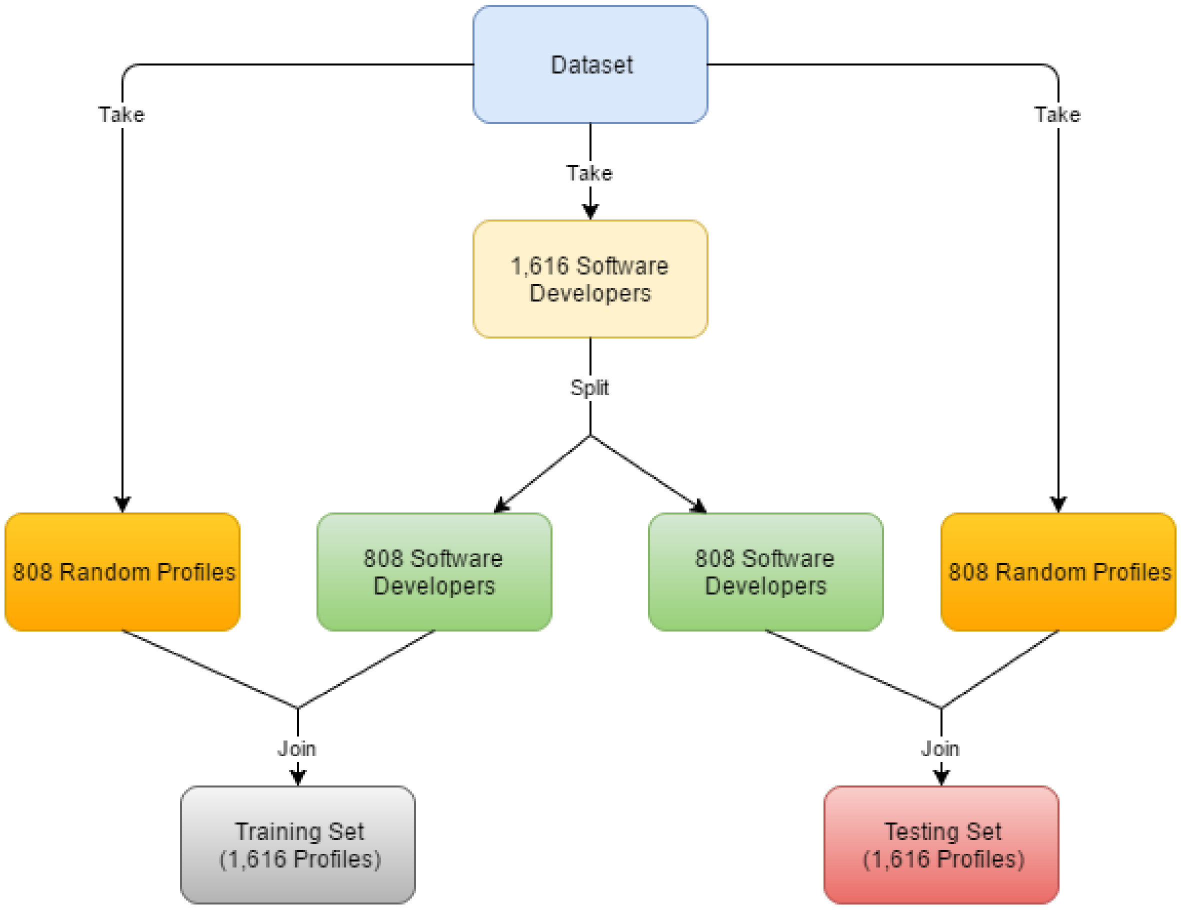


FIGURE 3.5: Workflow of how the Software Developer Training and Testing Sets were Created

As mentioned, this process of creating training and testing sets is repeated for various different job roles in order to have the ability to generate more results for evaluation. Other job roles shown in figure 3.4 were used to create these and the corresponding results of doing so can be seen in the results chapter of this paper.

Job Role and Company

In addition, my project required that I also created these training and testing sets for job role and company pairings. Once again I generated statistics to discover what are the most frequently occurring job and company pairings in the dataset and with this information I was able to build what I deemed to be the most appropriate sets to generate meaningful results.

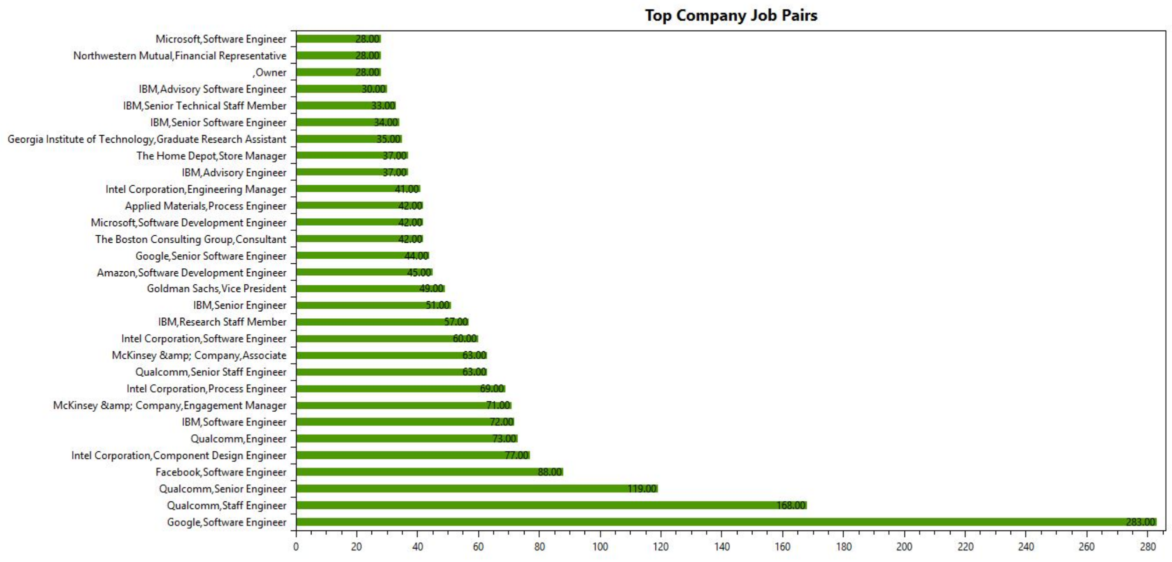


Figure 3.6: Most frequently occurring job role and company pairings

As can be seen by the above figure 3.6 'Google, Software Engineer' is the most frequently occurring pairing. For the same reasons as mentioned before, I decided that this pairing would be a suitable candidate for the creation of the sets. I then followed the procedure of selected other frequently occurring pairings to generate additional results with the intension of evaluating them too to better answer my proposal question.

3.5 Machine Learning

3.5.1 Introduction

Machine Learning is a method of data analytics that aims to allow a computer to learn and make predictions regarding data based on an algorithm. Broadly speaking there are three types of machine learning:

- Unsupervised. The machine learning algorithm tries to create a model purely based on the input. It looks for structure and patterns in the data.
- Supervised. The algorithm is presented with both input and the desired output. The algorithm aims to create a model that will best match the given inputs to the outputs. The goal of supervised machine learning is to build a concise model of the distribution of class labels in terms of predictor features Kotsiantis, Zaharakis, and Pintelas, 2007.
- Reinforcement. The algorithm is presented with a dynamic environment and a desired goal. It must learn which are the optimal actions it must undertake in order to achieve the target goal based on whatever state it is in.

As part of my project, I implemented and contrasted three separate machine learning algorithms to find which is the most suitable for my use case. I chose to implement one unsupervised algorithm and two supervised algorithms.

3.5.2 Unsupervised Machine Learning

K-means

Due to the popularity of the k-means machine learning algorithm, I decided that it would be interesting to investigate whether it could be a possible solution to my problem. K-means can be used to classify a given data set into a certain number of clusters, two in my case. These clusters are placed at different locations. Each item in the data set is then assigned to the nearest cluster based on the Euclidean distance between that item and the cluster. When all of the items have been assigned to a cluster, then the algorithm re-calculates the new centroids of the clusters resulting from the items assigned to that cluster in the previous step. Each item in the data set is again assigned to a cluster based on the Euclidean distance between the item and the new updated centroids. This process is repeated until there is no changes to the locations of the clusters centroids.

3.5.3 Supervised Machine Learning

As mentioned above, supervised machine learning uses both input and the desired output. It will attempt to generate a model that best matches the provided input to the output. This suits my use case as I wish to train a model to recognize a certain job position from the features of a LinkedIn user profile. I have the ability to supply the machine learning algorithm with the input, i.e. all of the features of a LinkedIn profile, as well as the desired output, i.e. the job position of that profile. In general, SVM tends to perform better than its machine learning counterparts when operating on multi-dimensions and continuous features whereas logic-based system (e.g. decision trees) tend to perform better when dealing with categorical and discrete features Kotsiantis, Zaharakis, and Pintelas, 2007. With this in mind, I implemented and tested how two separate machine learning algorithms performed against the data set - support vector machine and random forest.

Support Vector Machine (SVM)

Support vector machine (SVM) is a supervised machine learning algorithm which can be used for both classification and regression problems. For my project, I will be using it to solve a classification problem i.e. classifying a LinkedIn user profile into a profession. The algorithm works by plotting each item in the data set as a point in an n-dimensional space, where n is the number of features of the profile that are being used. The SVM algorithm then finds the hyperplane that best splits the two clusters. The SVM algorithm can make use of different kernels to achieve more accurate classifications. As you would expect, the linear kernel uses a linear function to differentiate between the classifications. The polynomial kernel uses a polynomial function which can in some cases provide more accurate results. SVM has the benefit of being effective in high dimensional spaces such as the one associated with my project. However, It doesn't perform well when the data set has more noise i.e. target classes are overlapping which can be the case with my dataset.

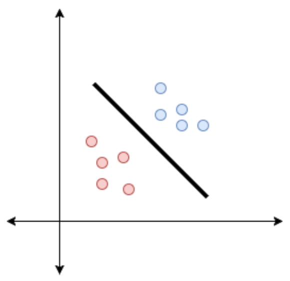


FIGURE 3.7: SVM Linear Kernel Classification Example

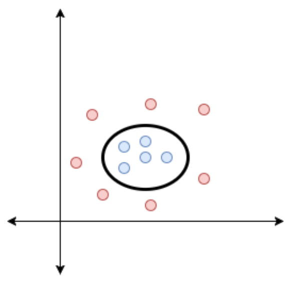


FIGURE 3.8: SVM Polynomial Kernel Classification Example

Random Forest

Random forest is an ensemble machine learning technique used to perform classification. Ensembles combine multiple hypotheses to form a (hopefully) better hypothesis. The main principle behind ensemble methods is that a group of 'weak learners' can come together to form a 'strong learner'. Random forest uses a multitude of decision trees (weak learners) to perform its classification. A decision tree is a predictive model that uses observations about an object to derive conclusions about its target value, i.e. the profession of a LinkedIn user profile. These individual decision trees are trained by each being given a bootstrapped version of the original training data and the labeled expected output. This bootstrapped version ensures that each decision tree is unique. The random forest algorithm takes the mode output prediction value across all of its decision trees based on the given input. This 'voting' system is designed so that the error cases from individual trees will not effect the mode decision of the forest. The process of growing each decision tree in the forest is:

1. Sample N cases at random with replacement, where N is the number of samples in the original data. This set of cases will be the training set for growing that individual tree.

2. If there are M input variables (features of the LinkedIn user profile), a number mM is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the forest growing.

3. Each tree is grown to the largest extent possible. There is no pruning.

Pruning is the technique of reducing the size of a decision tree by removing sections that do not have great importance in classifying input. This is done to reduce over fitting of the tree to the training data. Random forests too have the ability to over fit the data, Segal, 2004 showed this by demonstrating that modifying the depth of trees in a random forest can result in small gains in performance. However, the random forest algorithm does not incorporate pruning due to using full-grown trees seldom costs much and results in one less tuning parameter (Friedman, Hastie, and Tibshirani, 2001). Broadly speaking, the effect of overfitting in a random forest is mitigated by the fact that the samples used to train the individual trees are "bootstrapped" and also by the fact that you have a multitude of random trees using random features and thus the individual trees are strong but not so correlated with each other.

3.5.4 Training & Testing the Machine Learning Techniques

This section will speak about how I utilized the training and testing sets which I mentioned previously in this chapter. The training and testing of the machine learning techniques are two of the key steps in my project as they are how I will generate results that will allow myself to answer the project proposal question which I have stated earlier in this paper. The process which I utilized for this procedure will be outlined clearly and was rigorously followed when generating all of the results presented in this paper. This is so that the results obtained can be comparable to each other and also recreated.

The process of training and testing the machine learning techniques comprises of a number of sections; deciding what features from the data sets are needed from inputs to the machine learning technique, prepping that data, creating a model from the training data and evaluating how effective this model is again the test set. This must be applied to all three of the chosen machine learning techniques.

3.5.5 Deciding What to Test

As previously mentioned, I wish to discover whether I can use machine learning to develop an accurate personalized search over Linked in profiles. I aim to test this by classifying the profiles by:

1. Job role e.g. '*Software Engineer*'

2. Job role and company name e.g. '*Software Engineer* employed by *Google*'

With this in mind I analyzed the dataset to identify what users, job roles and companies would be good candidates for this testing process.

3.5.6 Prepping the Data

Prior to this stage each raw HTML file has been parsed into a data type that represents the Linked In user profile. These data types have then been grouped into testing and training sets. When analyzing this parsed data in the sets I discovered that there was a number of modifications and filtering that needed to be performed on it in order to make it suitable for machine learning.

Numerical vs Categorical Data

The features that I extracted from the Linked In user profiles can be grouped into one of two types - numerical and categorical.

Numerical data is data that can be measured. These figures can be placed in ascending and descending order. An example of numerical data from my dataset would be how many years experience a person has worked for. This data is well suited for machine learning as it is easy to plot these values and calculate the Euclidean distance between their points.

Categorical data deals with data which can be characterized into groups but cannot be measured or sorted in an order. For example, the skills which a person lists on their profile are a type of categorical data. However we cannot say that one skill has a higher value than another the same way we can with numerical data - i.e. we can't say that the skill *'Leadership'* has a higher value than the skill *'Programming'*. Due to this it is not possible to plot these skills and calculate the Euclidean distance between them without first representing these categorical differently.

In order to be able to calculate the Euclidean distance between all of these variables, I needed to represent each value numerically. To do this I created a feature in the machine learning input for each possible categorical value. If a user possesses that categorical value in their profile, then the corresponding feature will be assigned the value of 1. If the user does not possess that categorical value in their profile then the corresponding feature will be assigned the value of 0. This process is demonstrated below.

Table 3.1 shows a dataset containing simplified versions of two Linked In user profiles. Each profile possesses two categorical skill values.

Table 3.2 shows how the two Linked In profiles can be converted into a format which makes it possible to plot and calculate the Euclidean distance between the two profiles. Across this entire example dataset there are three possible skill values - *Leadership*,

TABLE 3.1: Example of a dataset with two simplified profiles and their corresponding skills

Profile #1 Skills	Profile #2 Skills
Leadership	Leadership
Management	Programming

Management and *Programming*. Both profiles contain the *Leadership* skill so they have been assigned the value of 1 for that feature. Profile #1 contains the *Management* skill and thus is given a value of 1 for that input. However Profile #1 does not have the *Programming* skill and thus is given the value 0 for that value. Similarly this logic is applied to Profile #2 to build its input.

TABLE 3.2: How the categorical skill data of the profiles can be represented as a numerical value to the machine learning input

	Leadership	Management	Programming
Profile #1	1	1	0
Profile #2	1	0	1

3.5.7 Building a Machine Learning Model

Since I have formatted and filtered the data from the sets into a suitable format, I am now able to use this data as an input into a machine learning algorithm. I will now detail how this formatted information is used as an input for each one of the three machine learning techniques I have implemented as well as any other parameters that are involved with the construction of these models.

K Means

As the K Means algorithm is an unsupervised technique, I do not need to supply labeled input data i.e. the formatted data which I have created is sufficient. However, I do need to specify the number of centroid (clusters) that I wish to run the algorithm with. For this I develop the algorithm by placing with two starting centroids at random locations throughout the multi dimensional space. I have chosen to place two as I want to generate two clusters - one representing the user profiles with the chosen job role, e.g. '*Software Engineer*', and one representing all other user profiles who do not have the chosen job role. The K Means algorithm will group the user profiles into clusters that have similar features and by doing so will hopefully result in the clusters representing the separate types of user profiles.

Support Vector Machine

The Support Vector Machine algorithm is a supervised technique. This means that I need to supply the algorithm with labeled data in order to build a model. To achieve this, I further enriched the formatted data from the training set with labels that correspond to the expected output of the item. With this new data inputs, support vector machine is capable of building the model.

Random Forest

Similarly to support vector machine, random forest also requires labeled input data. Once again, I enrich the formatted data from the training set with labels that correspond to the expected output of the item. in addition to this, the algorithm also requires the size of the forest as an input. This size relates to the number of decision trees that are used to populate the forest. As mentioned, in general more decision trees results in more accurate findings. I will test the random forest algorithm with a number of different forest sizes to see if this is the case with my data.

Accord.NET Framework

The Accord.NET Framework is a .NET framework for machine learning and scientific computing that is written completely in C#. It provides

FIGURE 3.9: Accord.NET Logo

functionality for building models for the three chosen machine learning techniques I have chosen. I have utilized it to assist my development of the techniques.

Additional Parameters

In addition to the technique specific parameters which were discussed to section 3.5.7, I also identified a number of additional parameters when creating the machine learning models. Unlike the technique specific parameters, these can be applied to all machine learning techniques. These additional parameters can significantly impact the results generated from the models. I modified these parameters during the testing phase to obtain a wider range of findings using the same training a testing data. They were:

- α i.e. a threshold for including the percentage of most frequently occurring skills in the model. This threshold was included as a parameter as the vast amount of features identified were derived from skill attributes. This is as a result of the skills being categorical variables and thus each skill needing its own feature input to the machine learning model. However, most of these skills were irrelevant as only a very small number of users possessed them and due to this they had little or no usefulness for classification. If using a low threshold number, the model would be created using only the most frequently occurring features. Increasing the number results in additional features being included as inputs to the model.

- **T** i.e. the minimum feature threshold. The '*minimum feature threshold*' represents a threshold that all profiles must adhere too. This threshold will remove any profiles that do not contain the required number of skills in their profile. In general, by setting a high '*minimum feature threshold*' more profiles will be included in the creation of the model that if a low number was set. This threshold was introduced to remove profiles with few or no features which can be hard to classify correctly.

3.6 Linked In Search User Interface

As part of this project, I created a user interface to assist the user in interacting with the system that I built. This user interface was built using the .NET framework. Here I will explain how I implemented it and the functionality that it provides to the user.

3.6.1 .NET framework - An Overview

The .NET framework has been developed by Microsoft and runs primarily on Microsoft Windows. The framework is designed to aid the process of developing code by providing a number of time saving features. I chose to use this framework as my primary programming language for my project as I felt it was a good option for a number of reasons.

There are a large number of resources available to .NET developers. These range from small open source libraries to large professionally developed frameworks. When designing my Linked In Searching system I made use of a number of these resources in order to simplify the tasks which needed to be completed. The .NET *Class Library* provides thousands of of object oriented classes. These classes can be used by any .NET language. Development time can be reduced hugely thanks to the availability of this code base.

Common Language Runtime (CLR). The CLR is the foundation of the .NET Framework. It provides many features to its users. Memory management is handled automatically. From a development standpoint, this means that I do not have to write code to perform memory management tasks when I develop applications that use .NET managed code. Automatic memory management can eliminate common problems, such as forgetting to free an object and causing a memory leak, or attempting to access memory for an object that has already been freed.

The CLR gives also gives developers the ability to write code in various .NET languages and have that code interact and communicate with each other. To achieve this, the CLR's language compiler converts the source code of the managed application into Microsoft Intermediate Language (MSIL). The MSIL code is then converted into CPU-specific code through using a Just-In-Time (JIT) compiler. This means that processing and memory is not being used to convert all of the MSIL into the CPU specific code at once, instead the MSIL is converted on an as-needed basis during execution.

.NET specifies rules that all of its data types and languages must follow. These rules include how types are declared, used and managed. The purpose of this is so that every type can be converted into MSIL code. These types can be 'Value Types' or 'Reference Types'. A 'Value Type' are stored on the stack while 'Reference Types' are stored on the heap. The Common Type System also provides a number of basic data types that any .NET language can avail of such as bools and integers.

In addition to these features, .NET also provides the C# programming language. This language is one that I am comfortable developing code with. I has a small amount of experience of using the language. It provides a number features which are attractive to myself from a programming point of view.

- It is object oriented. This means it will be relatively easy for me learn and use compared to other types of programming languages such as functional. This is due to the vast majority of my programming experience has been using object oriented languages such as Java and C++.

- It produces efficient programs. Which is important due to the requirement of my program to search through large amounts of data.

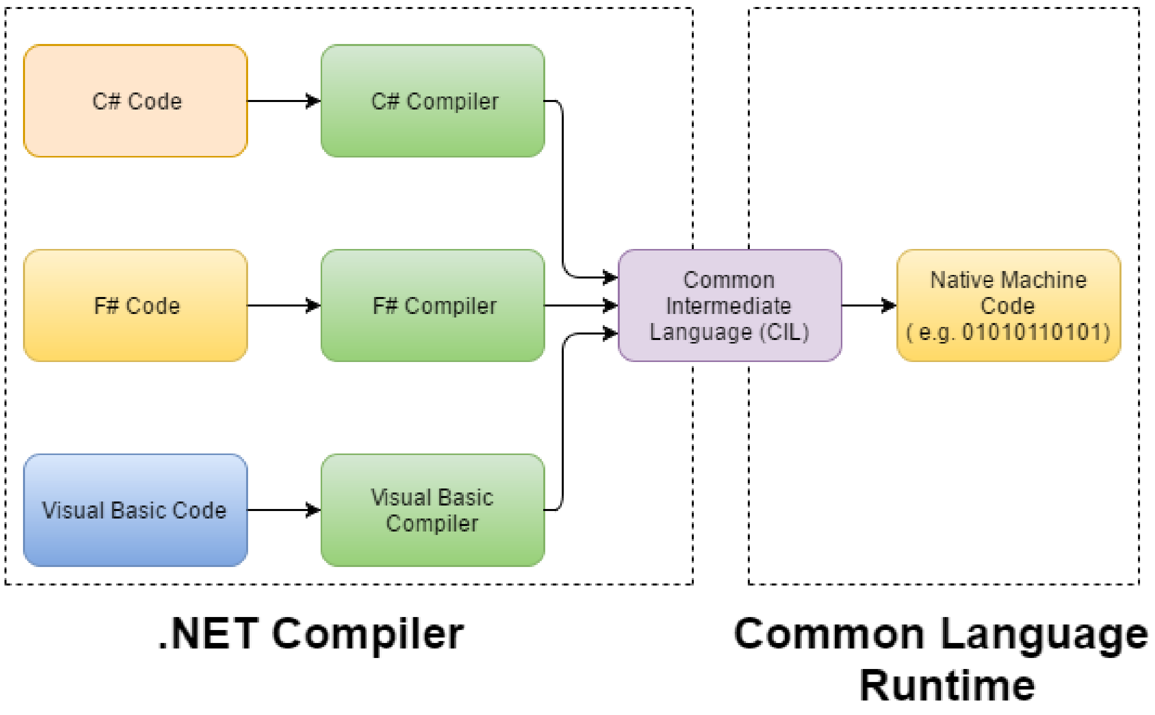


FIGURE 3.10: An overview of how code is compiled using the Common Language Runtime

3.6.2 Windows Presentation Foundation

Windows Presentation Foundation (WPF) is Microsoft's newest approach to a Graphical User Interface framework. "WPF is the answer for software developers and graphic designers who want to create modern user experiences without having to master several difficult technologies" Nathan, 2006.

In Nathan's book, he summarizes what he believes are WPF's main strengths:

- **Broad integration**. WPF provides its developers with the opportunity to create complex applications without the need to utilize a number of separate technologies. For example, previously it would be impossible to develop an application on Windows that had multiple features such as video, speech, 3D and 2D graphics by only using on technology. This functionality can now be achieved through the WPF framework.

- **Resolution independence**. WPF has placed a large emphasis on vector graphics. This has resulted in the ability to easily create graphical user interfaces (GUI's) that scale well to different screen resolutions.

- **Hardware acceleration**. WPF is built on the Direct3D framework which is an application programming interface (API) created by Microsoft that is used to render three-dimensional objects. UI content created in a WPF application is converted to Direct3D object which are then in turn rendered by hardware. This can result in WPF application receiving a performance boost through some of the computational work being handed off to Graphical Processing Units (GPUs) instead of Central Processing Units (CPUs).

- **Declarative Programming**. Nathan tells us how declarative programming is not unique to WPF but WPF has taken declarative programming to the "next level" through its implementation of Extensible Application Markup Language (XAML).

- **Rich composition and customization**. The controls that make up a WPF UI are highly adaptable. They can be easily modified to provide functionality vastly different to their original purpose.

Nathan states that WPF aims to combine the best attributes of a number of separate systems. Direct3D, Windows Forms (WPF's predecessor), Adobe Flash (powerful animation support), and HTML (declarative markup). Although WPF does not enable a programmer to develop systems that he could not have previously done, it does however make the process of developing much less costly in terms of both money and time spent. WPF makes it easier than ever before to create all kinds of rich user interfaces.

It is due to these reasons that I have opted to create a WPF application to develop my software.

3.6.3 Design Pattern

Structuring code is an important part of creating any piece of software. Using a design pattern can greatly reduce development time for creating new features, debugging code and implementing modifications to an existing code base. As I needed to develop a substantial piece of software I felt it was necessary to implement a robust design pattern that would allow my to quickly add new and change existing features. To achieve this I decided to utilize the model–view–viewmodel and dependency injection design patterns to structure my code.

Model View View-Model (MVVM)

MVVM is a design pattern similar to the widely used model-view-controller (MVC). Like all design patterns, its goal is to make the development of code more structured and easier to manage. It allows a developer to reuse code throughout the application and also assists in UI development. To achieve this it separates the front end UI code from the back end logic. UI components created in the view are bound to components within the view-model. The view-model has the ability to interact with the model in order to access the back end logic. This process is shown in figure 3.11.

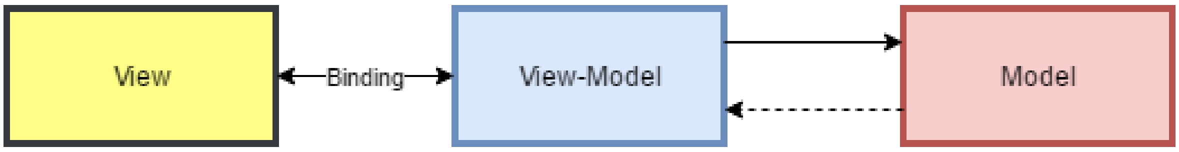


FIGURE 3.11: MVVM Overview

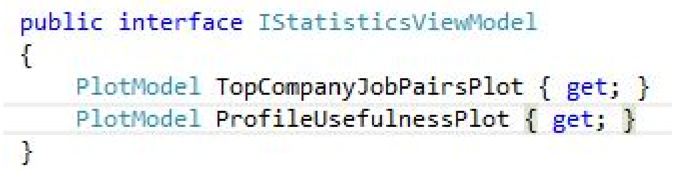
Dependency Injection

"Dependency injection is one of the techniques in software engineering that improves the maintainability of a software application by managing the dependent components."

A dependency is a reference to a foreign object from within an object. Dependency injection is a software development pattern where the foreign *dependency* is passed to the object via an external method. This passing of the dependency is the fundamental trait of the dependency injection pattern. This results in the ability to build loosely coupled software systems which are both easier to maintain and test.

I chose to incorporate the dependency injection design pattern into my system due to these reasons. I felt that spending the extra time designing the code at the beginning of the project using this pattern would make the development process easier further down the line. I believe that this design decision paid dividends as I was capable of modifying and restructuring my code without too much trouble throughout the process of developing the application. To implement the dependency injection design pattern into my system I opted to use the free open source lightweight .NET library Ninject.

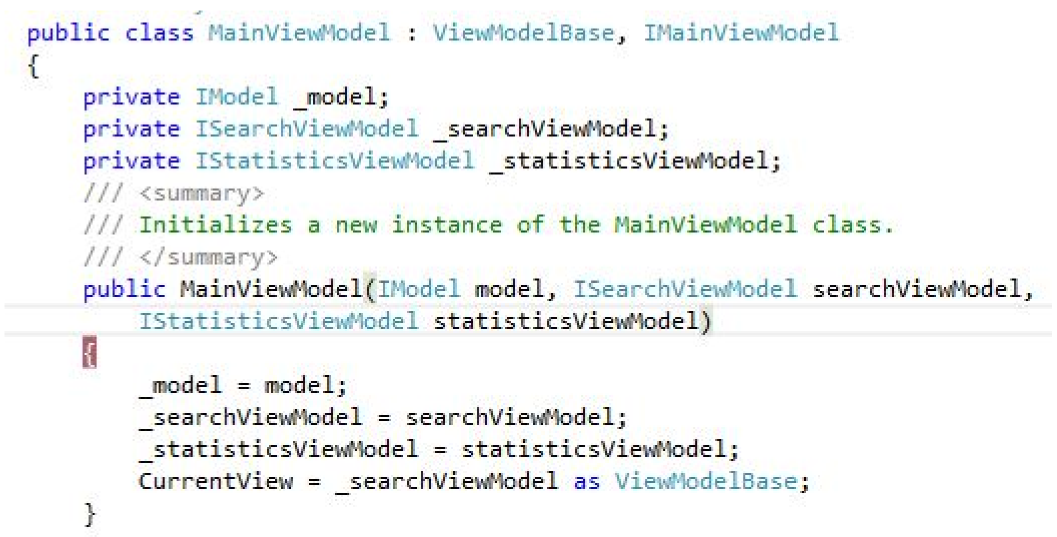
In order to utilize the pattern correctly, interfaces were created for all of the classes that would be *injected*. Using interfaces is a good software design practice as they act as a type of contract that enforces the system to interact with a class in a specified manner. These interfaces were then injected into the constructors of classes



```
public interface IStatisticsViewModel
{
    PlotModel TopCompanyJobPairsPlot { get; }
    PlotModel ProfileUsefulnessPlot { get; }
}
```

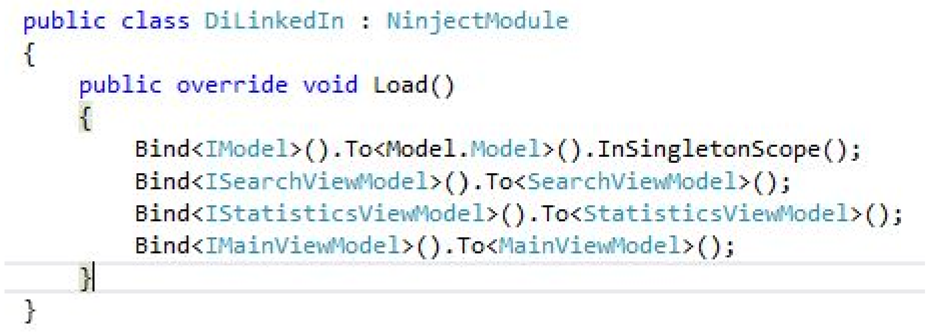
FIGURE 3.12: Example of an interface.

In order to resolve the injected interfaces to concrete classes, the system must be told what classes to implement. To achieve this the correct bindings must be created. Due to the relatively large size of my application I felt it was a good idea to simplify the adding of many bindings. Ninject provides an abstract class *NinjectModule* that helps build this functionality.



```
public class MainViewModel : ViewModelBase, IMainViewModel
{
    private IModel _model;
    private ISearchViewModel _searchViewModel;
    private IStatisticsViewModel _statisticsViewModel;
    /// <summary>
    /// Initializes a new instance of the MainViewModel class.
    /// </summary>
    public MainViewModel(IModel model, ISearchViewModel searchViewModel,
        IStatisticsViewModel statisticsViewModel)
    {
        _model = model;
        _searchViewModel = searchViewModel;
        _statisticsViewModel = statisticsViewModel;
        CurrentView = _searchViewModel as ViewModelBase;
    }
```

FIGURE 3.13: Example of a dependency injection through a constructor.



```
public class DiLinkedIn : NinjectModule
{
    public override void Load()
    {
        Bind<IModel>().To<Model.Model>().InSingletonScope();
        Bind<ISearchViewModel>().To<SearchViewModel>();
        Bind<IStatisticsViewModel>().To<StatisticsViewModel>();
        Bind<IMainViewModel>().To<MainViewModel>();
    }
}
```

FIGURE 3.14: Ninject's abstract class NinjectModule being implemented in the application.

3.6.4 Lucene.Net

As part of building this system. I decided to implement a traditional search function that I could use to query the many user profiles that I have at my disposal. Doing this would allow me to easily investigate the data through the UI I built. In addition, it would allow me to generate a base line search result that I could compare with and contrast against the search results of my machine learning models. This gives myself an extra method of evaluating the success of the machine learning models. To implement this search functionality, I opted to use Lucene. A lot of the information I used to implement Lucene was through reading the book '*Lucene in Action*' Hatcher and Gospodnetic, 2004.

Apache Lucene is a free and open-source information retrieval software library, originally written in 100% pure java. In recent years Lucene has become the most widely used information retrieval library. It is a high-performance, scalable information retrieval library that allows developers add search functionality to their projects. Lucene provides a core API for all of its features and through this I was granted the ability to fine tune my personalized talent search algorithm. It is because of these reasons that I decided to use it for my system.

Lucene.Net is an exact port of the original Lucene search engine library, written in C#. It provides a high performance, full featured text search engine library. Its features make it suitable for applications such as mine which require the ability to search through text.

3.6.5 Indexing Data using Lucene.Net

The Indexing process is one of the core functionality provided by Lucene.Net. Firstly a *Document* comprising of a number of *Fields* is created. A *Document* is a searchable item in Lucene.Net. It is up the the application to create these *Documents* during the indexing phase.

Fields are the most important and the foundation unit of indexing processing. It is the actual object containing the contents to be indexed.*Fields* are key-value pairs containing keys as names and values as contents to be indexed. Each *Field* can be configured to be either *stored* or *indexed.* If we would like to use a field during the search then the field must be *indexed.* If we only store the Field, it's value can't be reached by the search queries. During searching, when a Document is 'hit' Lucene.Net will return the entire Document object. The stored Fields in the document will be available within this returned the document whilst the indexed field will not be. An indexed Field is information used to find an Document, a stored Field is information

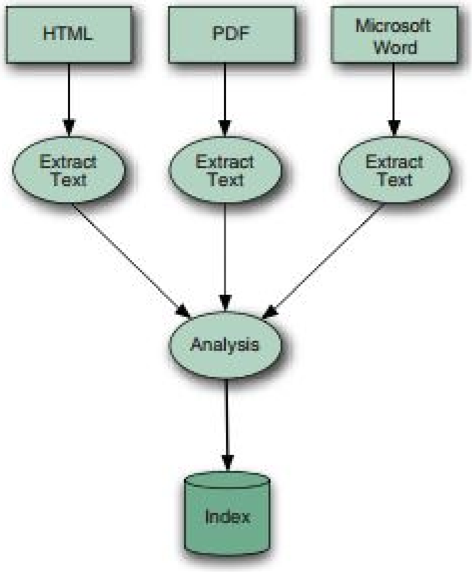


FIGURE 3.15: An overview of the indexing process performed by Lucene. Text is extracted from a document, analyzed and then added to the index. [McCandless, Michael, Erik Hatcher, and Otis Gospodnetic. Lucene in Action: Covers Apache Lucene 3.0. Manning Publications Co., 2010.]

returned with the Document. Two different things.

An indexed field can either be analyzed or not analyzed. If the field is analyzed then the Analyzer will break the field's value into a stream of tokens and each token is search able separately. This can result result in searches returning 'hits' when only a partial amount of the fields value in matched with the query. If the field is not analyzed then the whole field value must be matched to get a 'hit'.

When these *Documents* have been analyzed by an *Analyser* and added to the index, an *IndexWriter* is then used to create/update indexes during the indexing process. It is another core component of Lucene.Net indexing. The *IndexWriter* populates a *Directory* that is used to store the indexed data.

3.6.6 Searching Data using Lucene.Net

The Directory that was created during the indexing phase is passed to an IndexSearcher. The Directory is opened using an IndexReader. A Query with a Term is created and passed the the IndexSearcher to perform the search. A TopDocs object is returned which contains the ids and contents of the Documents that were matched by the search.

3.6.7 Scoring Data using Lucene.Net

When a document is matched during a search, Lucene assigns that document a score based on how good of a match the document is to the query.The higher the score that is generated, the more similar the matched document is to the query. The score is computed for each document (d) matching each term (t) in a query (q).

3.6.8 Scoring Formula

Lucene's scoring formula is called the 'similarity formula'. This name is given due to the formula computing the similarity between a query and each document that is matched. This algorithm computes a raw score in the form of a positive floating point number. By default, Lucene returns matching documents sorted by this score resulting in the top documents being the best matching ones. This formula is as follows:

$$\sum_{tinq}(t\,f(t\ in\ d)\ idf(t)\ boost(t.field\ in\ d)lengthNorm(t.field\ in\ d))coord(q,d)queryNorm(q) \tag{3.1}$$

Table 3.3: Factors in the scoring formula

Input	Description
tf(t in d)	Term frequency factor for the term (t) in the document (d). Documents that have more occurrences of a given term receive a higher score. This is beneficial as documents with more occurances of a term tend to be good matches for a query
idf(t)	Inverse Document Frequency of the term. This acts as a measure of how "unique" the term is i.e. its frequency among the matched documents. Rarer terms have a higher impact on the similarity score. The rationale behind this is common terms are less important than infrequent terms.
boost(t.field in d)	Field and document boost. This value is set when indexing the field. This can be used to statically boost certain fields and documents over others.
lengthNorm(t.field in d)	Measure of the importance of a term according to the total number of terms in the field. This value is computed during indexing and stored in the index norms. Shorter fields (ones that have fewer tokens) get a larger boost from this value.
coord(q, d)	Coordination factor, based on how many of the query terms are found in the document. Typically, a document that contains more of the query's terms will receive a higher score than another document with fewer query terms. The factor gives an AND-like boost to documents that contain more of the search terms than other documents.
queryNorm(q)	Normalization value for a query. Queries are normalizes so that they can be easily compared. This factor does not affect document ranking.

3.6.9 Boost Factors

A boost factor can be built into the equation to allow the ability to affect a query, fields or even documents influence on its score, and thus influencing the ranking of the document in the matched results. The default value for boosts is 1.0. This boost can be modified either at query-time or during the indexing phase.

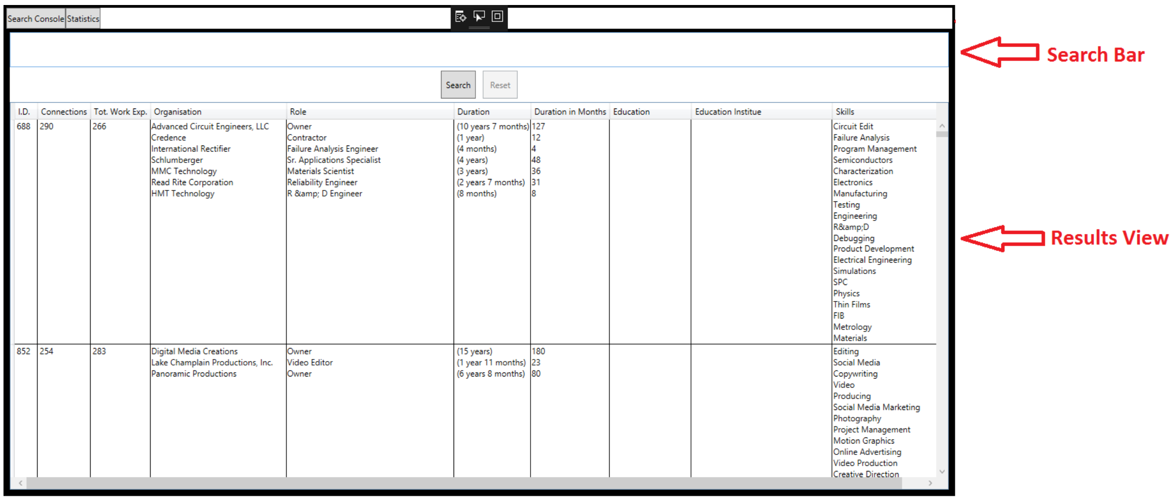
Other factors can be computed on a per-query basis as part of the queryNorm factor. When only a single term is present in a search, using a boost then becomes obsolete as the boost will impact each matched document equally. In a multi-term query, some documents may match whilst other may not. This can result in the boost factor discriminating between the returned documents.

3.6.10 User Interface

This section will show the implementation of the work that I have mention above. The UI comprises of two main sections, a search screen that allows a user to search through the user profiles and a statistics view that shows a range on information generated by the system.

Search Console

The search console is a simple view that has two primary components. The search bar which acts as a Google type search input and the results view which shows the information from the profiles returned by the search query inputted in the search bar The search console exploits the Lucene.NET search engine that I described above. AS can be seen from figure 3.16, The results view includes all of the extracted information from a user profile. Also, you can see that the users name is not present in the results as it has been removed from the profile due to the ethical reasons mentioned previously in this paper. The main purpose of the search console for this project was to provide myself with a method for investigating the data that I was using the train and test the models with.



I.D.	Connections	Tot. Work Exp.	Organisation	Role	Duration	Duration in Months	Education	Education Institue	Skills
688	290	266	Advanced Circuit Engineers, LLC Credence International Rectifier Schlumberger MMC Technology Read Rite Corporation HMT Technology	Owner Contractor Failure Analysis Engineer Sr. Applications Specialist Materials Scientist Reliability Engineer R &amp; D Engineer	(10 years 7 months) (1 year) (4 months) (4 years) (3 years) (2 years 7 months) (8 months)	127 12 4 48 36 31 8			Circuit Edit Failure Analysis Program Management Semiconductors Characterization Electronics Manufacturing Testing Engineering R&amp;D Debugging Product Development Electrical Engineering Simulations SPC Physics Thin Films FIB Metrology Materials
852	254	283	Digital Media Creations Lake Champlain Productions, Inc. Panoramic Productions	Owner Video Editor Owner	(15 years) (1 year 11 months) (6 years 8 months)	180 23 80			Editing Social Media Copywriting Video Producing Social Media Marketing Photography Project Management Motion Graphics Online Advertising Video Production Creative Direction

FIGURE 3.16: The Search Console.

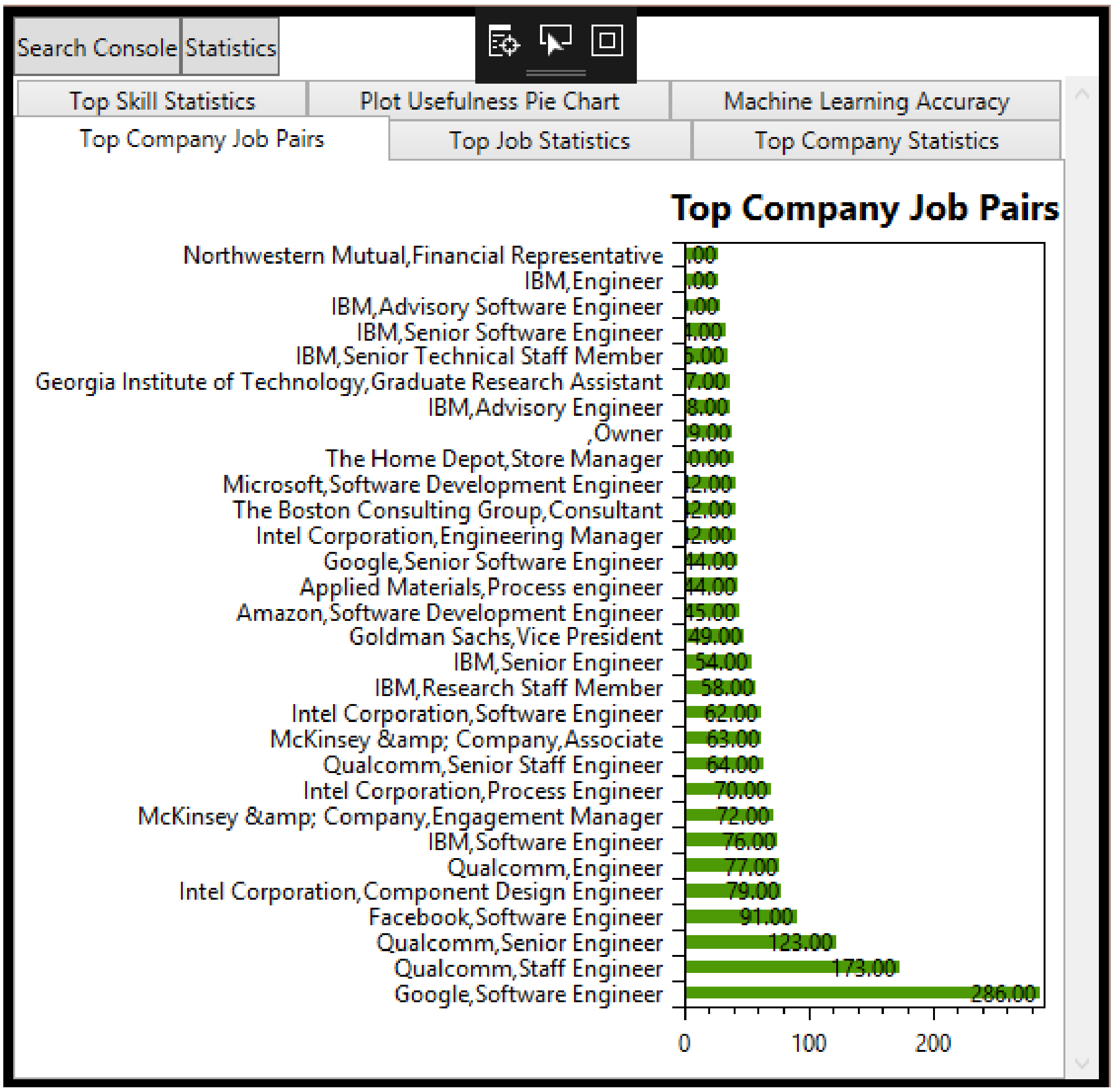
Statistics View

The statistics view visualizes statistical data from the system. The view has six sections, each of which contain different data relevant to the project. The six sections are:

- Top Company Job Pairs - Shows information regarding the most frequently occurring company job pairings throughout the dataset.
- Top Job Statistics - Shows information regarding the most frequently occurring jobs throughout the dataset.
- Top Skill Statistics - Shows information regarding the most frequently occurring skills throughout the dataset.
- Plot Usefulness Pie Chart - Shows information regarding the usefulness of the LinkedIn user profiles.
- Top Company Statistics - Shows information regarding the most frequently oc-

curring companies throughout the dataset.

- Machine Learning Accuracy - Shows information regarding the accuracy of the machine learning models that were created.



Search Console
Statistics
Top Skill Statistics
Plot Usefulness Pie Chart
Machine Learning Accuracy
Top Company Job Pairs
Top Job Statistics
Top Company Statistics
Top Company Job Pairs
Northwestern Mutual,Financial Representative
.00
IBM,Engineer
.00
IBM,Advisory Software Engineer
.00
IBM,Senior Software Engineer
4.00
IBM,Senior Technical Staff Member
5.00
Georgia Institute of Technology,Graduate Research Assistant
7.00
IBM,Advisory Engineer
8.00
,Owner
9.00
The Home Depot,Store Manager
0.00
Microsoft,Software Development Engineer
12.00
The Boston Consulting Group,Consultant
12.00
Intel Corporation,Engineering Manager
12.00
Google,Senior Software Engineer
44.00
Applied Materials,Process engineer
44.00
Amazon,Software Development Engineer
45.00
Goldman Sachs,Vice President
49.00
IBM,Senior Engineer
54.00
IBM,Research Staff Member
58.00
Intel Corporation,Software Engineer
62.00
McKinsey &amp; Company,Associate
63.00
Qualcomm,Senior Staff Engineer
64.00
Intel Corporation,Process Engineer
70.00
McKinsey &amp; Company,Engagement Manager
72.00
IBM,Software Engineer
76.00
Qualcomm,Engineer
77.00
Intel Corporation,Component Design Engineer
79.00
Facebook,Software Engineer
91.00
Qualcomm,Senior Engineer
123.00
Qualcomm,Staff Engineer
173.00
Google,Software Engineer
286.00
0
100
200

FIGURE 3.17: The Statistics View.

Chapter 4

Research Findings

4.1 Introduction

This chapter will present the various research findings that my project has generated. Firstly, I will present the statistics that were used to evaluate how successful a given machine learning model was on test data. I will then discuss the results that were generated from each of the experiments I conducted. The result tables for each of the experiments can be found in the appendices.

4.2 Method to Evaluate the Results

Confusion Matrix

In machine learning, a confusion matrix is used to represent visually the performance of a machine learning model. Each column represents an instance of the predicted classification of an item and each row represents the actual classification of an item. The states in a confusion matrix can be summarized as follows:

- *True Positives (TP)* - The True Positive value represents the number of correct item that were correct classified by the algorithm. A high True Positive value is preferential as it corresponds to a better performance of the algorithm.

- *False Positives (FP)* - The False Positive score represents the number of correct items that have been incorrectly predicted by the algorithm. A low False Positive score is preferential.

- *True Negatives (TN)* - The True Negative score represents the number of incorrect items the have been correctly classified as being incorrect. Higher True Negative scores are better.

- *False Negatives (FN)* - The False Negative score represents the number of correct items that have been wrongly classified as incorrect. A low value for this is preferential.

TABLE 4.1: Confusion Table Layout

	Predicted Positive	Predicted Negative
Actual Positive	TP	FP
Actual Negative	FN	TN

Precision

In information retrieval, the precision score is the fraction of retrieved results which are relevant to the query. Having a high precision score means that there is a high probability that the retrieved results will be accurate. Having a low precision score will result in the retrieved results having a poor accuracy. The precision score can be calculated as follows:

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{4.1}$$

Recall

The recall score represents the percentage of the correct results that are returned from the from the query out of all of the correct results that exist in the dataset. Having a high recall value means that the query returns a large amount of the correct results. A low value will result in the opposite.

$$Precision = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{4.2}$$

Recall and precision are not the same. For example, suppose I have a dataset with 100 items - 50 of which are correct and 50 of which are incorrect. Now suppose that I perform a query over this dataset which will return a single item and that this item is correct. This query will have a very high precision score, 100% to be exact, as all of its results are correct. However it will have a very poor recall score of 2% as the query only managed to return 1 correct result out of a possible 50.

Ideally a query will return a high percentage of all of the correct items and a low percentage of the incorrect ones. This will results in both a high precision score and a high recall score.

F Score

The F Score is a measurement of a query's accuracy. The F score is the mean value of the query's precision and recall scores.

$$FScore = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{4.3}$$

Lucene Baseline Accuracy

The accuracy score here, is determined by the number of correctly classified items i.e. the number of True Positives and True Negatives dived by the total number of items in the dataset.

$$Accuracy = \frac{TruePositives + TrueNegative}{TruePositives + FalsePositives + FalseNegatives + FalsePositives} \tag{4.4}$$

The Lucene baseline is used as an additional method for evaluating the machine learning algorithms. The accuracy score represents how accurate Lucene is at returning the relevant documents from the query when the job and/or company data has been removed does not exist.

4.3 Results from the Training Data

Here, I present the findings that I have discovered from running the experiments. For testing the data, I chose to implement a number of separate use cases. Machine learning models were trained specifically for each use case. The complete list of results which I have included in this dissertation can be found in the appendices at the end of the paper. I will use references when speaking about specific result sets.

As mentioned in the previous chapter, the tables which show the data have two input parameters α and $\mathbf{T}$. These tables show the results achieved by the models using different values for the two input parameters. Also, each cell in these tables represents a new machine learning model that was trained and tested with data that corresponds to the two parameters.

4.3.1 Classifying Profiles by Job Only

The first objective of the testing phase was attempting to classify the profiles by only the job role that exists in the user profiles.

Software Engineers

As Software Engineer was the most frequently occurring job role throughout the dataset, it was selected to be the first job to be tested.

F score
The SVM results for the classification of the Software Engineer testing set are positive. The models created all achieve FScore results of greater that 70%. Also, there is a positive correlation between the FScore and the α value. The scores achieved by Random Forest are competitive with those achieved by SVM when there is a high α value and low **T** value. When this is a not the case, then the random forest model is not nearly as competitive. The Fscore results of the KMeans models are very inconsistent. The model did not manage to clusters the users very successfully.

Recall
The SVM model achieved positive recall results across the boards for all tests performed. It can be noted that in general, there was a positive correlation between the α value and the recall score. Less successful than SVM and more successful than KMeans - random forest achieves recall results fluctuate a lot depending on the input parameters. Similarly to the FScore, having a high α value and low **T** value results in a better result. The KMeans recall results were again relatively poor when compared against its counterparts.

Precision

The precision of the SVM model does not match the high scores achieved by its recall results. The precision scores consistently range between 60% - 70%. Random Forest achieves the best precision scores for classifying the data, slightly eclipsing the results of SVM. Similarly to the FScore and Recall results, the precision results of the KMeans models do not satisfy

Research Assistants

Following on from the training and testing of different types of machine learning models using the 'Software Engineer' dataset, I wanted to select a job role which was both frequently occurring and would comprise of mostly different features to that of a Software Engineer. From reviewing the most frequently occurring jobs, I selected the job of 'Research Assistant' as I felt it would be a good candidate for generating results due to it satisfying the above requirements in my opinion.

F score

The Fscore achieved by SVM when classifying the Research Assistant dataset is very positive. All models achieved an F-score of approximately 80%. Random forest generated successful F-scores too. These scores were slightly more erratic when compared to those associated with SVM, however the results are still encouraging as they at times even eclipsed those achieved by SVM. At times, K Means scored good F-scores but these positives are mixed with negatives. Overall, the performance of the models on this dataset is an encouraging improvement when compared to the previous testing that has been undertook.

Recall

The results show extremely accurate recall results for the SVM models. An average of approximately 90% across all models proves how SVm can be a good candidate for classifying the user profiles which are sought after. A high recall score of 94.5% shows the potential that random forest has for identifying the desired profiles. The rest of the results are also positive especially those achieved using an α value of 0. The K-means model has generated very inconsistent recall result values. The recall scores of 1 that can be seen in the table are as a result of all data clustering to one cluster. These corresponding models have generated poor precisions scores.

Precision
Although they are not bad, the SVM precision scores do not match Svms recall scores. It is due to this that the SVM F-scores are substantially lower than the recall scores. Once again, random forest has provided consistent and relatively high precision models when compared to its counterparts. K-means provides poor and erratic precision results as can be seen in the relevant figures in the appendices.

4.3.2 Classifying Profiles by Job and Company Pairings

As explained previously, the main goal of this dissertation was to investigate whether it is possible to classify profiles based on both job role and company. This sections outlines the findings I made from this investigation.

Software Engineers working for Google

By generating statistics regarding the job and company pairings that exist in the dataset(see figure 4.1), I discovered that the most frequently occurring pairing was Software Engineers who were employed by Google. Due to this, I opted to use that pairing as the first to test.

F score
Support Vector Machine. Looking firstly at the FScore results obtained by the SVM model we can see that in general the score improves as the α threshold increases.

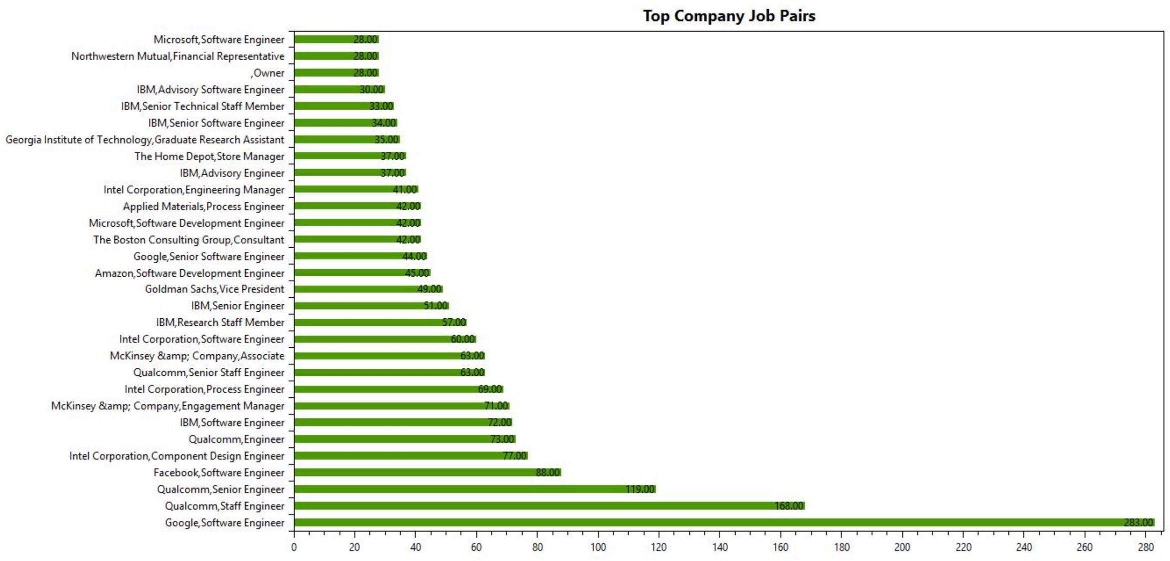


FIGURE 4.1: The top 30 Company Job Pairs

Modifying the **T** parameter seems to have little impact on the resulting FScore result.
Random Forest. Interestingly, the random forest algorithm performed best when it used a value of α=1 and **T**=0 (see figure). This means that the most successful version of the model was created through not including any skills that were present in a profile as features. The classification was made using only the other features of the profile e.g. number of connections, number of previous jobs etc.
KMeans. The table of findings shows the Fscore results of the KMeans model. In general, low α vlues coupled with middling **T** values achieved the highest results for the K Means model.

Recall
Support Vector Machine. SVM achieves very accurate results for the recall values. Averaging 80%+, this shows that SVM can be relied upon to return a high proportion of the correct profiles.
Random Forest. This model does not provide the same accuracy consistently when it comes to recall as SVM. It does occasionally match he values of SVM when there is a low **T** threshold.
KMeans. KMeans has extremely fluctuating recall results. It peaks in the high 90%

and dips to its lowest at 25%. This fluctuation can be attributed to the random location of the starting centroids. Also, The KMeans model has poor precision scores which correspond to it's high recall scores.

Precision
Support Vector Machine. Figure XX, shows that SVM provided consistent precision values of between 60% - 70% across all of the models tested.
Random Forest. Similarly to SVM, random forest too provided consistent precision values. Random forest did however provide more accurate precision values compared to SVM.
KMeans. Looking at the findings, we can see that the KMeans models scored significantly lower that its machine learning counterparts for precision.

Component Design Engineer working for Intel Corporation

Similarly to the job only classification, for the job and company classification I training and tested the algorithms using a secondary dataset. The pairing that I chose to implement this time was *Component Design Engineer*s who are employees at the *Intel Corporation.* I chose this pairing because I felt it had a significant different to the previous pairing that was tested and may produce different results.

F-score
SVM scores much higher than its counterparts in the F-score results. Varying the value of **T** does not affect the result, however varying the value of α can have a significant impact, with higher values having more accurate results. The Random Forest implementation achieves positive F-scores when using high α values, and more importantly low **T** scores. Overall, random forest provides consistent F-score results. The K Means model achieved very erratic scores. There does not seem to be any discernible pattern in the result that could identify what parameters produce favorable result for the model.

Recall
Once again, SVM achieves a high recall value across all of its model implementations. Higher α values correlate to higher F-scores for the models. Following the trend from the other tests that were undertook, random forest achieves consistent but less accurate recall values when compared to SVM. Also following the trends established in the previous tests, K-means produces erratic and much lower scoring results than its machine learning counterparts.

Precision
Again following the established trends from the results of the other tests, the SVM models produce precision scores which are significantly lower than their F-scores and recall scores. Scoring similar scores to SVM, random forest achieves precision results which are also lower than its F-scores and recall scores. K-means achieves somewhat consistent, yet low scoring precision scores against the data across the range of models which were generated.

4.4 Discussion of Results

Analyzing the Individual Machine Learning Techniques

The large amount of results that were generated from the experiments conducted as part of this dissertation have offered a number of interesting insights. By testing the three machine learning algorithms across multiple use cases and datasets, it is clear that patterns have emerged. I will firstly outline what I have discovered about the three individual machine learning approaches from my experiments with the data.

1. In general, Support vector machine has been the most successful at classifying the data across all use cases. SVM boasts the ability to achieve very high recall rates, especially when classifying the '*Research Assistant*' data. SVM is affected little by the adding of additional 'unuseful' features to the items being classified

(an 'unuseful' feature being a feature that does not help the model classify an item). The high recall values are offset by lower precision values.

2. Random forest proved itself to be capable of challenging SVM as the most accurate classifier and during some examples even performed better than SVM. However, most of the time random forest achieved lower F-scores and recall scores than SVM. However, Random forest did generate more accurate precision scores than its supervised learning counterpart. Also, random forest seemed to perform better when there was high α values and low $\mathbf{T}$ values.

3. K-means proved to be an unsuccessful candidate for classifying the data in question. The technique regularly returned inaccurate and erratic results when attempting to classify the data.

Supervised vs Unsupervised - *Which is Better?*

From analyzing the above summaries of the three machine learning algorithms, it can be inferred that supervised machine learning is far superior technique compared to un-supervised machine learning at classifying the LinkedIn user profiles. The results obtained from both of the supervised machine learning techniques greatly surpassed the results generated by the un-supervised technique

How Successful were the Results?

The aim of this project was to create a machine learning model that could accurately classify user profiles based on the features within it. As has been stated above, support vector machine was the most successful at this classification process. However in order to evaluate if the results achieved by my implementation of support vector machine are good enough to justify it outside the realm of this project, it is necessary to have something to compare them against. Two methodsI have identified as being candidates for this comparison are:

1. Lucene search base line. As can be seen in the results found in the appendix, lucene search was extremely inaccurate at identifying relevant user profiles when the job and/or company information has been removed. The machine learning techniques that were implemented far outperformed Lucene's results.

2. Previous studies. As mentioned in the State of the Art chapter, a previous project conducted by Lops et al., 2011 also attempted to classify LinkedIn user profiles. The most successful results which they achieved from their work see XX, were an accuracy score of 0.71. The support vector machine models which were developed as part of this project regularly tested much higher than this score. Although these are different projects with different objectives, the raw data being used to create the classification models is similar.

It is because of the above reasons which have been derived from the results I have shown, that I believe that machine learning **CAN** be used to accurately classify LinkedIn user profiles

Chapter 5

Conclusion

This chapter summarizes the work that was undertaken and completed as part of this dissertation project. Also, I offer a few thoughts of my own regarding this branch of information retrieval.

5.1 The Software System

The system which was designed with the requirement of allowing a user interact with the information retrieval and machine learning process that was a part of this project was successfully implemented. The code base was structured, developed and maintained to a high standard throughout the many months of developement.

5.2 Summary of Results

As outlined in chapter 4, I believe that the experiments conducted and the associated results help to justify the proposal that information retrieval and machine learning can be used to accurately classify LinkedIn user profiles into suitable and unsuitable employees for a company. I have shown that supervised machine learning is advantageous over un-supervised machine learning in this scenario. More specifically, support vector machine is the most successful of the tested techniques.

5.3 Potential Impact

The trend that employee recruitment is becoming more automated has the potential to have a huge impact on the talent management process. As has been shown in this paper, machine learning has the ability to identify suitable employees for a company. If these findings can be replicated and improved upon in real world industry scenarios, this autonomous process has the potential to completely revolutionize the Human Resource sector. Will organizations strive to develop the most accurate algorithms? Will machines replace managers in the recruitment process? These are just two of the questions that arise.

As for the potential lone impact of this dissertation, I hope that it justifies the research proposal that was set out at the beginning of this paper. I also hope that it has contributed to this avenue of data analytics and will also encourage other researchers to further pursue this topic.

5.4 Future Work

A main objective in future work would be to improve the accuracy of the current machine learning models. This could be done via a number of methods such as:

1. Additional feature extraction from Linked In profiles. This could include extracting job description information, using information from connections etc...

2. Increase dataset size. By having a larger resource pool of LinkedIn user profiles, more complicated machine learning models can be created.

Another objective that I would encourage future researchers in this field to investigate would be to implement different supervised machine learning techniques. My implementations of support vector machine and random forest during this dissertation project has shown has they can effectively be used in this scenario. It would be interesting to evaluate a number of other techniques to compare and contrast their results.

In addition to implementing more supervised machine learning techniques, I also think it would be a worthwhile experiment to employ reinforcement machine learning in this scenario. Doing so would give a wider range of results which can ultimately help to better evaluate the system.

An additional step that would be a part of future work but was far outside the realm of this dissertation project would be to implement this data driven decision making process in a real world organization. Testing how successful applying these technique to recruitment in an actual company would be an insightful method of evaluation.

5.5 Final Thoughts

Every successful organization strives to better themselves on a consistent basis. I believe I have shown that this area of data science has the potential to benefit an organization via improving the recruitment system of said company. Due to this, companies could potentially have a keen interest in utilizing the methods presented in this paper in the future in order to gain an advantage over their competition.

Appendix

Appendix A

Results

Table A.1: K-means Precision Scores for 'Software Engineer' Job Only Classification

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.401	0.564	0.401	0.564	0.401	0.564
0.1	0.578	0.578	0.578	0.578	0.389	0.578
0.2	0.572	0.572	0.572	0.572	0.572	0.572
0.3	0.372	0.594	0.372	0.594	0.594	0.594
0.4	0.422	0.422	0.558	0.558	0.557	0.42
0.5	0.392	0.574	0.574	0.392	0.393	0.393
0.6	0.588	0.588	0.376	0.588	0.588	0.588
0.7	0.594	0.375	0.375	0.594	0.594	0.375
0.8	0.379	0.605	0.379	0.379	0.379	0.379
0.9	0.576	0.576	0.406	0.576	0.576	0.406
1	0.594	0.392	0.6	0.594	0.392	0.386

Table A.2: k-Means Recall Scores for 'Software Engineer' Job Only Classification

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.317	0.683	0.317	0.683	0.317	0.683
0.1	0.679	0.679	0.679	0.679	0.321	0.679
0.2	0.699	0.699	0.699	0.699	0.699	0.699
0.3	0.315	0.685	0.315	0.685	0.685	0.685
0.4	0.356	0.356	0.647	0.647	0.644	0.353
0.5	0.319	0.681	0.681	0.319	0.321	0.321
0.6	0.689	0.687	0.313	0.689	0.689	0.689
0.7	0.677	0.323	0.323	0.677	0.677	0.323
0.8	0.352	0.648	0.352	0.352	0.352	0.352
0.9	0.635	0.635	0.365	0.635	0.635	0.365
1	0.634	0.366	0.638	0.634	0.366	0.362

Table A.3: K-means F-scores for 'Software Engineer' Job Only Classification

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.354	0.618	0.354	0.618	0.354	0.618
0.1	0.624	0.624	0.624	0.624	0.352	0.624
0.2	0.629	0.629	0.629	0.629	0.629	0.629
0.3	0.341	0.636	0.341	0.636	0.636	0.636
0.4	0.386	0.386	0.599	0.599	0.598	0.384
0.5	0.352	0.623	0.623	0.352	0.354	0.354
0.6	0.634	0.634	0.342	0.634	0.634	0.634
0.7	0.633	0.347	0.347	0.633	0.633	0.347
0.8	0.365	0.626	0.365	0.365	0.365	0.365
0.9	0.604	0.604	0.385	0.604	0.604	0.385
1	0.614	0.378	0.618	0.614	0.378	0.374

Table A.4: Random Forest Precision Scores for 'Software Engineer' Job Only Classification

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.636	0.636	0.644	0.647	0.64	0.635
0.1	0.63	0.626	0.63	0.637	0.636	0.655
0.2	0.631	0.63	0.635	0.631	0.623	0.644
0.3	0.658	0.664	0.671	0.658	0.675	0.682
0.4	0.618	0.648	0.647	0.643	0.643	0.65
0.5	0.665	0.654	0.65	0.645	0.659	0.673
0.6	0.667	0.657	0.681	0.673	0.689	0.681
0.7	0.64	0.645	0.661	0.665	0.675	0.68
0.8	0.683	0.667	0.652	0.658	0.668	0.669
0.9	0.654	0.672	0.676	0.686	0.678	0.695
1	0.69	0.681	0.682	0.702	0.693	0.714

Table A.5: Random Forest Recall Scores for 'Software Engineer' Job Only Classification

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.722	0.717	0.691	0.684	0.676	0.653
0.1	0.834	0.667	0.634	0.648	0.657	0.65
0.2	0.836	0.681	0.68	0.66	0.641	0.654
0.3	0.842	0.667	0.651	0.645	0.657	0.654
0.4	0.854	0.701	0.68	0.671	0.691	0.698
0.5	0.721	0.682	0.676	0.661	0.704	0.691
0.6	0.775	0.695	0.685	0.673	0.685	0.669
0.7	0.813	0.685	0.709	0.685	0.709	0.687
0.8	0.738	0.718	0.679	0.692	0.65	0.64
0.9	0.787	0.756	0.737	0.749	0.702	0.679
1	0.787	0.765	0.754	0.765	0.724	0.728

Table A.6: Random Forest F-scores for 'Software Engineer' Job Only Classification

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.676	0.674	0.667	0.665	0.657	0.644
0.1	0.718	0.646	0.632	0.643	0.646	0.653
0.2	0.719	0.655	0.657	0.645	0.632	0.649
0.3	0.739	0.666	0.661	0.651	0.666	0.668
0.4	0.717	0.673	0.663	0.656	0.666	0.673
0.5	0.692	0.668	0.662	0.653	0.681	0.682
0.6	0.717	0.676	0.683	0.673	0.687	0.675
0.7	0.716	0.665	0.684	0.675	0.692	0.684
0.8	0.71	0.692	0.665	0.674	0.659	0.654
0.9	0.715	0.712	0.705	0.716	0.69	0.687
1	0.735	0.721	0.716	0.732	0.708	0.721

Table A.7: SVM Precision Scores for 'Software Engineer' Job Only Classification

α \ **T**	0	0.01	0.02	0.03	0.04	0.05
0	0.622	0.621	0.621	0.622	0.623	0.62
0.1	0.628	0.629	0.628	0.629	0.629	0.628
0.2	0.629	0.628	0.628	0.628	0.628	0.628
0.3	0.662	0.663	0.663	0.662	0.663	0.663
0.4	0.633	0.632	0.631	0.632	0.633	0.632
0.5	0.64	0.641	0.64	0.64	0.641	0.64
0.6	0.655	0.655	0.655	0.654	0.656	0.655
0.7	0.632	0.631	0.632	0.63	0.632	0.633
0.8	0.674	0.674	0.673	0.674	0.67	0.674
0.9	0.619	0.619	0.618	0.618	0.619	0.618
1	0.656	0.655	0.658	0.655	0.657	0.655

Table A.8: SVM Recall Scores for 'Software Engineer' Job Only Classification

α \ **T**	0	0.01	0.02	0.03	0.04	0.05
0	0.816	0.818	0.817	0.816	0.823	0.827
0.1	0.845	0.847	0.845	0.845	0.845	0.847
0.2	0.844	0.844	0.844	0.844	0.844	0.844
0.3	0.831	0.829	0.829	0.831	0.829	0.831
0.4	0.822	0.82	0.816	0.819	0.822	0.819
0.5	0.843	0.844	0.844	0.848	0.846	0.843
0.6	0.833	0.835	0.835	0.835	0.835	0.833
0.7	0.838	0.838	0.836	0.836	0.834	0.832
0.8	0.832	0.834	0.832	0.834	0.837	0.834
0.9	0.851	0.851	0.848	0.848	0.851	0.848
1	0.877	0.873	0.869	0.877	0.873	0.873

Table A.9: SVM F-scores for 'Software Engineer' Job Only Classification

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.706	0.706	0.706	0.706	0.709	0.708
0.1	0.721	0.722	0.721	0.721	0.721	0.721
0.2	0.721	0.72	0.72	0.72	0.72	0.72
0.3	0.737	0.737	0.737	0.737	0.737	0.737
0.4	0.715	0.714	0.712	0.713	0.715	0.713
0.5	0.728	0.729	0.728	0.729	0.729	0.727
0.6	0.733	0.734	0.734	0.733	0.734	0.733
0.7	0.721	0.72	0.72	0.718	0.719	0.719
0.8	0.745	0.745	0.744	0.745	0.744	0.745
0.9	0.717	0.717	0.715	0.715	0.717	0.715
1	0.751	0.749	0.749	0.75	0.75	0.749

Table A.10: Lucene Baseline Accuracy for Software Engineer' Job Only Classification

T / α	0
0	0.057
0.1	0.055
0.2	0.055
0.3	0.055
0.4	0.06
0.5	0.057
0.6	0.055
0.7	0.06
0.8	0.067
0.9	0.076
1	0.075

Table A.11: K Means Precision Scores for 'Research Assistant' Job Only Classification

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.262	0.617	0.617	0.617	0.617	0.262
0.1	0.627	0.629	0.629	0.629	0.629	0.264
0.2	0.61	0.275	0.275	0.61	0.61	0.61
0.3	0.613	0.282	0.613	0.282	0.282	0.613
0.4	0.614	0.614	0.288	0.288	0.288	0.614
0.5	0.621	0.619	0.265	0.619	0.265	0.266
0.6	0.64	0.64	0.64	0.64	0.261	0.261
0.7	0.661	0.266	0.663	0.266	0.267	0.66
0.8	0.273	0.65	0.273	0.273	0.273	0.65
0.9	0.266	0.663	0.663	0.266	0.266	0.663
1	0.5	0.5	0.5	0.222	0.675	0.675

Table A.12: K-means Recall Scores for 'Research Assistant' Job Only Classification

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.173	0.827	0.827	0.827	0.827	0.173
0.1	0.812	0.812	0.812	0.812	0.812	0.188
0.2	0.82	0.18	0.18	0.82	0.82	0.82
0.3	0.807	0.193	0.807	0.193	0.193	0.807
0.4	0.798	0.798	0.202	0.202	0.202	0.798
0.5	0.82	0.826	0.18	0.826	0.18	0.18
0.6	0.807	0.807	0.807	0.807	0.193	0.193
0.7	0.78	0.216	0.78	0.216	0.22	0.784
0.8	0.218	0.782	0.218	0.218	0.218	0.782
0.9	0.218	0.782	0.782	0.218	0.218	0.782
1	1	1	1	0.172	0.828	0.828

Table A.13: K Means F-Scores for 'Research Assistant' Job Only Classification

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.209	0.707	0.707	0.707	0.707	0.209
0.1	0.708	0.709	0.709	0.709	0.709	0.22
0.2	0.699	0.218	0.218	0.699	0.699	0.699
0.3	0.697	0.229	0.697	0.229	0.229	0.697
0.4	0.694	0.694	0.237	0.237	0.237	0.694
0.5	0.707	0.708	0.214	0.708	0.214	0.215
0.6	0.714	0.714	0.714	0.714	0.222	0.222
0.7	0.716	0.238	0.717	0.238	0.241	0.717
0.8	0.242	0.71	0.242	0.242	0.242	0.71
0.9	0.239	0.718	0.718	0.239	0.239	0.718
1	0.667	0.667	0.667	0.194	0.744	0.744

Table A.14: Random Forest Precision Scores for 'Research Assistant' Job Only Classification

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.72	0.708	0.703	0.703	0.704	0.706
0.1	0.708	0.718	0.718	0.72	0.716	0.73
0.2	0.691	0.7	0.687	0.693	0.692	0.69
0.3	0.717	0.728	0.733	0.727	0.729	0.733
0.4	0.718	0.701	0.697	0.708	0.713	0.711
0.5	0.692	0.713	0.694	0.709	0.709	0.729
0.6	0.735	0.714	0.716	0.697	0.7	0.704
0.7	0.736	0.731	0.753	0.735	0.751	0.747
0.8	0.749	0.749	0.764	0.755	0.749	0.763
0.9	0.701	0.701	0.739	0.733	0.723	0.713
1	0.72	0.717	0.748	0.746	0.735	0.72

Table A.15: Random Forest Recall Scores for 'Research Assistant' Job Only Classification

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.862	0.785	0.759	0.742	0.763	0.749
0.1	0.918	0.758	0.733	0.761	0.763	0.751
0.2	0.893	0.773	0.752	0.713	0.747	0.726
0.3	0.89	0.775	0.759	0.751	0.743	0.743
0.4	0.882	0.756	0.762	0.725	0.759	0.759
0.5	0.902	0.765	0.787	0.771	0.78	0.771
0.6	0.885	0.78	0.776	0.763	0.742	0.79
0.7	0.86	0.74	0.768	0.756	0.748	0.732
0.8	0.871	0.812	0.802	0.807	0.782	0.782
0.9	0.872	0.737	0.782	0.756	0.769	0.763
1	0.945	0.773	0.812	0.805	0.781	0.805

Table A.16: Random Forest F-scores for 'Research Assistant' Job Only Classification

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.785	0.744	0.73	0.722	0.733	0.727
0.1	0.8	0.738	0.725	0.74	0.739	0.74
0.2	0.779	0.734	0.718	0.703	0.719	0.707
0.3	0.794	0.751	0.746	0.739	0.736	0.738
0.4	0.791	0.728	0.728	0.716	0.735	0.734
0.5	0.783	0.738	0.737	0.739	0.743	0.75
0.6	0.803	0.746	0.745	0.728	0.72	0.744
0.7	0.793	0.736	0.76	0.746	0.749	0.739
0.8	0.805	0.779	0.783	0.78	0.765	0.773
0.9	0.777	0.719	0.76	0.744	0.745	0.737
1	0.818	0.744	0.779	0.774	0.758	0.76

Table A.17: SVM Precision Scores for 'Research Assistant' Job Only Classification

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.709	0.711	0.709	0.711	0.711	0.711
0.1	0.701	0.701	0.7	0.701	0.701	0.701
0.2	0.69	0.693	0.694	0.694	0.695	0.691
0.3	0.697	0.695	0.695	0.695	0.698	0.695
0.4	0.711	0.711	0.711	0.711	0.711	0.711
0.5	0.699	0.697	0.697	0.699	0.699	0.699
0.6	0.698	0.7	0.7	0.698	0.7	0.7
0.7	0.729	0.728	0.729	0.729	0.729	0.729
0.8	0.689	0.691	0.692	0.692	0.691	0.689
0.9	0.7	0.697	0.697	0.697	0.697	0.7
1	0.699	0.699	0.703	0.703	0.699	0.703

Table A.18: SVM Recall Scores for 'Research Assistant' Job Only Classification

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.895	0.895	0.895	0.892	0.895	0.892
0.1	0.918	0.918	0.918	0.918	0.918	0.918
0.2	0.901	0.901	0.901	0.901	0.898	0.901
0.3	0.906	0.903	0.906	0.906	0.903	0.903
0.4	0.891	0.891	0.891	0.891	0.891	0.891
0.5	0.915	0.912	0.912	0.915	0.915	0.915
0.6	0.925	0.925	0.925	0.925	0.925	0.925
0.7	0.884	0.88	0.884	0.884	0.884	0.884
0.8	0.911	0.906	0.901	0.901	0.906	0.911
0.9	0.942	0.942	0.942	0.942	0.942	0.942
1	0.945	0.945	0.945	0.945	0.945	0.945

Table A.19: SVM F-scores for 'Research Assistant' Job Only Classification

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.791	0.793	0.791	0.791	0.793	0.791
0.1	0.795	0.795	0.794	0.795	0.795	0.795
0.2	0.781	0.783	0.784	0.784	0.784	0.782
0.3	0.788	0.786	0.787	0.787	0.787	0.786
0.4	0.791	0.791	0.791	0.791	0.791	0.791
0.5	0.793	0.79	0.79	0.793	0.793	0.793
0.6	0.796	0.797	0.797	0.796	0.797	0.797
0.7	0.799	0.797	0.799	0.799	0.799	0.799
0.8	0.785	0.784	0.783	0.783	0.784	0.785
0.9	0.803	0.801	0.801	0.801	0.801	0.803
1	0.804	0.804	0.807	0.807	0.804	0.807

Table A.20: Lucene Accuracy Research Assistant

T / α	0	0.01	0.02	0.03	0.04	0.05	0.06
0	0.262	0.617	0	0.617	0.617	0.617	0.262
0.1	0.627	0.629	0	0.629	0.629	0.629	0.264
0.2	0.61	0.275	0	0.275	0.61	0.61	0.61
0.3	0.613	0.282	0	0.613	0.282	0.282	0.613
0.4	0.614	0.614	0	0.288	0.288	0.288	0.614
0.5	0.621	0.619	0	0.265	0.619	0.265	0.266
0.6	0.64	0.64	0	0.64	0.64	0.261	0.261
0.7	0.661	0.266	0	0.663	0.266	0.267	0.66
0.8	0.273	0.65	0	0.273	0.273	0.273	0.65
0.9	0.266	0.663	0	0.663	0.266	0.266	0.663
1	0.5	0.5	0	0.5	0.222	0.675	0.675

Table A.21: kMeansRecallResultRows.csv Research Assistant

α \ T	0	0.01	0.02	0.03	0.04	0.05	0.06
0	0.173	0.827	0	0.827	0.827	0.827	0.173
0.1	0.812	0.812	0	0.812	0.812	0.812	0.188
0.2	0.82	0.18	0	0.18	0.82	0.82	0.82
0.3	0.807	0.193	0	0.807	0.193	0.193	0.807
0.4	0.798	0.798	0	0.202	0.202	0.202	0.798
0.5	0.82	0.826	0	0.18	0.826	0.18	0.18
0.6	0.807	0.807	0	0.807	0.807	0.193	0.193
0.7	0.78	0.216	0	0.78	0.216	0.22	0.784
0.8	0.218	0.782	0	0.218	0.218	0.218	0.782
0.9	0.218	0.782	0	0.782	0.218	0.218	0.782
1	1	1	0	1	0.172	0.828	0.828

Table A.22: Lucene Baseline Accuracy for 'Research Assistant' Job Only Classification

α \ T	0
0	0.133
0.1	0.144
0.2	0.144
0.3	0.145
0.4	0.151
0.5	0.155
0.6	0.169
0.7	0.156
0.8	0.168
0.9	0.167
1	0.148

Table A.23: K-Means Precision Software Engineer & Google

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.561	0.561	0.561	0.468	0.468	0.561
0.1	0.519	0.519	0.479	0.519	0.519	0.479
0.2	0.48	0.514	0.514	0.48	0.48	0.514
0.3	0.507	0.5	0.5	0.158	0.566	0.5
0.4	0.515	0.529	0.488	0.539	0.539	0.466
0.5	0.524	0.532	0.508	0.531	0.462	0.462
0.6	0.483	0.516	0.483	0.483	0.483	0.483
0.7	0.462	0.541	0.548	0.185	0.548	0.456
0.8	0.513	0.517	0.5	0.5	0.554	0.547
0.9	0.556	0.556	0.556	0.439	0.511	0.429
1	0.549	0.436	0.111	0.111	0.111	0.549

Table A.24: K Means Recall Software Engineer & Google

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.385	0.385	0.385	0.615	0.615	0.385
0.1	0.548	0.548	0.452	0.548	0.548	0.452
0.2	0.397	0.603	0.603	0.397	0.397	0.603
0.3	0.619	0.398	0.602	0.051	0.949	0.398
0.4	0.455	0.5	0.545	0.5	0.5	0.5
0.5	0.99	0.584	0.99	0.594	0.416	0.416
0.6	0.462	0.538	0.462	0.462	0.462	0.462
0.7	0.474	0.526	0.526	0.066	0.526	0.474
0.8	0.984	0.984	0.984	0.984	0.59	0.574
0.9	0.625	0.625	0.625	0.375	0.979	0.375
1	0.975	0.425	0.025	0.025	0.025	0.975

Table A.25: K-Means FScore Software Engineer & Google

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.456	0.456	0.456	0.532	0.532	0.456
0.1	0.533	0.533	0.465	0.533	0.533	0.465
0.2	0.434	0.555	0.555	0.434	0.434	0.555
0.3	0.557	0.443	0.546	0.077	0.709	0.443
0.4	0.483	0.514	0.515	0.519	0.519	0.482
0.5	0.685	0.557	0.671	0.561	0.438	0.438
0.6	0.472	0.527	0.472	0.472	0.472	0.472
0.7	0.468	0.533	0.537	0.097	0.537	0.465
0.8	0.674	0.678	0.663	0.663	0.571	0.56
0.9	0.588	0.588	0.588	0.404	0.671	0.4
1	0.703	0.43	0.041	0.041	0.041	0.703

Table A.26: Random Forest Precision Software Engineer & Google

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.693	0.697	0.688	0.712	0.701	0.71
0.1	0.67	0.648	0.664	0.667	0.658	0.664
0.2	0.714	0.637	0.611	0.614	0.66	0.632
0.3	0.601	0.649	0.652	0.645	0.653	0.652
0.4	0.627	0.655	0.657	0.65	0.714	0.676
0.5	0.625	0.718	0.726	0.745	0.731	0.736
0.6	0.636	0.633	0.624	0.651	0.633	0.659
0.7	0.667	0.677	0.683	0.7	0.712	0.7
0.8	0.618	0.672	0.694	0.731	0.771	0.809
0.9	0.702	0.698	0.69	0.674	0.651	0.69
1	0.762	0.667	0.744	0.676	0.706	0.788

Table A.27: Random Forest Recall Software Engineer & Google

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.664	0.643	0.601	0.587	0.622	0.615
0.1	0.621	0.653	0.605	0.629	0.605	0.621
0.2	0.661	0.595	0.57	0.579	0.579	0.595
0.3	0.805	0.737	0.636	0.585	0.559	0.619
0.4	0.809	0.673	0.627	0.609	0.682	0.645
0.5	0.743	0.733	0.683	0.723	0.673	0.663
0.6	0.692	0.626	0.637	0.615	0.626	0.593
0.7	0.711	0.553	0.566	0.645	0.553	0.553
0.8	0.902	0.705	0.557	0.623	0.607	0.623
0.9	0.688	0.625	0.604	0.604	0.583	0.604
1	0.8	0.75	0.725	0.625	0.6	0.65

Table A.28: Random Forest FScore Software Engineer & Google

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.679	0.669	0.642	0.644	0.659	0.659
0.1	0.644	0.651	0.633	0.647	0.63	0.642
0.2	0.687	0.615	0.59	0.596	0.617	0.613
0.3	0.688	0.69	0.644	0.613	0.603	0.635
0.4	0.706	0.664	0.642	0.629	0.698	0.66
0.5	0.679	0.725	0.704	0.734	0.701	0.698
0.6	0.663	0.63	0.63	0.633	0.63	0.624
0.7	0.688	0.609	0.619	0.671	0.622	0.618
0.8	0.733	0.688	0.618	0.673	0.679	0.704
0.9	0.695	0.659	0.644	0.637	0.615	0.644
1	0.78	0.706	0.734	0.649	0.649	0.712

Table A.29: SVM Precision Software Engineer & Google

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.614	0.611	0.614	0.611	0.613	0.614
0.1	0.601	0.601	0.601	0.601	0.598	0.601
0.2	0.606	0.606	0.609	0.609	0.605	0.605
0.3	0.603	0.603	0.603	0.607	0.603	0.603
0.4	0.612	0.615	0.612	0.611	0.612	0.615
0.5	0.629	0.629	0.632	0.627	0.629	0.632
0.6	0.631	0.631	0.631	0.631	0.631	0.631
0.7	0.61	0.624	0.61	0.615	0.61	0.615
0.8	0.617	0.602	0.595	0.602	0.602	0.602
0.9	0.639	0.639	0.639	0.639	0.639	0.627
1	0.623	0.623	0.623	0.623	0.623	0.623

Table A.30: SVM Recall Software Engineer & Google

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.79	0.79	0.79	0.79	0.797	0.79
0.1	0.79	0.79	0.79	0.79	0.79	0.79
0.2	0.802	0.802	0.81	0.81	0.81	0.81
0.3	0.89	0.89	0.89	0.89	0.89	0.89
0.4	0.818	0.827	0.818	0.827	0.818	0.827
0.5	0.772	0.772	0.782	0.782	0.772	0.782
0.6	0.769	0.769	0.769	0.769	0.769	0.769
0.7	0.842	0.829	0.842	0.842	0.842	0.842
0.8	0.82	0.82	0.82	0.82	0.82	0.82
0.9	0.812	0.812	0.812	0.812	0.812	0.771
1	0.95	0.95	0.95	0.95	0.95	0.95

Table A.31: SVM FScore Software Engineer & Google

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.691	0.689	0.691	0.689	0.693	0.691
0.1	0.683	0.683	0.683	0.683	0.681	0.683
0.2	0.69	0.69	0.695	0.695	0.693	0.693
0.3	0.719	0.719	0.719	0.722	0.719	0.719
0.4	0.7	0.705	0.7	0.703	0.7	0.705
0.5	0.693	0.693	0.699	0.696	0.693	0.699
0.6	0.693	0.693	0.693	0.693	0.693	0.693
0.7	0.707	0.712	0.707	0.711	0.707	0.711
0.8	0.704	0.694	0.69	0.694	0.694	0.694
0.9	0.716	0.716	0.716	0.716	0.716	0.692
1	0.752	0.752	0.752	0.752	0.752	0.752

Table A.32: K-means Precision Scores Intel Corporation & Component Design Engineer

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.4	0.562	0.4	0.379	0.571	0.4
0.1	0.535	0.579	0.409	0.545	0.1	0.579
0.2	0.625	0.615	0.333	0.625	0.615	0.593
0.3	0.45	0.523	0.561	0.45	0.45	0.45
0.4	0.381	0.564	0.579	0.564	0.564	0.381
0.5	0.5	1	0.417	0.417	0.525	0.5
0.6	0.559	0.545	0.385	0.385	0.385	0.333
0.7	0.684	0.607	0.125	0.7	0.7	0.125
0.8	0.545	0.545	0.545	0.455	0.545	0.545
0.9	0.5	0.5	0.5	0.5	0.5	0.533
1	0.5	0.5	0.5	0.5	0.5	0.5

Table A.33: K-means Recall Scores Intel Corporation & Component Design Engineer

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.308	0.692	0.308	0.282	0.718	0.308
0.1	0.697	1	0.273	0.727	0.03	1
0.2	0.758	0.727	0.273	0.758	0.727	0.97
0.3	0.281	0.719	1	0.281	0.281	0.281
0.4	0.267	0.733	0.733	0.733	0.733	0.267
0.5	1	0	0.192	0.192	0.808	1
0.6	0.826	0.783	0.217	0.217	0.217	0.174
0.7	0.722	0.944	0.056	0.778	0.778	0.056
0.8	0.545	0.545	0.545	0.455	0.545	0.545
0.9	0.625	0.375	0.625	0.375	1	0.415
1	0.571	0.429	0.571	0.429	0.429	0.571

Table A.34: K-means F-scores Intel Corporation & Component Design Engineer

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.348	0.621	0.348	0.324	0.636	0.348
0.1	0.605	0.733	0.327	0.623	0.047	0.733
0.2	0.685	0.667	0.3	0.685	0.667	0.736
0.3	0.346	0.605	0.719	0.346	0.346	0.346
0.4	0.314	0.638	0.647	0.638	0.638	0.314
0.5	0.667	0	0.263	0.263	0.636	0.667
0.6	0.667	0.643	0.278	0.278	0.278	0.229
0.7	0.703	0.739	0.077	0.737	0.737	0.077
0.8	0.545	0.545	0.545	0.455	0.545	0.545
0.9	NaN	0.556	0.429	0.556	0.429	0.696
1	0.533	0.462	0.533	0.462	0.462	0.533

Table A.35: Random Forest Precision Intel Corporation & Component Design Engineer

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.625	0.64	0.698	0.727	0.697	0.7
0.1	0.659	0.719	0.733	0.724	0.774	0.778
0.2	0.676	0.7	0.667	0.714	0.724	0.7
0.3	0.609	0.581	0.595	0.61	0.581	0.595
0.4	0.632	0.618	0.606	0.655	0.625	0.576
0.5	0.625	0.708	0.633	0.607	0.654	0.654
0.6	0.633	0.615	0.625	0.615	0.652	0.615
0.7	0.765	0.8	0.75	0.786	0.786	0.786
0.8	0.8	0.889	1	0.9	1	0.9
0.9	0.556	0.556	0.625	0.714	0.625	0.714
1	0.75	0.75	0.75	0.75	0.75	0.75

Table A.36: Random Forest Recall Results Intel Corporation & Component Design Engineer

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.897	0.821	0.769	0.821	0.59	0.718
0.1	0.818	0.697	0.667	0.636	0.727	0.636
0.2	0.758	0.636	0.606	0.606	0.636	0.636
0.3	0.875	0.781	0.781	0.781	0.781	0.781
0.4	0.8	0.7	0.667	0.633	0.667	0.633
0.5	0.769	0.654	0.731	0.654	0.654	0.654
0.6	0.826	0.696	0.652	0.696	0.652	0.696
0.7	0.722	0.667	0.667	0.611	0.611	0.611
0.8	0.727	0.727	0.636	0.818	0.909	0.818
0.9	0.625	0.625	0.625	0.625	0.625	0.625
1	0.857	0.857	0.857	0.857	0.857	0.857

Table A.37: Random Forest F-Scores Intel Corporation & Component Design Engineer

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.737	0.719	0.732	0.771	0.639	0.709
0.1	0.73	0.708	0.698	0.677	0.75	0.7
0.2	0.714	0.667	0.635	0.656	0.677	0.667
0.3	0.718	0.667	0.676	0.685	0.667	0.676
0.4	0.706	0.656	0.635	0.644	0.645	0.603
0.5	0.69	0.68	0.679	0.63	0.654	0.654
0.6	0.717	0.653	0.638	0.653	0.652	0.653
0.7	0.743	0.727	0.706	0.688	0.688	0.688
0.8	0.762	0.8	0.778	0.857	0.952	0.857
0.9	0.588	0.588	0.625	0.667	0.625	0.667
1	0.8	0.8	0.8	0.8	0.8	0.8

Table A.38: SVM Precision Scores Intel Corporation & Component Design Engineer

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.63	0.636	0.636	0.636	0.636	0.636
0.1	0.593	0.571	0.593	0.571	0.593	0.593
0.2	0.638	0.638	0.638	0.646	0.638	0.63
0.3	0.58	0.58	0.58	0.596	0.58	0.596
0.4	0.614	0.614	0.614	0.614	0.614	0.614
0.5	0.622	0.622	0.622	0.622	0.622	0.622
0.6	0.571	0.571	0.571	0.571	0.571	0.571
0.7	0.654	0.654	0.654	0.654	0.654	0.654
0.8	0.6	0.6	0.6	0.6	0.6	0.6
0.9	0.533	0.533	0.667	0.533	0.533	0.533
1	0.667	0.667	0.667	0.667	0.667	0.667

Table A.39: SVM Recall Scores Intel Corporation & Component Design Engineer

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.872	0.897	0.897	0.897	0.897	0.897
0.1	0.97	0.97	0.97	0.97	0.97	0.97
0.2	0.909	0.909	0.909	0.939	0.909	0.879
0.3	0.906	0.906	0.906	0.875	0.906	0.875
0.4	0.9	0.9	0.9	0.9	0.9	0.9
0.5	0.885	0.885	0.885	0.885	0.885	0.885
0.6	0.87	0.87	0.87	0.87	0.87	0.87
0.7	0.944	0.944	0.944	0.944	0.944	0.944
0.8	0.545	0.545	0.545	0.545	0.545	0.545
0.9	1	1	0.75	1	1	1
1	0.857	0.857	0.857	0.857	0.857	0.857

Table A.40: SVM F-score Intel Corporation & Component Design Engineer

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.731	0.745	0.745	0.745	0.745	0.745
0.1	0.736	0.719	0.736	0.719	0.736	0.736
0.2	0.75	0.75	0.75	0.765	0.75	0.734
0.3	0.707	0.707	0.707	0.709	0.707	0.709
0.4	0.73	0.73	0.73	0.73	0.73	0.73
0.5	0.73	0.73	0.73	0.73	0.73	0.73
0.6	0.69	0.69	0.69	0.69	0.69	0.69
0.7	0.773	0.773	0.773	0.773	0.773	0.773
0.8	0.571	0.571	0.571	0.571	0.571	0.571
0.9	0.696	0.696	0.706	0.696	0.696	0.696
1	0.75	0.75	0.75	0.75	0.75	0.75

Table A.41: K-means Precision Scores for Intel Corporation & Component Design Engineer

α \ **T**	0	0.01	0.02	0.03	0.04	0.05
0	0.401	0.564	0.401	0.564	0.401	0.564
0.1	0.578	0.578	0.578	0.578	0.389	0.578
0.2	0.572	0.572	0.572	0.572	0.572	0.572
0.3	0.372	0.594	0.372	0.594	0.594	0.594
0.4	0.422	0.422	0.558	0.558	0.557	0.42
0.5	0.392	0.574	0.574	0.392	0.393	0.393
0.6	0.588	0.588	0.376	0.588	0.588	0.588
0.7	0.594	0.375	0.375	0.594	0.594	0.375
0.8	0.379	0.605	0.379	0.379	0.379	0.379
0.9	0.576	0.576	0.406	0.576	0.576	0.406
1	0.594	0.392	0.6	0.594	0.392	0.386

Table A.42: K-means Recall Scores for Intel Corporation & Component Design Engineer

α \ **T**	0	0.01	0.02	0.03	0.04	0.05
0	0.317	0.683	0.317	0.683	0.317	0.683
0.1	0.679	0.679	0.679	0.679	0.321	0.679
0.2	0.699	0.699	0.699	0.699	0.699	0.699
0.3	0.315	0.685	0.315	0.685	0.685	0.685
0.4	0.356	0.356	0.647	0.647	0.644	0.353
0.5	0.319	0.681	0.681	0.319	0.321	0.321
0.6	0.689	0.687	0.313	0.689	0.689	0.689
0.7	0.677	0.323	0.323	0.677	0.677	0.323
0.8	0.352	0.648	0.352	0.352	0.352	0.352
0.9	0.635	0.635	0.365	0.635	0.635	0.365
1	0.634	0.366	0.638	0.634	0.366	0.362

Table A.43: K-means F-scores for Intel Corporation & Component Design Engineer

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.354	0.618	0.354	0.618	0.354	0.618
0.1	0.624	0.624	0.624	0.624	0.352	0.624
0.2	0.629	0.629	0.629	0.629	0.629	0.629
0.3	0.341	0.636	0.341	0.636	0.636	0.636
0.4	0.386	0.386	0.599	0.599	0.598	0.384
0.5	0.352	0.623	0.623	0.352	0.354	0.354
0.6	0.634	0.634	0.342	0.634	0.634	0.634
0.7	0.633	0.347	0.347	0.633	0.633	0.347
0.8	0.365	0.626	0.365	0.365	0.365	0.365
0.9	0.604	0.604	0.385	0.604	0.604	0.385
1	0.614	0.378	0.618	0.614	0.378	0.374

Table A.44: Random Forest Precision Scores for Intel Corporation & Component Design Engineer

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.636	0.636	0.644	0.647	0.64	0.635
0.1	0.63	0.626	0.63	0.637	0.636	0.655
0.2	0.631	0.63	0.635	0.631	0.623	0.644
0.3	0.658	0.664	0.671	0.658	0.675	0.682
0.4	0.618	0.648	0.647	0.643	0.643	0.65
0.5	0.665	0.654	0.65	0.645	0.659	0.673
0.6	0.667	0.657	0.681	0.673	0.689	0.681
0.7	0.64	0.645	0.661	0.665	0.675	0.68
0.8	0.683	0.667	0.652	0.658	0.668	0.669
0.9	0.654	0.672	0.676	0.686	0.678	0.695
1	0.69	0.681	0.682	0.702	0.693	0.714

Table A.45: Random Forest Recall Scores for Intel Corporation & Component Design Engineer

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.722	0.717	0.691	0.684	0.676	0.653
0.1	0.834	0.667	0.634	0.648	0.657	0.65
0.2	0.836	0.681	0.68	0.66	0.641	0.654
0.3	0.842	0.667	0.651	0.645	0.657	0.654
0.4	0.854	0.701	0.68	0.671	0.691	0.698
0.5	0.721	0.682	0.676	0.661	0.704	0.691
0.6	0.775	0.695	0.685	0.673	0.685	0.669
0.7	0.813	0.685	0.709	0.685	0.709	0.687
0.8	0.738	0.718	0.679	0.692	0.65	0.64
0.9	0.787	0.756	0.737	0.749	0.702	0.679
1	0.787	0.765	0.754	0.765	0.724	0.728

Table A.46: Random Forest F-scores for Intel Corporation & Component Design Engineer

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.676	0.674	0.667	0.665	0.657	0.644
0.1	0.718	0.646	0.632	0.643	0.646	0.653
0.2	0.719	0.655	0.657	0.645	0.632	0.649
0.3	0.739	0.666	0.661	0.651	0.666	0.668
0.4	0.717	0.673	0.663	0.656	0.666	0.673
0.5	0.692	0.668	0.662	0.653	0.681	0.682
0.6	0.717	0.676	0.683	0.673	0.687	0.675
0.7	0.716	0.665	0.684	0.675	0.692	0.684
0.8	0.71	0.692	0.665	0.674	0.659	0.654
0.9	0.715	0.712	0.705	0.716	0.69	0.687
1	0.735	0.721	0.716	0.732	0.708	0.721

Table A.47: SVM Precision Scores for Intel Corporation & Component Design Engineer

α \ T	0	0.01	0.02	0.03	0.04	0.05
0	0.622	0.621	0.621	0.622	0.623	0.62
0.1	0.628	0.629	0.628	0.629	0.629	0.628
0.2	0.629	0.628	0.628	0.628	0.628	0.628
0.3	0.662	0.663	0.663	0.662	0.663	0.663
0.4	0.633	0.632	0.631	0.632	0.633	0.632
0.5	0.64	0.641	0.64	0.64	0.641	0.64
0.6	0.655	0.655	0.655	0.654	0.656	0.655
0.7	0.632	0.631	0.632	0.63	0.632	0.633
0.8	0.674	0.674	0.673	0.674	0.67	0.674
0.9	0.619	0.619	0.618	0.618	0.619	0.618
1	0.656	0.655	0.658	0.655	0.657	0.655

Table A.48: SVM Recall Scores for Intel Corporation & Component Design Engineer

α \ T	0	0.01	0.02	0.03	0.04	0.05
0	0.816	0.818	0.817	0.816	0.823	0.827
0.1	0.845	0.847	0.845	0.845	0.845	0.847
0.2	0.844	0.844	0.844	0.844	0.844	0.844
0.3	0.831	0.829	0.829	0.831	0.829	0.831
0.4	0.822	0.82	0.816	0.819	0.822	0.819
0.5	0.843	0.844	0.844	0.848	0.846	0.843
0.6	0.833	0.835	0.835	0.835	0.835	0.833
0.7	0.838	0.838	0.836	0.836	0.834	0.832
0.8	0.832	0.834	0.832	0.834	0.837	0.834
0.9	0.851	0.851	0.848	0.848	0.851	0.848
1	0.877	0.873	0.869	0.877	0.873	0.873

Table A.49: SVM F-scores Intel Corporation & Component Design Engineer

T / α	0	0.01	0.02	0.03	0.04	0.05
0	0.706	0.706	0.706	0.706	0.709	0.708
0.1	0.721	0.722	0.721	0.721	0.721	0.721
0.2	0.721	0.72	0.72	0.72	0.72	0.72
0.3	0.737	0.737	0.737	0.737	0.737	0.737
0.4	0.715	0.714	0.712	0.713	0.715	0.713
0.5	0.728	0.729	0.728	0.729	0.729	0.727
0.6	0.733	0.734	0.734	0.733	0.734	0.733
0.7	0.721	0.72	0.72	0.718	0.719	0.719
0.8	0.745	0.745	0.744	0.745	0.744	0.745
0.9	0.717	0.717	0.715	0.715	0.717	0.715
1	0.751	0.749	0.749	0.75	0.75	0.749

Bibliography

Alpaydin, Ethem (2014). *Introduction to machine learning.* MIT press.

Ashton, Chris and Lynne Morton (2005). “Managing talent for competitive advantage: Taking a systemic approach to talent management”. In: *Strategic HR Review* 4.5, pp. 28–31.

Becker, Brian E and Mark A Huselid (2006). “Strategic human resources management: where do we go from here?” In: *Journal of management* 32.6, pp. 898–925.

Cappelli, Peter (2008). “Talent management for the twenty-first century”. In: *Harvard business review* 86.3, p. 74.

Collings, David G and Kamel Mellahi (2009). “Strategic talent management: A review and research agenda”. In: *Human resource management review* 19.4, pp. 304–313.

Friedman, Jerome, Trevor Hastie, and Robert Tibshirani (2001). *The elements of statistical learning.* Vol. 1. Springer series in statistics Springer, Berlin.

Hatcher, Erik and Otis Gospodnetic (2004). “Lucene in action”. In:

Huselid, Mark A, Richard W Beatty, and Brian E Becker (2005). “‘A players’ or ‘A positions’?” In: *harvard business review* 83.12, pp. 110–117.

Jantan, Hamidah, Abdul Razak Hamdan, and Zulaiha Ali Othman (2009). “Classification for talent management using decision tree induction techniques”. In: *Data Mining and Optimization, 2009. DMO’09. 2nd Conference on.* IEEE, pp. 15–20.

Jantan, Hamidah, Norazmah Mat Yusoff, and Mohamad Rozuan Noh. “Towards Applying Support Vector Machine Algorithm in Employee Achievement Classification”. In:

Kotsiantis, Sotiris B, I Zaharakis, and P Pintelas (2007). *Supervised machine learning: A review of classification techniques.*

Lops, Pasquale et al. (2011). "Leveraging the linkedin social network data for extracting content-based user profiles". In: *Proceedings of the fifth ACM conference on Recommender systems*. ACM, pp. 293–296.

Lynne, M (2005). "Talent Management Value Imperatives: Strategies for Execution". In: *The Conference Board*.

Mäkelä, Kristiina, Ingmar Björkman, and Mats Ehrnrooth (2010). "How do MNCs establish their talent pools? Influences on individuals' likelihood of being labeled as talent". In: *Journal of World Business* 45.2, pp. 134–142.

Manning, Christopher D, Prabhakar Raghavan, Hinrich Schütze, et al. (2008). *Introduction to information retrieval*. Vol. 1. 1. Cambridge university press Cambridge.

Michaels, Ed, Helen Handfield-Jones, and Beth Axelrod (2001). *The war for talent*. Harvard Business Press.

Nathan, Adam (2006). *Windows presentation foundation unleashed*. Pearson Education.

Nishii, Lisa H, David P Lepak, and Benjamin Schneider (2008). "Employee attributions of the "why" of HR practices: Their effects on employee attitudes and behaviors, and customer satisfaction". In: *Personnel psychology* 61.3, pp. 503–545.

Özsoy, Salih, Gökhan Gümüş, and Savriddin KHALILOV (2015). "C4. 5 Versus Other Decision Trees: A Review". In: *Computer Engineering and Applications Journal* 4.3, pp. 173–182.

Segal, Mark R (2004). "Machine learning benchmarks and random forest regression". In: *Center for Bioinformatics & Molecular Biostatistics*.

Sparrow, Paul R (2007). "Globalization of HR at function level: four UK-based case studies of the international recruitment and selection process". In: *The International Journal of Human Resource Management* 18.5, pp. 845–867.