TÉCNICO LISBOA

Applying Machine Learning Techniques to Implement a Decision Support Mechanism for Human Resources Recruitment

Henrique José Trindade Delgado

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Prof. Dr. Nuno Cavaco Gomes Horta

Eng. João Vieira da Luz

Examination Committee:

Chairperson: Prof. Dra. Teresa Maria Sá Ferreira Vazão Vasques

Supervisor: Prof. Dr. Nuno Cavaco Gomes Horta

Members of the Committee: Prof. Dr. João Nuno de Oliveira e Silva

November 2019

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the University of Lisbon.

Acknowledgments

I would like to firstly thank the collaboration and support given by my supervisors, Professor Nuno Horta, Pedro Caetano and João Luz who provided the knowledge and tools I needed to give my 100% on this work.

Secondly, I would like to thank Polarising, my co-workers and friends for the amazing environment and support I have every day, and for the opportunity to work on this interesting topic.

Thirdly, I would like to thank the Integrated Circuits group from Instituto de Telecomunicações for all the support given during the development of this work.

Lastly, I would like to thank my family, friends, Anna and my cat for all the love and support given throughout this journey, you really made me proud for what I accomplished so far.

Abstract

This work describes a new approach based on machine learning techniques to automatically evaluate candidates to open positions in IT companies and its integration on the recruitment processes. This approach comprises four main layers: a processing layer, a clustering layer, a classification layer and an output layer. The processing layer receives CVs, extracting their features in order to be used in the latter layers. The clustering layer studies the underlying structure of the company while the classification layer is responsible for the evaluation of an incoming candidate, relying on the structured data obtained from the previous layer. The first implements the K-Means clustering algorithm while the latter implements the KNN classifier. At last, the final layer generates several outputs displayed in a web application, describing the evaluated candidate and his/her adequacy for the company. This work shows promising results regarding its usefulness as a tool to guide the recruitment process, standing out from similar systems by the deeper analysis performed on each candidate and simpler user interfaces.

Keywords: Clustering, Classification, Decision-Support, Machine Learning, Recruitment Process, Web Application

Resumo

Este trabalho descreve uma nova abordagem baseada em técnicas de machine learning na avaliação de candidatos para cargos de trabalho em empresas de IT e a sua integração nos processos de recrutamento. Esta abordagem é constituída por quatro camadas principais: camada de processamento, camada de agrupamento, camada de classificação e camada de output. A camada de processamento recebe CVs, extraindo os atributos dos mesmos de modo a serem usados nas camadas seguintes. A camada de agrupamento estuda a estrutura da empresa enquanto que a camada de classificação é responsável pela avaliação de um novo candidato, dependendo dos dados obtidos na camada anterior. A primeira implementa o algoritmo K-Means enquanto que a última implementa o classificador KNN. Por fim, a última camada gera vários outputs representados numa aplicação web, descrevendo o candidato e o quão adequado é para a empresa a que se candidata. Este trabalho revela resultados promissores em relação à sua utilidade como ferramenta de guia do processo de recrutamento, destacando-se de sistemas semelhantes por uma análise mais aprofundada sobre cada candidato e pelas simples interfaces de utilizador apresentadas.

Palavras-Chave: Agrupamento, Aprendizagem Automática, Aplicação Web, Classificação, Processo de Recrutamento, Suporte de Decisão

Table of Contents

List of Figures

List of Tables

List of Acronyms

HR	Human Resources
HRM	Human Resource Management
SHRM	Strategic Human Resource Management
CV	Curriculum Vitae
GPA	Grade Point Average
IT	Information Technology
CBR	Cluster Based Ranking
KNN	K Nearest Neighbors
LDA	Latent Dirichlet Allocation
SVM	Support-Vector Machine
LR	Linear Regression
SVR	Support-Vector Regression
PUK	Pearson Universal Kernel
API	Application Programming Interface
IoC	Inversion of Control
POJO	Plain Old Java Object
NB	Naïve Bayes
CLI	Command Line Interface
SSE	Sum of Squared Errors
WCSS	Within Cluster Sum of Squared Errors

Chapter 1

Introduction

In this first chapter, a brief overview of the history of human resource management and the recruitment process is performed. Afterwards, this thesis' motivation and objectives are defined and explained. At last, is described this work's layout and structure.

1.1 Human Resource Management Overview

Human Resource Management development can be characterized into four different stages [1]:

- Stage one (1900s-1940s): administration stage
- Stage two (1940s-mid 1970s): welfare and administration stage
- Stage three (mid-1970s-late 1990s): human resource management (HRM) and strategic human resource management (SHRM) stage
- Stage four (Beyond 2000): SHRM into the future

In **stage one**, personnel functions were only performed by line managers, supervisors and early specialists. This period is long before the establishment of any national association representing a profession of personnel or human resource management. During this stage, job design, structured reward systems and personnel management practices were improved, specifically in the recruitment and placement of skilled employees. Furthermore, behavioral science added psychological testing and motivational systems, while management science supported and contributed to performance management programs. Regarding the social environment, society was generally stable with low unemployment rates until the World War I and the Great Depression (1930s).

Stage two marked the beginning of a more specialized and professional approach to personnel management. The consequences and repercussions of World War II were very significant, especially for the Business and Economy sectors, and for the labor market. Since many men were in the military

service, there were not enough workers and women started becoming more involved in all areas of the industry. In consequence of financial, social and family pressures, employees' productivity decreased significantly, and they become harder to recruit. When the war ended, returning soldiers flooded the labor market, often with few work skills. In order to increase productivity, attract and maintain employees, the welfare services' provision increased.

In **stage three**, during the 1970s, the business and economic environments were very turbulent and competitive between australian, american, european and the emerging asian markets. During this stage, personnel management was evolving towards human resource management, acknowledging employees as human resources who are crucial organizational assets. Employees possess knowledge, skills and potential, for those reasons, they require integrated management strategies. For example, performance management and rewards programs assure their individual and collective contributions to the achievement of organizational goals and objectives.

At last, **stage four** represents the present and future of Human Resource Management.

Although predicting the nature of HRM in the next years is difficult, there are strong indications that its theory and practice will be continually transformed as a consequence of globalization and new technologies being developed every day, changing the nature of work and jobs.

That being said, in order to qualify companies to follow the fast-growing pace in HRM and stay competitive, there must be a significant investment in the integration of technology, alongside with a more strategic and future-driven approach.

1.2 Recruitment Process Overview

The candidate's selection process and human recruitment (Figure 1.1) has been slowly changing but there is still huge potential and room for improvement.

Figure 1.1 - Evolution of the modern recruiter [2]

Until **1900s**, all jobs were advertised in newspapers [2], however, newspapers are still a place today for job offering and seeking. A few decades after, around **1940s**, due to the limited resources available, recruiters had a hard job hunting down the best person for each job position, relying mainly on personal connections.

From the **1980s** until the **1990s**, personal computers and the internet appeared and started to be more and more integrated in the recruitment process: Resumes databases, online job offerings and applicant tracking systems. This increased the efficiency of the recruitment process and significantly broadened the candidate pool.

From the **2000s** until today, mostly everything is done online. The whole recruitment process has been digitalized, focusing on social media, company career sites, mobile apps and online job boards.

1.3 Motivation

In the past, the lack of resources and information was the main barrier between the candidates and the recruiters. Nowadays, the vast amount of information available is the issue.

For the past decades, we have witnessed that people are choosing the web as their preferred way for job seeking. In consequence, companies are being increasingly overloaded with resumes and applications. Most recruitment processes still rely on personal interviews and human CV (Curriculum Vitae) processing, which have a cost and take time. Choosing to tackle this problem by allocating human resources to process each candidate's application is no longer a viable option if a company wants to be competitive in recruiting the best candidates.

Although companies are striving to innovate the recruitment process, in most of them, this is still a big issue. Different components of the recruitment process are still loosely coupled, having little to no knowledge of the definitions of other separate components. The recruiter's job has been gluing all the system together.

Companies need to start integrating more technology in this department in order to implement a more deterministic and automated recruitment process, instead of an empirical one.

1.4 Purpose and Objectives

The purpose of this thesis is to study the advantages of applying machine learning techniques to the recruitment process of a company, enabling human resources to find the best and most adequate candidates in the shortest period, allocating the least amount of human resources possible.

This thesis implements a decision-support system applied to the recruitment process at Polarising [18], a Portuguese consultancy company. The software is projected to process a dataset of approximately 150 collaborators' CVs and, based on the information obtained, be able to synchronously find the best candidates, and how they compare themselves with the rest of the company. Besides the processing component of the proposed system (back-end), the system will provide a web application (front-end), displaying useful information for the recruiter.

Regarding the implementation aspect of this thesis, Polarising's main technologies, such as the Spring framework for backend development (using Java language) and Angular for the frontend development (using TypeScript, CSS and HTML) will be used as the system's implementation driving forces.

The focus of the machine learning techniques will be, in an initial stage, understanding the underlying structure and patterns of the company (e.g. how many groups can be identified in Polarising and which characteristics define them) and, afterwards, compare the incoming candidates to the existing personnel distribution (e.g. in which group a candidate is more adequate). The study of Polarising's workforce distribution is as important as the evaluation of incoming candidates because it may identify where the company lacks resources, guiding the recruitment process in order to strengthen those areas.

The system aims at greatly innovating the recruitment process but not to replacing human analysis. The outcome of this work intends to implement a decision-support system for human resources, improving the recruiters' everyday job.

Although the proposed software is applied to the human resources at Polarising, this thesis aims at giving an overall picture of the best solutions to this problem that can be applied to all companies, independently of their dimension and dataset size.

1.5 Thesis Outline

This thesis can be divided into six chapters:

- **Chapter 1** – Introduction - brief overview of the evolution of human resources management and the recruitment process.
- **Chapter 2** – State-of-the-art - related works are showcased regarding their inputs, outputs and system approaches.
- **Chapter 3** – System Architecture - details the system's architecture, design decisions and interaction between external applications.
- **Chapter 4** – System Implementation - describes the technologies used based on the identified requirements and specifications, detailing the implementation of each system component.
- **Chapter 5** – System Validation - describes the metrics used, alongside with several case studies evaluating each component of the system.
- **Chapter 6** – Conclusions and Future Work - Summarizes all the work done, methodologies used and conclusions regarding the results obtained, together with tweaks and ideas to improve the implemented system.

Chapter 2

State-of-the-art

This section describes some of the previous work made in this topic. The section is divided in system inputs and pre-processing, system outputs and system approaches (Figure 2.1). In the inputs and pre-processing's subsection is described the multiple inputs that different systems consider and how they manage to organize the information. The outputs' section describes the most common outputs that current solutions give. At last, the system approaches' section describes how state-of-the-art systems are implemented and how they solve the problem of the candidate selection. The reference selection criteria was based on previous works that relate the most to the problem in hand, trying to demonstrate all the possible strategies used in current systems.

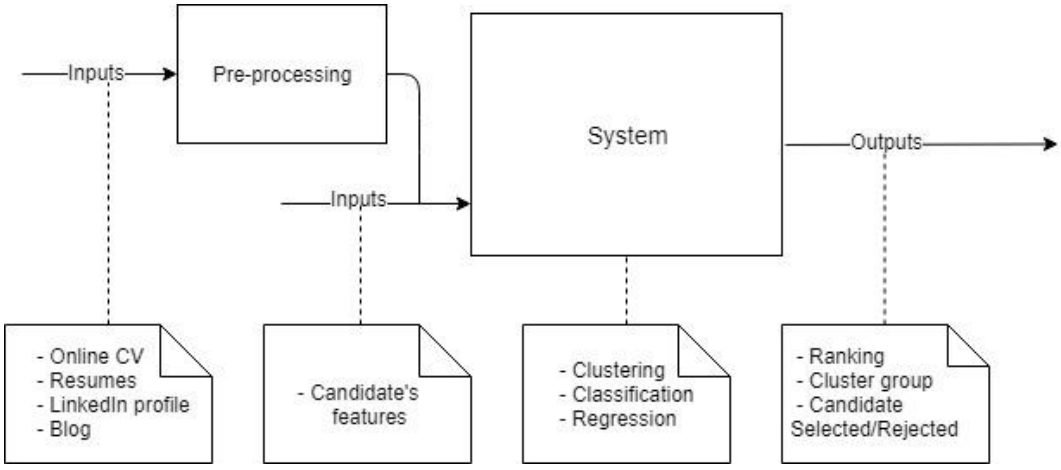


Figure 2.1 - State-of-the-art systems' structure

2.1 Inputs and Pre-Processing

Since the problem described is related to companies' recruitment processes, in order to characterize each candidate, we expect most previous works in this topic consider resumes/CV related information

to be inputs of their systems. However, previous works also try to integrate other metrics in order to perform a deeper analysis on each candidate.

Some systems consider the candidate's features, attributes that characterize each candidate, as inputs of their systems, and some consider inputs that rely on a pre-processing task before entering the main processing component.

2.1.1 Direct Inputs

References [3] and [7] consider candidate's features as inputs. Reference [3] uses a training data set gathered from a software firm considering the following parameters: Grade point average (GPA) [17] in C course, GPA in Java, Algorithm, Data Structure, Project work, Thesis work, Total cumulative grade point average (CGPA), Extracurricular activities, Database management and System design. For each attribute, a rank is given: 1 – Very poor knowledge, 2 – Poor knowledge, 3 – Medium knowledge, 4 – High knowledge. From these features, a classification process is performed.

In [7], the system's input variables are performance factors for selected attributes of a candidate. The attributes and respective values are shown in the following table:

Table 2.1 - Performance factors for selected attributes [7]

Attribute	Description
Category	P – Professional, S – Support Staff
Gender	Male and Female
Qualification	Doctorate, Master, Bachelor, Diploma and Certificate
$PK_{1..6}$	Work Outcome (50%)
$PM_{1..6}$	Knowledge and Skill (25%)
$KP_{1..6}$	Individual Quality (20%)
$KS_{1..6}$	Activities and Contribution (5%)
$YEAR_{1..6}$	Evaluation mark (100%)
Target/Class	Recommendation for promotion (Yes or No)

2.1.2 Inputs requiring Processing

On the other hand, inputs of system's in papers [4], [5], [6] and [8], require some pre-processing. Most choose resumes/CVs as main inputs in order to characterize candidates, although [5] and [8] have complementary entries.

Referencing paper [4], considers as inputs a knowledge base acquired from domain experts alongside a database maintained by an IT industry (Figure 2.2).

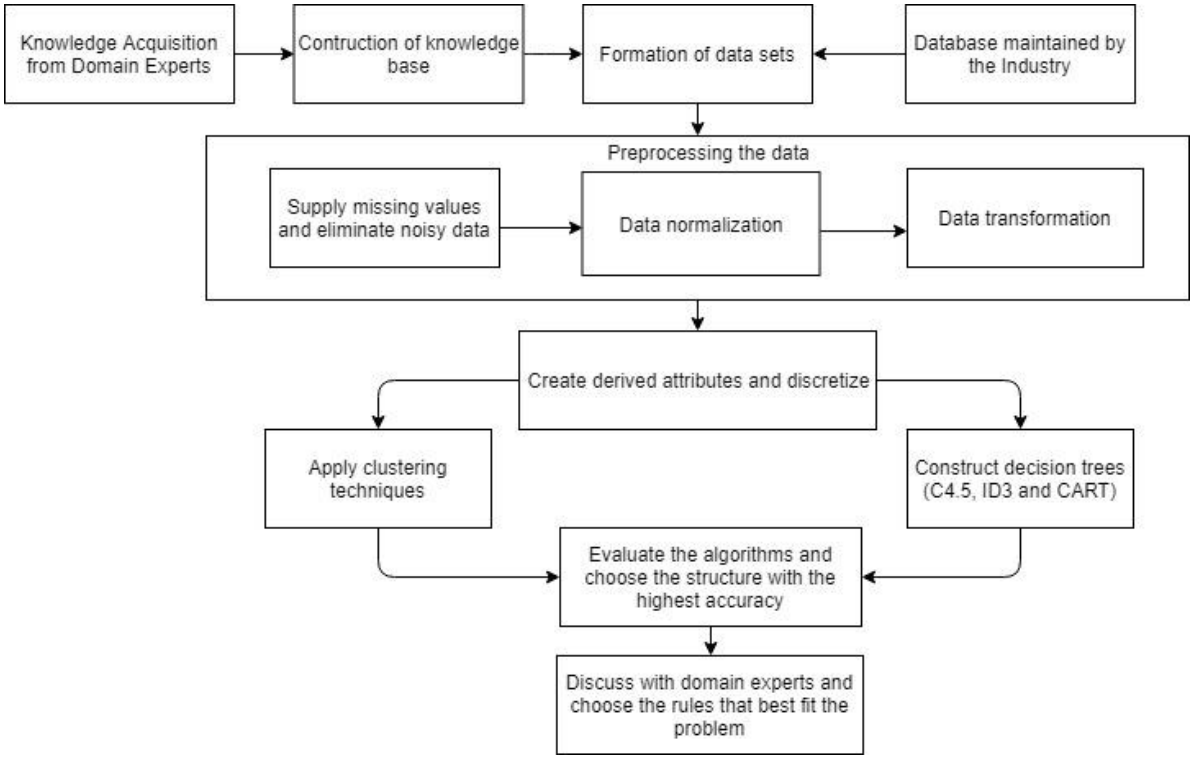


Figure 2.2 - System architecture [4]

The knowledge base comprises a collection of sample data and the database contains information of all the applicants for an industry, which was gathered by contacting an industry's HR department. From these two components, data sets are created. From the data sets, a data pre-processing task is performed in order to improve the quality of data and then, attributes are extracted and discretized. From this stage, clustering and classification techniques are applied.

Paper [6] uses resumes collected from *indeed.com* [19], a web application for job searching and offering. Based on the resumes, tries to extract keywords from all words in order to build a term document matrix. The term document matrix describes the frequency count of words among all resumes, each row representing one resume, each column representing a word and each entry representing the frequency count of a word in a resume. The OFC represents the frequency count of a word W in a resume dataset (1). Some pre-processing was performed to build the term document matrix, such as removing punctuation marks and stop words (natural language words having little to no meaning - A, An, Of etc.), and upper-case letters were converted to lower case in order to standardize the matrix.

$$OFC = \sum_{i=1}^{n} W(i) \qquad (1)$$

Several OFC values were considered (50,75,100) and the results were the following:

Table 2.2 - Attributes and corresponding OFC values considered [6]

Attribute	OFC=50	OFC=75	OFC=100
Number of words in the term document matrix	606	427	345
Number of relevant words for resume ranking	83	63	54
Comparison	Words are biased towards specific resumes, skills are not generic, and OFC is very low	Words are not biased towards specific resumes, skills are generic	Very few words. Unable to capture all generic skills

From the table above, OFC=75 provided the best results and, therefore, was chosen.

Regarding which words are being searched in each resume, they can be divided in multiple categories. For each of those categories, the system tries to find the following words:

Table 2.3 - Words searched by category [6]

Category	Words
Role	Administrator, Analysis, Analyst, Application, Architecture, Backend, Client, Database, Debug, Descript, Design, Develop, Front, Intern, Network, Server, Web
Language, Database and Web related skills	CSS, HTML, JavaScript, jQuery, Json, MySQL, PHP, PL-SQL, Servlet, Shell, SQL, SQLite, XML
Software Packages, Tools and Frameworks	ADO.NET, AJAX, ASP, ASP.NET, Bootstrap, Eclipse, JSP, MVC, SDK, Visual, WordPress, XCode
Experience Skill mentioned in the resume	API, App, Automation, Bug, Cloud, Code, Configuration, Core, Data, Feature, Module, Query, Scripting, Software, Website
Operating System	Android, iPhone, Linux, Unix, Windows

Reference [5] considers a different approach by integrating an online form with a LinkedIn [20] profile and a candidate's blog [13] [14] (Figure 2.3). Both online form and LinkedIn profile are the main inputs of the system and are responsible for containing the more technical, experience and skills related characteristics of each candidate. The candidate's blog can also be considered in order to derive personal traits of the candidate.

Figure 2.3 - Candidate's application user interface [5]

In reference [8], an online CV and a candidate's desired position were considered as inputs. Foremost, the company identified a set of activities for their main task - developing a proprietary software – as:

- Software development
- Testing the product
- Server side and plug-in
- Developing of internal tools

From these activities, a set of job positions (classes) were derived as shown:

Table 2.4 - Set of job positions derived from set of activities [8]

Internship	Junior	Middle	Senior	Not matched
QA (1)	QA (5)	QA (9)	QA (13)	(17)
Python (2)	Python (6)	Python (10)	Python (14)	
C# (3)	C# (7)	C# (11)	C# (15)	
Java (4)	Java (8)	Java (12)	Java (16)	

Regarding the criteria on which the candidate selection is made, the following attributes were chosen: Education, Experience, Foreign Languages (English, French and German), Programming Languages (C#, Java, JavaScript and Python), Salary Range, Gender, Age and Personal Skills. From these set of attributes, a candidate can be described by extracting these features from the online CV (Figure 2.4) and later applied to a classification process.

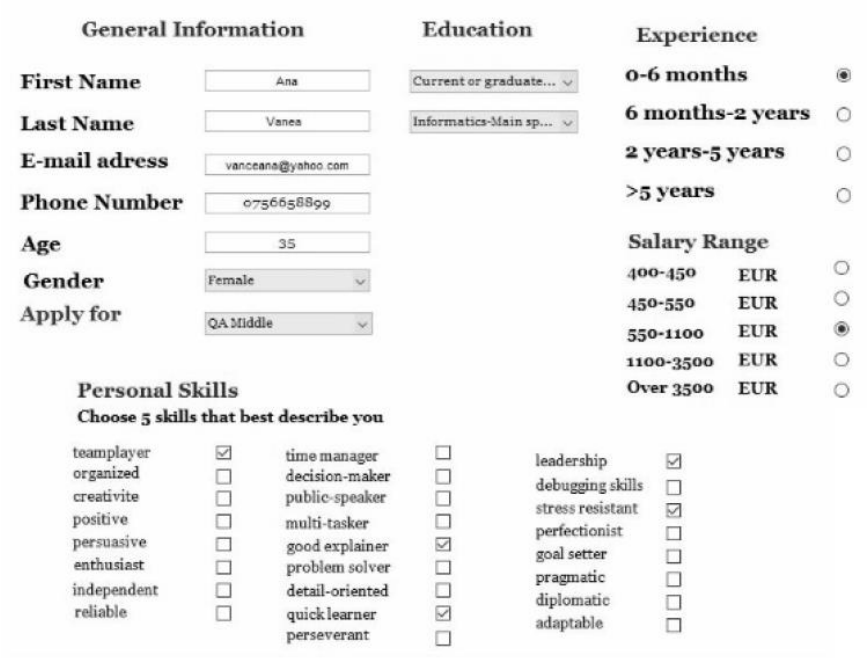


Figure 2.4 - Online CV application's user interface [8]

2.2 Outputs

Regarding system's outputs, state-of-the-art systems give the following information:

- Candidate's score/ranking
- Candidate selected/rejected regarding a certain job position

2.2.1 Score/Ranking

Referencing [5], system outputs a candidate ranking through a classification process based on the candidate's attributes for a given job position (Figure 2.5).

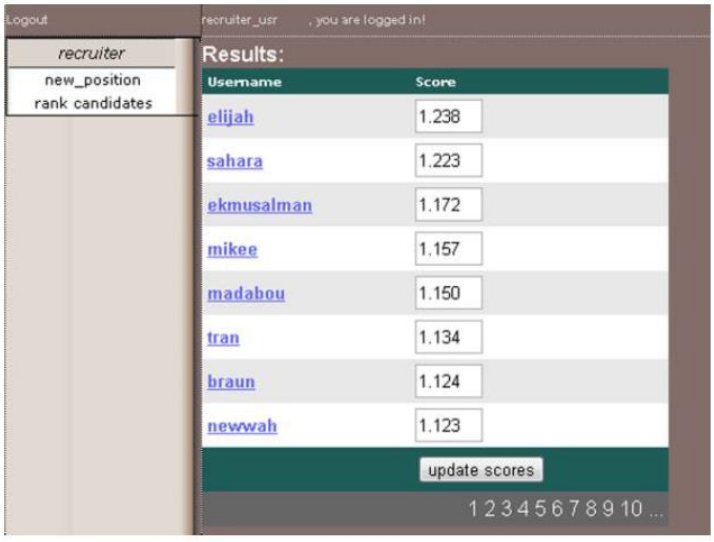


Figure 2.5 - Outputs' application user interface [3]

In [6], the system's output is a cluster-based ranking (CBR). Firstly, the feature vector is based on 63 words for all resumes and the cluster value of the resume has been considered as the target value. Secondly, ReliefF Algorithm [27] was used to calculate the weights to be assigned to each word count.

The **Relief algorithm** (ReliefF's base algorithm) calculates a feature score for each feature in order to help choosing top scoring features for feature selection. Besides, the scores obtained can be applied as feature weights to be used by the system. The algorithm considers a data with *n* instances and *p* features belonging to two known classes, each feature being scaled within the interval [0,1], and iterates *m* times. Starting with a *p*-long weight vector (W) of zeros, each iteration considers a feature vector X belonging to one random instance and the feature vectors of the instances closest to X (smallest Euclidean distances) from each class. The closest same-class instance obtained is called "near-hit" and the closest different-class instance is called "near-miss". Then, based on the calculated distances, the weight vector is updated (2).

$$W(i) = W(i) - (X(i) - nearHit_i)^2 + (X(i) - nearMiss_i)^2 \quad (2)$$

After m iterations, each element of the weight vector is divided by *m*, obtaining the relevance vector. If the feature relevance is higher than a predefined threshold *r*, the feature is selected.

Lastly, each word count C(i) was then multiplied with the corresponding weight W(i), and after the sum of all products, the CBR is obtained *(3)*.

$$CBR = \sum_{i=1}^{63} C(i) * W(i), \ where\ i\ represents\ a\ word \quad (3)$$

2.2.2 Selected/Rejected

In [3], based on the Naïve Bayes classifier, two probabilities are calculated:

- Probability of being eligible for the job ($V_{NB(yes)}$)
- Probability of not being eligible for the job ($V_{NB(no)}$)

Based on the higher probability, the system can conclude if a candidate is eligible for a certain job position or not.

In [8], the system outputs if the candidate is suitable or not for the job and to which class is more suitable (Figure 2.6).

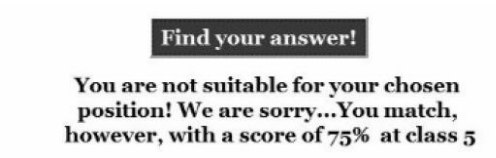


Figure 2.6 - Online CV application's user interface with output [8]

After assigning a rule from all the set of rules derived from the decision tree for each candidate, [5] system can decide, for each candidate, if he suits the job or not (Table 2.5).

Table 2.5 - Example of classification rules derived from the decision tree algorithm [5]

Rule Number	Classification Rules	Result
1	IF (Knowledge5 <= 22.26) AND IF (Outcome4 <= 45) AND IF (Contribution2 <= 3.75) AND IF (Contribution1 <= 3) AND IF (Knowledge6 <= 22.05) AND IF (Contribution3 <= 3.25) AND IF (Gender=Male) AND IF (Outcome2 <= 43.5)	NO
2	IF (Knowledge5 <= 22.26) AND IF (Outcome4 <= 45) AND IF (Contribution2 <= 3.75) AND IF (Contribution1 <= 3) AND IF (Knowledge6 <= 22.05) AND IF (Contribution3 <= 3.25) AND IF (Gender=Male) AND IF (Outcome2 > 43.5)	YES
3	IF (Knowledge5 <= 22.26) AND IF (Outcome4 <= 45) AND IF (Contribution2 <= 3.75) AND IF (Contribution1 <= 3) AND IF (Knowledge6 <= 22.05) AND IF (Contribution3 <= 3.25) AND IF (Gender=Female)	NO

Paper [4] outputs are very alike system in paper [7], outputting if a candidate is selected or rejected based on the rules inferred from decision trees.

2.3 System Approaches

In this section is described how state-of-the-art systems process their inputs. From previous works, system approaches can be divided into two different learning methods:

- Supervised Learning
- Unsupervised Learning

Furthermore, this learning methods commonly comprise three different strategies [25]:

- Clustering algorithms
- Classification algorithms
- Regression algorithms

Regarding **clustering** algorithm-based systems, its intended to cluster the system inputs, generating labelled data. The obtained labelled dataset can be used to train a classifier or to calculate cluster-based outputs. On the other hand, **classification** algorithms [9] seek to classify a candidate into a certain class, while **regression** algorithms output a continuous value, for example a ranking.

Despite the differences between them, all approaches aim at finding how well a candidate suits for a certain job position.

2.3.1 Supervised Learning

In **supervised learning** the goal is to, given a set of input variables and the corresponding output variables, learn the function *f* that maps the input to the output *(4)*.

$$Y = f(X) \qquad (4)$$

Based on the mapping function and considering some new input data X, is possible to predict the output variables Y for that data. Supervised learning is typically performed in the context of classification, mapping input to discrete output labels, or regression, mapping input to a continuous output.

2.3.1.1 Classification

In reference [4], apart from the clustering approach, a classification approach was also considered to build the system. Regarding the classification approach, many decision trees [23] were tested. A **decision tree** is a tree structure, where internal nodes denote a test on an attribute, each branch represents the outcome of the test and the leaf node represents the class labels (Figure 2.7). From the generated tree several classification rules can be deduced.

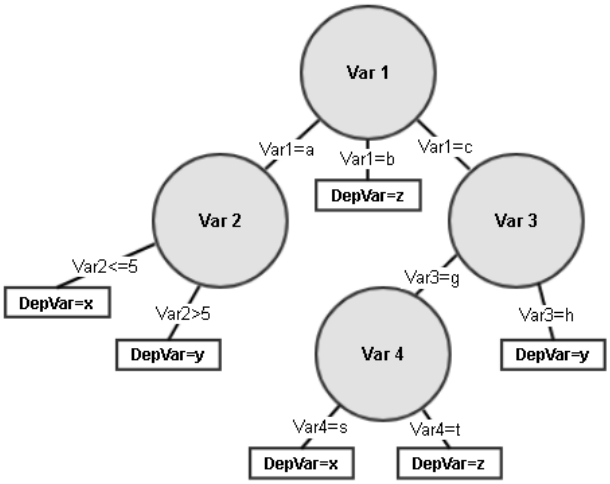


Figure 2.7 - Decision tree example

The results obtained for the several decision trees were the following:

Table 2.6 - Results of decision trees trained with dataset1 and tested with dataset2 [4]

Algorithm	Accuracy (%)
Id3	45.1
C4.5	77.3
C4.5 Unpruned	76.7

Table 2.7 - Results of decision trees trained with dataset2 and tested with dataset1 [4]

Algorithm	Accuracy (%)
Id3	50.3
C4.5	79.1
C4.5 Unpruned	78.7

The **ID3** (Iterative Dichotomiser 3) begins with an original set *S* as the root node and iterates through every unused attribute of that set, calculating the entropy *H(S)* of that attribute. Then, the attribute with the smallest entropy value is selected and ***S*** is split by that attribute, producing subsets of the data. The algorithm runs recursively on each subset, considering attributes never selected before. Throughout the algorithm, the decision tree is constructed with each internal node representing the selected attribute on which the data was split, and each terminal node (leaves) representing the class label of the final subset of the branch.

Regarding **C4.5,** is basically an improved version of ID3 with the following advantages:

- Accepts both continuous and discrete features
- Handle incomplete data points
- Each feature can have different weights
- Solve over-fitting problem by Pruning

Overfitting comprises the training of a model with too much data, which starts learning from noise and inaccurate data entries, resulting in an incorrect categorization of the data.

Regarding **Pruning**, is a technique that reduces the size of decision trees by removing sections of the tree that provide little power to classify instances, reducing complexity of the final classifier and improving the predictive accuracy by the reduction of overfitting.

For classification algorithms, pruned trees gave better results, with C4.5 having the best accuracy. From the decision tree chosen, several rules were deduced.

In [7] C4.5 decision tree classifier is selected. The system's goal is to discover employees' performance patterns from the existing employees' performance data using the C4.5 decision tree classifier. From the decision tree built by the system, 43 classification rules are derived from 590 instances in order to evaluate and bind each candidate to a rule. The C4.5 classifier produced the following results:

Table 2.8 - Results from C4.5 classifier [7]

Correctly Classified Instances	Incorrectly Classified Instances	Mean absolute error	Root mean squared error	Relative absolute error	Root relative squared error	Total Number of Instances
561 (95.1%)	29 (4.9%)	~0.09	~0.21	25.3%	50.4%	590

Based on the features extracted, reference [3] calculates frequencies and probabilities, applying Naïve Bayes [24] theorem (5) to compute most probable class (V_{MAP}) (6) regarding each candidate. **Naïve Bayes** being a simple probabilistic classifier based on the application of Bayes' theorem with strong (naïve) independence assumptions between the features.

$$P(A|B) = \frac{\mathrm{P(B|A)P(A)}}{\mathrm{P(B)}}, Where\ A\ and\ B\ are\ events\ and\ P(B) \neq 0 \qquad (5)$$

$$Vmap = \arg\max_{vj \in V} P(vj \mid a1, a2, \ldots, an) = arg \max_{vj \in V} \frac{P(a1,a2,\ldots,an \mid vj)P(vj)}{P(a1,a2,\ldots,an)} \quad (6)$$

From the features extracted for each candidate, system [8] tests several classifiers such as KNN, LDA, J48, Naïve Bayes and SVM [25].

K-Nearest Neighbors algorithm is among the simplest machine learning algorithms, calculating the *k* closest points in the feature space to the instance being tested, predicting its class based on the predominant class from the *k* neighbors.

The algorithm receives as input vectors in a multidimensional feature space, each instance with a given label. The training phase consists only in the storage of the feature vectors with corresponding labels. Yet, the classification phase requires a specified value *k* and new un-labelled instance.

First, the algorithm calculates the distances between each point in the feature space and the new instance to be classified. Several distance metrics can be selected; however, the Euclidean distance is the most common (7).

$$d(p,q) = \sqrt{(a_1-b_1)^2 + (a_2-b_2)^2 + \cdots + (a_n-b_n)^2}, a \; and \; b \; as \; feature \; vectors \quad (7)$$

Based on the chosen parameter *k*, the algorithm obtains the k-shortest distances between every data point and the instance to be classified.

Then, the predominant class of the *k* nearest elements will be the class assigned to the new instance considered. For example, in figure 2.8, if the *k* value chosen is equal to 1, the new instance will be classified as class 1, however if the k value considered is equal to 3, the new instance will be classified as class 2 (class 2 has two elements near the new instance and class 1 only has one element).

Regarding the *k* parameter selection, usually larger values of k reduce the noise effect on the classification but make boundaries between classes less distinct. Several heuristics can help the selection of an adequate *k* value, such as the hyperparameter optimization.

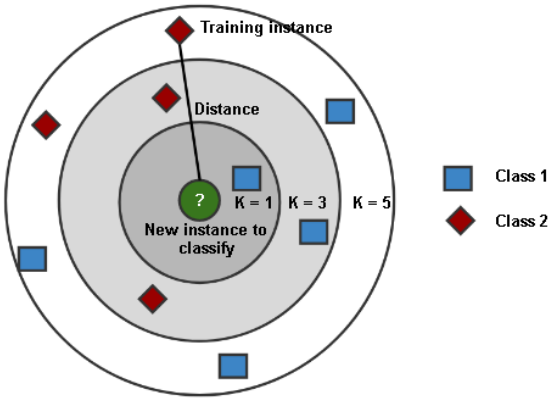


Figure 2.8 - KNN classification example

J48 is simply a Java implementation of C4.5 algorithm.

SVM are supervised learning models with associated learning algorithms that given a set of training data, each belonging to one of two categories. An SVM model is a representation of the examples as points in space, mapped so that the data of the separate categories are divided by a clear gap that is as wide as possible. New instances are then mapped into that same space and predicted to belong to a category based on their position in the given space.

The data set contains 172 instances representing the candidates for the given jobs, each candidate being described by 17 attributes (Figure 2.9).

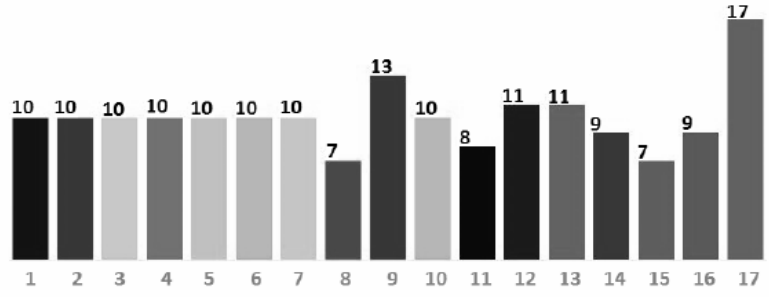


Figure 2.9 - Graphical representation of the data set (Number of instances per each of the 17 classes) [8]

The system tries to predict the class in which the candidate is more suitable, and the results obtained vary according to the way the data set was created and to its complexity (number of attributes and classes) (Table 2.9).

Table 2.9 - Error obtained for each classifier and based on the dataset chosen [8]

Setup	Error (%)			
	LDA	KNN	NB	TREE
All 172 instances, but only based on the first 15 attributes	9.30	11.05	17.44	11.63
All 172 instances with all 18 attributes	10.47	8.72	12.21	11.00
Only 121 instances (70%) with all 18 attributes	7.44	9.09	14.05	12.40

2.3.1.2 Regression

In reference [5], several regression algorithms are studied and tested to find out which one performs best in this context. The different approaches were:

- Linear Regression
- Regression Tree
- Support Vector Regression

In **linear regression**, the relevance score y_i of the i^{th} candidate is predicted as a linear function of the selection criteria, which comprises the candidate's feature vector x_i plus noise *e* (regression error) (8).

$$y_i = w^T x_i + e \quad (8)$$

The linear regression algorithm finds the optimal parameter vector w that minimizes the regression error. However, this approach is not appropriate when selection criteria interacts in a complex and non-linear way.

Regarding **regression trees**, they can be a viable alternative to linear regression when non-linearities of the selection criteria are verified. Regression trees recursively partition the predictor space using a divide and conquer approach, with internal nodes containing tests and leaves containing predictions for the class value. The types of regression trees used were M5 model tree and REP tree.

Lastly, **support vector regression** was tested. The objective of support vector regression is to find a function *f* that minimizes the expected error according to the unknown probability distribution of the data (9).

$$f(x) = \sum_{i=1}^{N} a_i K(x, x_i) + b, b \; and \; a_i \; \epsilon \; R \quad (9)$$

The results obtained regarding all the different implementations were the following:

Table 2.10 - Correlation coefficient of different job positions for each classifier [5]

Pearson's Correlation coefficient	LR	M5 Tree	REP Tree	SVR (poly)	SVR (PUK)
Sales engineer	0.74	0.81	0.81	0.61	0.81
Junior programmer	0.79	0.85	0.84	0.81	0.84
Senior programmer	0.64	0.63	0.68	0.62	0.73

Table 2.11 - Pearson's coefficient and relative error for each classifier [5]

	LR	M5 Tree	REP Tree	SVR (poly)	SVR (PUK)
Pearson's coefficient	0.63	0.63	0.65	0.28	0.65
Relative error	25.3%	25.3%	22.5%	57.4%	23.1%

In the first experiment, the correlation of the scores output from the system with the actual scores assigned by the recruiters was tested using the Pearson's correlation coefficient [31] metric. From those results, we can state that the tree models and the SVR with a PUK kernel produce the best results [26]. On the other hand, linear regression doesn't deliver good results, which suggests that the selection criteria are not linearly separable.

In the second experiment, the effectiveness of the personality mining component is put to test. Based on the textual data extracted from the candidate's blog, the system and an expert recruiter predict an extraversion score. This system's score is predicted by training a regression model to the scores

assigned from the recruiter to each of the 100 candidates. Although predicting recruiter's evaluation is a hard problem, a significant correlation was obtained with a Pearson's coefficient of up to 0.65.

2.3.2 Unsupervised Learning

On the other hand, **unsupervised learning** tries to model the underlying structure or distribution in the data and is typically done in the context of clustering.

In this type of learning method, the corresponding output variables for each input data of the training datasets are not available.

2.3.2.1 Clustering

Both systems in references [4] and [6] consider a clustering approach. In reference [6], K-means [21] is used in order to cluster the resumes having only the relevant word counts as features.

K-means is one the simplest unsupervised learning algorithms for clustering problems and aims at forming K clusters of n objects.

The algorithm selects k centroids in a multidimensional feature space, each centroid representing each cluster mean or center. Then, the algorithm iteratively alternates between two steps: assignment and update. First, each object is assigned to the cluster they are most similar with, based on the distance between the object and the cluster mean. Usually the distance metric chosen is the Euclidean distance (7). Then, the new mean is computed for each cluster, each mean being calculated as the average coordinates of all points in the cluster (10).

$$new\ mean = \left(\frac{a_1+b_1+\cdots+x_1}{n}, \frac{a_2+b_2+\cdots+x_2}{n}, \dots, \frac{a_p+b_p+\cdots+x_p}{n}\right), \tag{10}$$

$$a, b\ and\ x - points\ in\ the\ cluster, \qquad n - number\ of\ points\ in\ the\ cluster,$$
$$p - \ number\ of\ features$$

The algorithm iterates until the criterion function converges, generally when the distance between the previous state centroid and the current state centroid is smaller than a predefined convergence epsilon, for all *k* clusters.

In this work, all the resumes have been categorized into one of the K clusters (determined by the Elbow method [30]) and each resume goes into the cluster with the nearest mean. Regarding the clustering techniques in [4], K-means and Fuzzy C-means [22] were tested.

Fuzzy C-Means is a method of clustering which allows one piece of data to belong to two or more clusters. This algorithm works by assigning membership to each data point based on the distance between the data point and each cluster center. The closest the point is to the cluster center; more is its membership towards that cluster center. The sum of membership of each data point should be equal to one.

Before deciding which approaches to choose for the decision support system, built models were compared and evaluated. The evaluation measure considered was accuracy and is determined by the

ratio of records correctly classified/clustered during testing to the total number of records tested. The data sets contained 770 records for the first set and 2808 for the second set.

The results are shown below, demonstrating that clustering techniques achieve low accuracies and are not suitable for this problem due to the nature of the data.

Table 2.12 - Results of Clustering algorithms trained with dataset1 and tested with dataset2 [4]

Algorithm	Accuracy (%)
Fuzzy C-means	52.1
K-means	53.5

Table 2.13 - Results of Clustering algorithms trained with dataset2 and tested with dataset1 [4]

Algorithm	Accuracy (%)
Fuzzy C-means	63.1
K-means	69.6

2.4 Chapter Conclusions

From table 2.14, we can have a global understanding of the implementations regarding state-of-the-art systems in order to decide which direction to take in the solution's architectural design.

The table is divided in the following columns:

- Article/Paper
- Inputs – describes which inputs were considered by the system
- System – describes which machine learning techniques were used
- Outputs – describes which outputs were given by the system
- Accuracy – describes the ratio between the number of correctly classified instances and the total number of instances

From table 2.14, regarding inputs, most systems consider CV's derived information. Concerning clustering algorithms, due to the nature of the data, the systems couldn't obtain great results. However, with the adequate data sets, K-Means is a very promising algorithm and, therefore, important to do further testing in this context.

Regarding classification and regression algorithms, trees-based algorithms are among all, the most common and accurate, C4.5 being one of the best. However, Naïve Bayes, KNN, LDA and SVR also achieved significant accuracies.

At last, most systems output either a ranking or a selected/rejected result, however, in this case the proposed system will follow Polarising's requirements and specifications in order to obtain the required information.

Table 2.14 - Comparison between state of the art systems and accuracies achieved

	Article/ Paper	Inputs	System	Outputs	Accuracy
Clustering Algorithms	[6]	Resumes	K-means	Ranking	*Good but no numerical evidence*
	[4]	Knowledge Base	Fuzzy C-means	Candidate's Cluster Group	52.1% - 63.1%
			K-means		53.5% - 69.6%
Classification Algorithms			Id3	Candidate Selected/Rejected	45.1% - 50.3%
			C4.5		76.7% - 79.1%
			CART		72.1% - 77.3%
	[3]	Features	Naïve Bayes	Candidate Selected/Rejected	*Good but no numerical evidence*
	[7]	Features	C4.5	Candidate Selected/Rejected	77.0%
	[8]	Online CV + Desired job position	LDA	Candidate Selected/Rejected	89.5% - 92.6%
			KNN		89.0% - 91.3%
			Naïve Bayes		82.6% - 87.8%
			TREE		87.6% - 89.0%
Regression Algorithms	[5]	LinkedIn profile + Blog + Online CV	LR	Ranking	~70.0%
			M5 Tree		~70.0%
			REP Tree		~65.0%
			SVR		~75.0%

Chapter 3

System Architecture

This chapter describes the system's architecture by characterizing its different layers and how they communicate with each other and with external applications.

3.1 Interaction between Functional Blocks

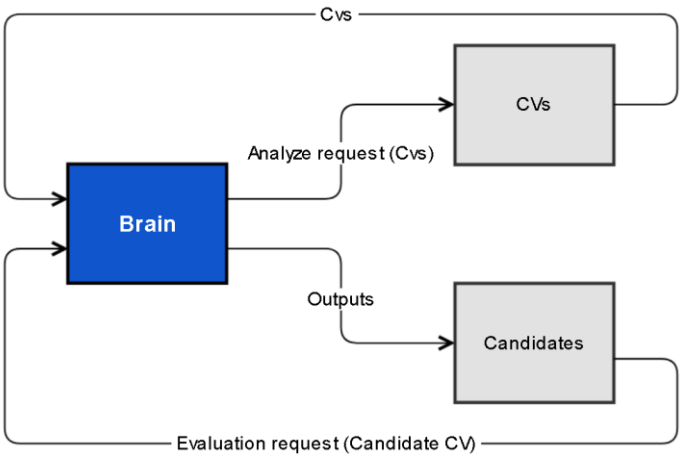


Figure 3.1 - Interaction between functional blocks

The system within this thesis scope is mainly **Brain**, with CVs and Candidates being already implemented. However, some changes are required to the Candidates' and CVs' block in order to enable the integration of all applications.

3.1.1 Authentication

In order to securely communicate between applications, every functional block must have an authentication layer, distinguishing secure peers from unsecure ones. Since the interaction between applications will be through requests and responses, each request must be sent with a valid access

token. Then the system receiving the request must validate whether the requesting system has access to the intended resource. If it does, the system responds with the required information (200 OK HTTP status code), if it doesn't, the system will receive a message informing it is unauthorized to access the resource (401 Unauthorized HTTP status code).

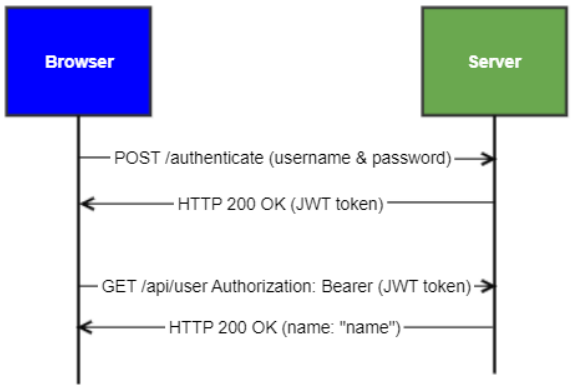


Figure 3.2 - Authentication flow

3.1.2 CVs

The CVs block is where the data and CVs from all Polarising's collaborators are stored and organized. Regarding the company's context, it provides a web application where each collaborator can view and update their own CV.

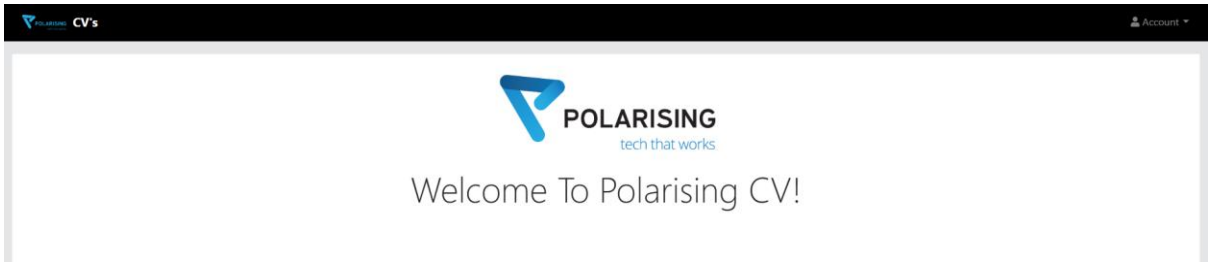


Figure 3.3 - CVs' web application

3.1.3 Candidates

The Candidates' block works as the bridge between all the main recruitment processes and the candidates. For each application, the recruiter will trigger the evaluate function call in order to characterize/analyze an incoming candidate.

This block is the main communication block with the proposed system, displaying most Brain's outputs.

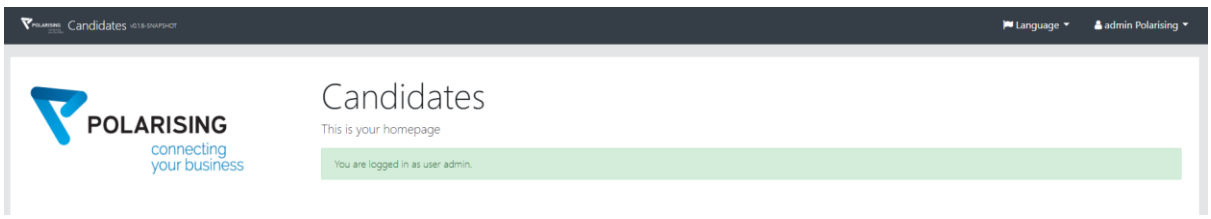


Figure 3.4 - Candidates' web application

3.1.4 Application Programming Interface (API)

The project's application programming interface implements two different functionalities:

- Analyze
- Evaluate

The **Analyze** function will be triggered when Brain starts running or by user input at its web application, requesting all collaborators CVs to the CVs application, performing a process of analyzing the data. This function call should always be executed when the characterization methodology changes, for example, changing the importance of an attribute/feature.

On the other hand, the **Evaluate** function call will be triggered by the candidates' block and, upon receiving the request, Brain's will evaluate the incoming candidate, comparing it to all collaborators and outputting the top individuals who match closer to the candidate and his main characteristics/attributes. Furthermore, will predict the candidate's class and output several rankings based on his features.

At last, for each evaluate function calls, Brain will reply to the Candidate's block with specific outputs related to the corresponding request.

3.4 Internal System Structure (Brain)

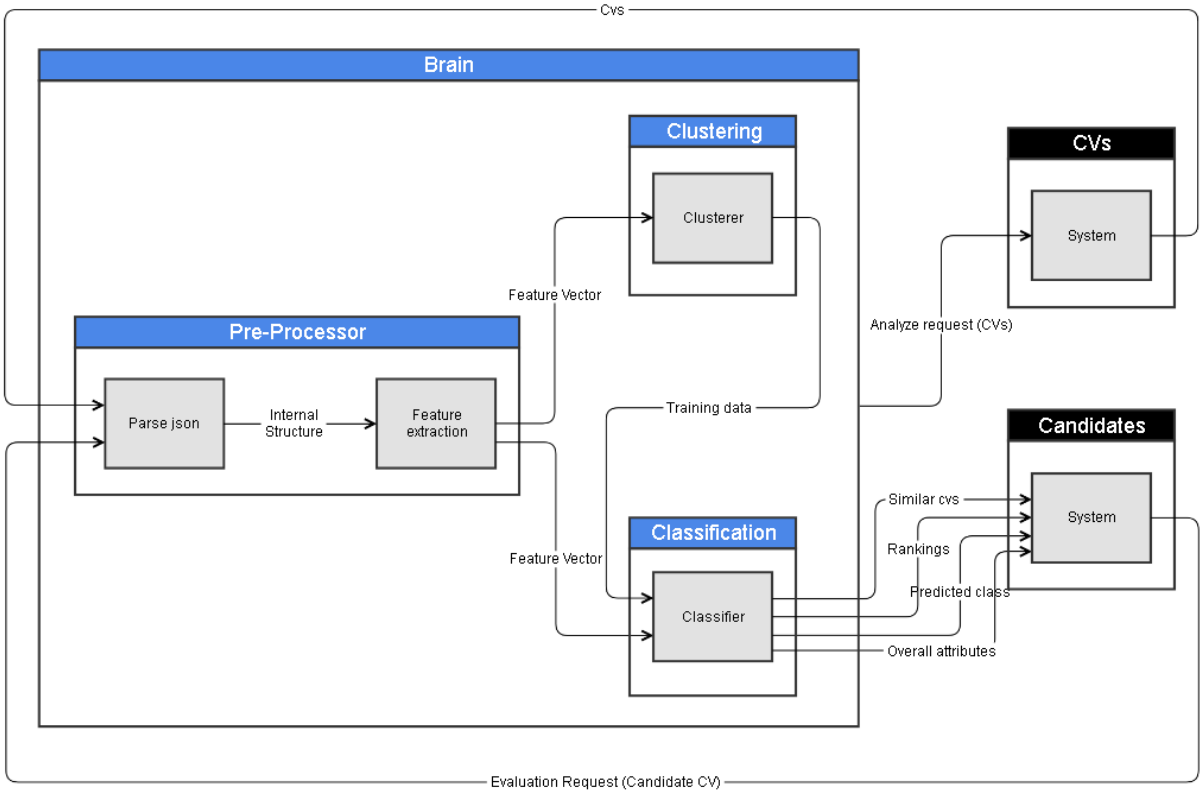


Figure 3.5 - Brain system architecture

In this chapter is explained how each component of the system is structured and implemented. Regarding the internal system structure (Figure 3.5), as described in chapter 2, we can divide it into:

- Inputs and pre-processing
- Outputs
- System approach

3.4.1 Inputs and Pre-Processing

First, Brain requests all the available data to the CVs system (analyze request) before performing any other task. Then, based on the data received, each collaborator CV will be pre-processed and converted into a known format by system, feeding the clustering stage. This stage considers all collaborators CVs in order to study the underlying structure of the company, outputting every CV along with a label corresponding which cluster they belong – training data. The referred data is used to train a classifier in order to evaluate incoming candidates.

After the system initialization is performed, Brain is ready to receive evaluation requests. This happens by a Candidates' block evaluation request, sending to Brain a candidate's CV. The incoming CV will be pre-processed the same way as any other company CV, feeding the classifier. Based on the obtained training data through the clustering stage along with the pre-processed candidate's CV, the classification stage is performed, obtaining several outputs and replying them to the Candidates application.

3.4.2 Outputs

Regarding system's outputs, the system will generate the following outputs:

- Overall attributes
- Similar collaborators
- Rankings
- Predicted cluster overall characteristics

The overall attributes represent the main characteristics the candidate possesses, concerning several predefined categories of evaluation. Based on that categories the system searches for relevant information in order to characterize the individual and finding his stronger points.

Similar collaborators corresponds to a list of the more resembling collaborators regarding an evaluated candidate. Based on the characteristics and attributes of each CV, the system finds the closest individuals from the whole company.

For each incoming candidate, several rankings are calculated. These rankings evaluate how adequate a candidate is for a certain open job position. After defining objective job positions, the system obtains a ranking for each position regarding the evaluated candidate.

At last, the system outputs a class prediction for a certain candidate, representing the cluster where the candidate is more adequate to belong based on his characteristics. Also, for that class prediction, the

cluster overall characteristics are obtained through the same means as the calculation of the candidate overall attributes.

3.4.3 System Approach

In terms of the system approach, we can specify two phases: clustering and classification.

First, we need to focus on the existing company's CVs, preparing them to be used as training data in the latter phase. The **clustering** stage occurs at the system's boot and every time the web application's user intends to re-evaluate the company's distribution. Since no CV is already labelled, the first stage is to label each collaborator's CV by applying a clustering algorithm. From the study of state-of-the-art systems, K-means is chosen as the clustering algorithm.

The **classification** stage is triggered every time a candidate needs to be evaluated during the recruitment process. Since Polarising's main goal is to find out how each candidate compares himself with the existing company personnel and how well he fits for the open job positions, KNN will be used in order to find the nearest collaborators of a certain candidate and classify the candidate into a certain class/group.

3.5 Web Applications

In addition to the outputs generated and displayed in the Candidates' web application, some information is visible in the proposed system's web application.

3.5.1 Brain

The Brain web application serves the purpose of providing statistical information regarding Polarising's current workforce distribution, alongside with some adjustable parameters regarding the system's clustering task.

The user can adjust the weights assigned to each attribute/feature and to change the number of clusters to be found by the system. Furthermore, the clusters/groups found are displayed and characterized regarding (Figure 3.6):

- Technical skills' categories
- Work experiences' job positions
- Group elements

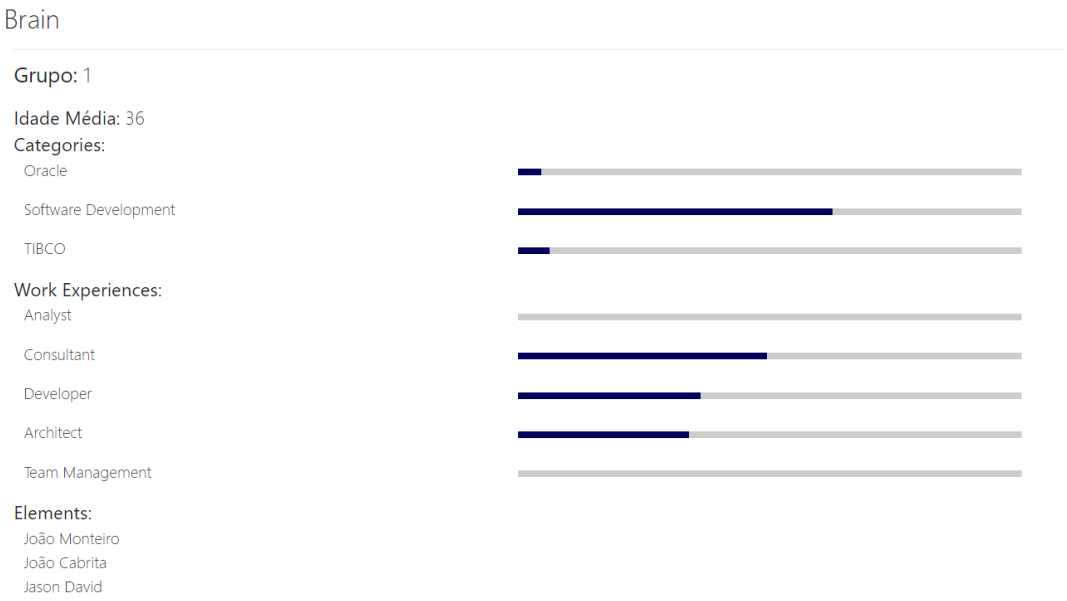


Figure 3.6 - Brain's web application

This information is calculated the same way as described in the outputs section for candidate's predicted class/cluster.

Based on this data, the user will have a better understanding of the company's distribution by knowing which person is in each group and what characteristics defines them. Additionally, this information may be very valuable in identifying where are the most skilled individuals, and where the company lacks resources, guiding the recruitment process with the purpose of strengthening the company's workforce.

3.5.2 Candidates

On the other hand, the Candidates' web application displays the outputs generated by the proposed system. There are three main views regarding the application: **candidates' view**, **candidate's edition** and **candidate's detail**.

The **candidates' view** displays all current candidates enrolled in the recruitment process, along with some candidate specific information, such as personal information, status, area which he applied and the source of his application.

The **candidate's edition** screen allows to fill in the information about the candidate and specifically his technical skills and work experiences (Figure 3.7). Doing so, allows the candidate to request Brain's evaluation and display its outputs.

At last, by clicking the view button on a candidate, the **candidate's detail** screen is rendered (Figure 3.8). After the click, if the candidate contains any technical skills or work experiences, an immediate request is sent to the proposed system, synchronously displaying brain's outputs regarding that candidate.

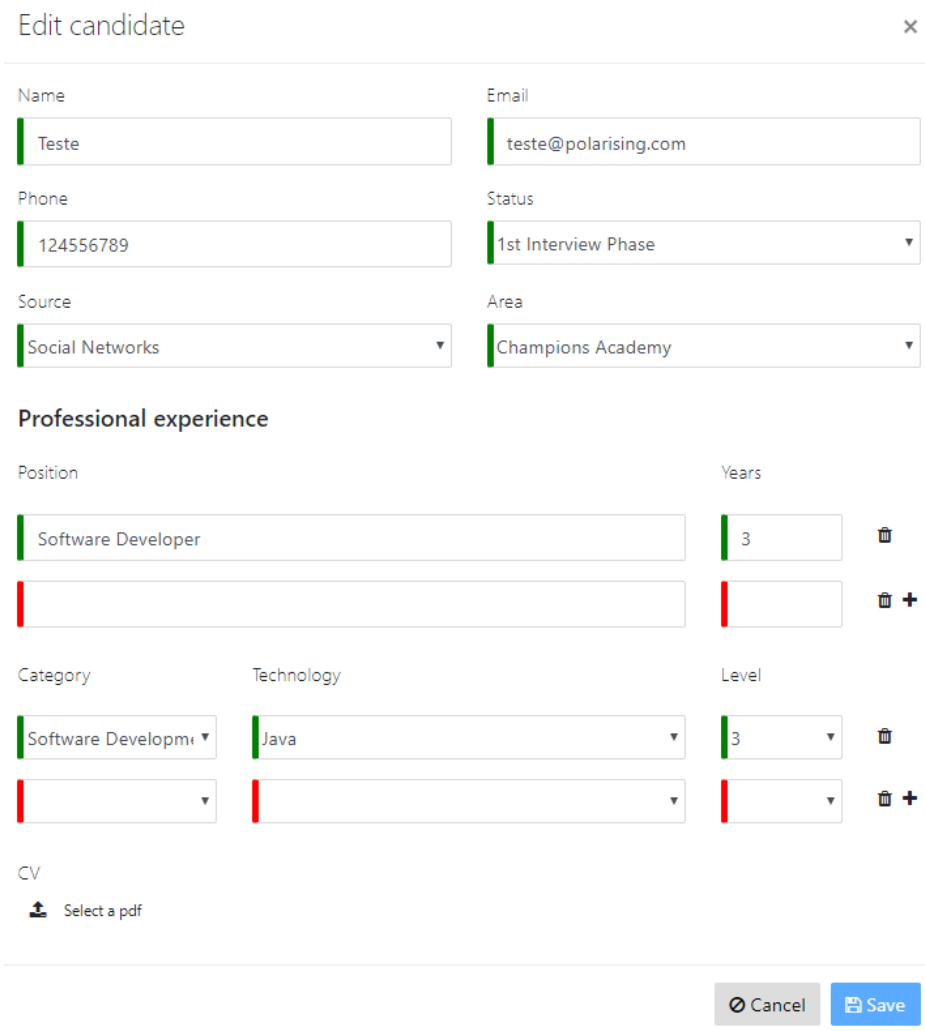


Figure 3.7 - Candidate's edition page

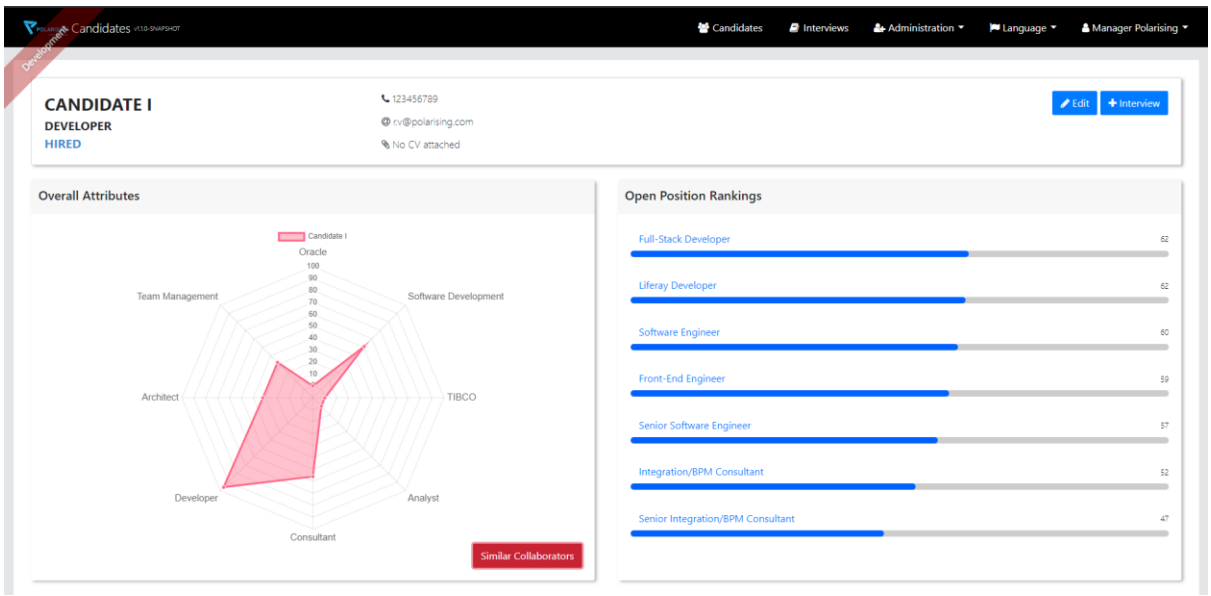


Figure 3.8 - Candidate's detail page

Chapter 4

System Implementation

This chapter describes the requirements and specifications identified for the system alongside with the implementation of each system component.

4.1 Requirements and Specifications

Based on the interaction between the presented applications, several requirements and specifications emerge.

Every design decision is based on either the company's restrictions and specifications or based on the state-of-the-art systems that achieved the best results. The proposed solution intends to handle the number of CV’s reaching Polarising everyday by building a decision support system based on machine learning techniques.

Polarising is very interested in finding out how each candidate compares himself with the existing company personnel and how well he fits for the open job positions. Based on the technologies used in Polarising, the system is implemented mainly in Java language using the Spring framework [32]. However, some components related to the system’s front-end and graphical user interface are implemented in Typescript, CSS and HTML using Angular web application framework [28].

Since Spring and Angular will be the main technologies used to implement the proposed system, JHipster [35] will be used to boost the development by quickly generating the project with all the needed initial requirements.

4.1.1 Spring

The Spring framework [32] is an open source **application framework** and **IoC container** (Figure 4.1) for the Java platform. The IoC container enables the configuration and management of Java objects using a concept called reflection. Reflection is simply the ability of a computer program to lookup, examine and modify its own structure at runtime.

The container is also responsible for the management of specific objects' lifecycles by:

- Creating objects
- Calling object initialization methods
- Wiring objects together

Objects that were created by the IoC container are called beans and the container can be configured by loading XML files or detecting specific Java annotations through the @ notation.

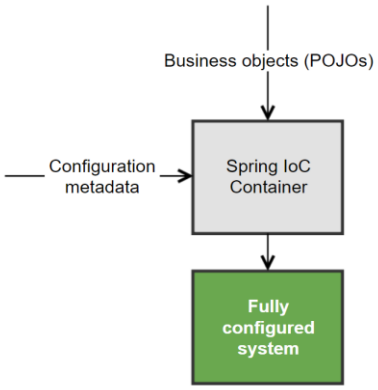


Figure 4.1 - Spring IoC Container

Objects can be obtained via their dependency lookup or dependency injection (Figure 4.2). **Dependency lookup** is simply a pattern where a caller asks the container object for an object with a specific name or type whereas **dependency injection** is a pattern where the container passes objects by name to other objects through constructors, properties or factory methods. This way, all the objects are instantiated and initialized by Spring and injected in the right places (Servlets, Web frameworks, Business classes, DAOs etc.).

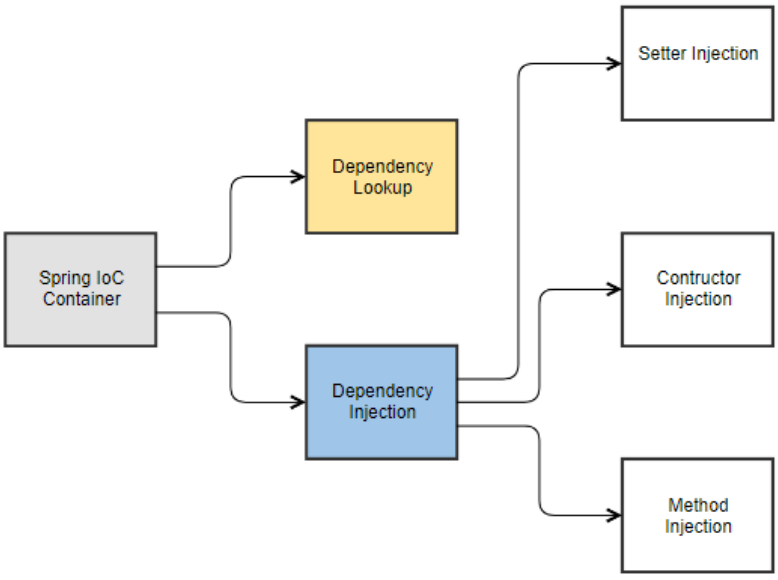


Figure 4.2 - Dependency Lookup and Dependency Injection

The framework's core features can be used by any Java application, although there are several extensions/modules [34] that can be used in order to add more functionalities and reduce user configuration, making the developer's work easier and faster.

4.1.2 Angular

Angular is a TypeScript-based open-source web application framework. Combines declarative templates, dependency injection, end-to-end tooling, and integrated best practices to solve development challenges. Angular empowers developers to build applications that live on the web, mobile, or the desktop. Besides the TypeScript side of Angular (application logic), comprises also HTML (application page structure and content) and CSS (application visual enhancer).

An Angular application can have the following project structure:

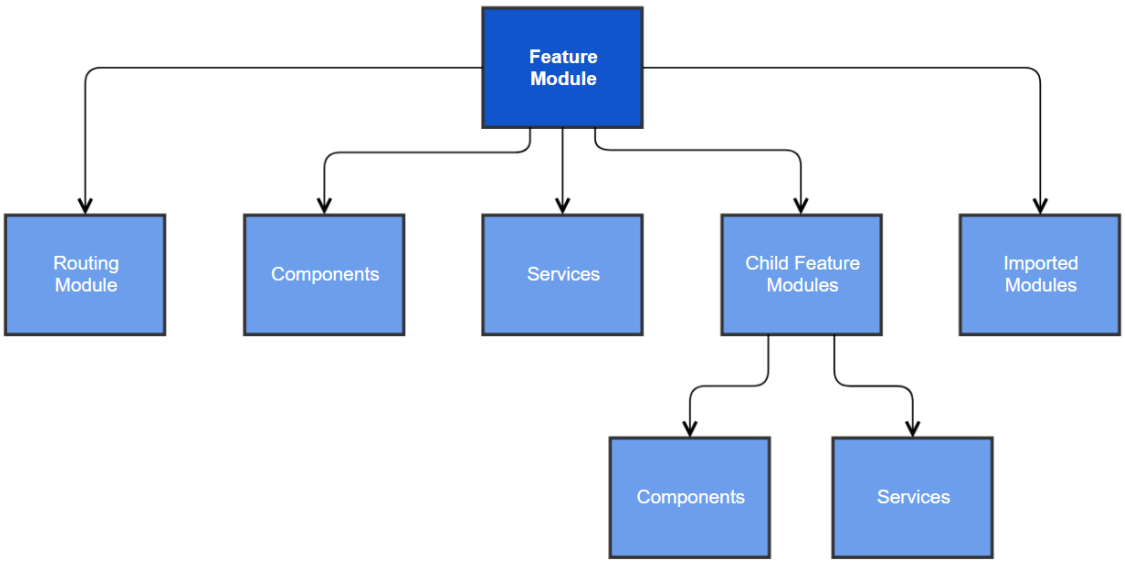


Figure 4.3 - Angular application structure

Regarding figure 4.3, the **feature module** represents a section of the application, the **routing module** represents the routing definitions for content in feature module and **imported modules** contain the shared or node modules with content needed for the given feature. Furthermore, **components** and **services** can be used by feature module or **child feature modules**, which are sections of the feature module having their own independent components and services.

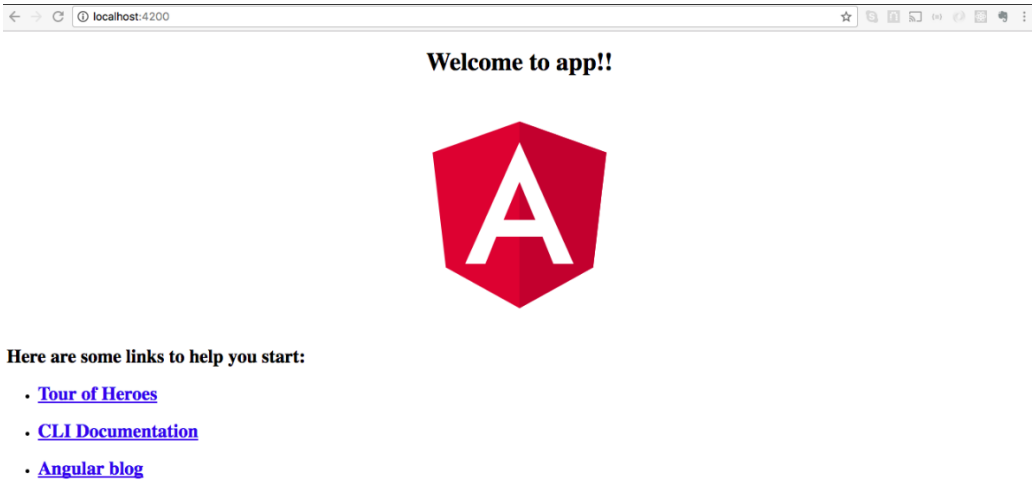


Figure 4.4 - Angular default web application example

4.1.3 Keycloak

In order to provide a secure channel of communication between applications, Keycloak is used, providing single sign-on along different systems. Single sign-on is a property of controlling the access of multiple related, yet independent applications through a centralized authentication system. With single sign-on, a user logs in with a single id and password, gaining access to multiple applications.

First, Brain will provide its system credentials (id and password) to the Keycloak server, obtaining an access token and a refresh token. With the access token it can make requests to the other related applications providing the token in the request headers and, in case of the expiration of the token, it can request another valid token to the Keycloak server by providing the refresh token.

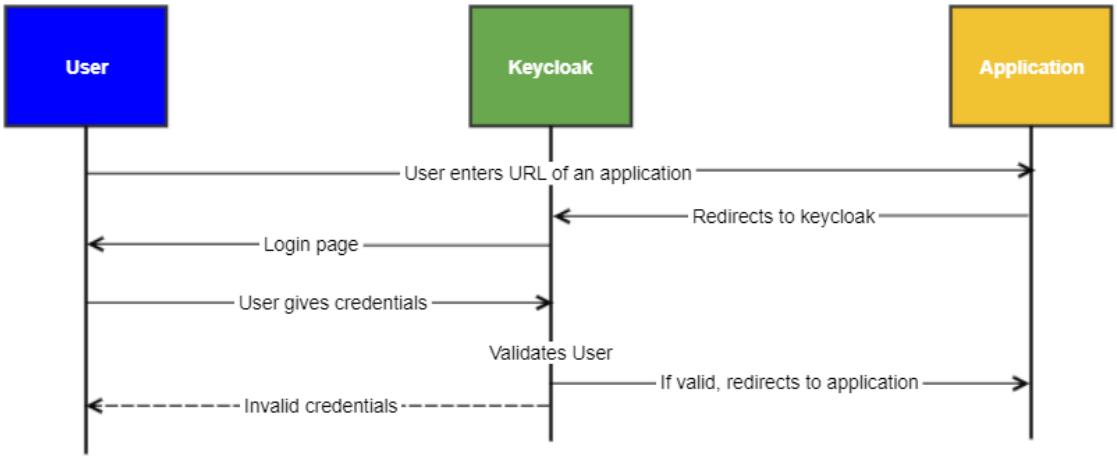


Figure 4.5 - Keycloak's single sign on authentication flow

4.1.4 JHipster

Figure 4.6 - JHipster project generation CLI

JHipster is a free and open-source application generator used to quickly develop web applications and microservices using Angular or React and the Spring framework.

Figure 4.6 shows the command line interface, where you can generate your project structure by selecting/configuring your application specifications.

4.2 Project Structure Generation

Regarding the generation of the project, the JHipster tool was chosen to boost initial configurations. During this stage, several questions were prompted through JHipster's CLI, affecting the system's structure.

4.2.1 Application type

JHipster provides four different predefined application types: **Monolithic**, **Microservice application**, **Microservice gateway** and **JHipster UAA server**.

Monolithic is the default application type, used mostly for simple projects. This architecture contains both the backend and the frontend source code. On the other hand, microservices' architecture style has a distinct characteristic, splitting the front-end from the back-end. By doing this, it makes the job of scaling and adapting an application to infrastructure issues easier. A microservice application is simply the back-end side of the microservices' architecture, handling REST requests. Regarding a microservice gateway, it represents the front-end side of the microservices' architecture, handling web traffic and serving an Angular application. At last, JHipster UAA server consists of a user accounting and authorizing service for securing JHipster microservices using the OAuth2 authorization protocol. JHipster UAA is a fully configured OAuth2 authorization server with the users and roles endpoints inside, wrapped into a usual JHipster application.

This said, the main decision was between the monolithic architecture style or the microservices' one. Since the application to implement won't need significant infrastructural changes after the initial implementation stage, the simpler and cohesive **monolithic architectural style** was selected.

4.2.2 Names

Several questions were prompted regarding project naming, namely: the application's **base name** and the application's **package name**. Regarding the project's base name, according to the name decided with Polarising, **brain** was locked. In terms of the package name selected, based on the conventions adopted within Polarising, **com.polarising.brain** was chosen.

4.2.3 Authentication type

Regarding authentication types, JHipster provides: **JWT authentication** – using a JSON Web Token (JWT) and is the default and most common choice among users, **OAuth 2.0 / OIDC authentication** – using an OpenId Connect server, like Keycloak or Okta, handling authentication outside of the application, **HTTP Session authentication** – the classical session-based authentication mechanism, alongside with Spring Security, and **Authentication with JHipster UAA server**, using a JHipster UAA server that must be generated separately, being an OAuth2 server that also handles authentication outside the application.

Since current in-develop Polarising projects are using Keycloak single sign-on for authentication, the JHipster's authentication type configuration chosen was **OAuth 2.0 / OIDC Authentication**, instead of the other types provided.

4.2.4 Database type

JHipster provides different databases to choose from. H2 being the only database already integrated in the generated package, requiring no additional configuration by the user. On the other hand, MySQL, MariaDB, PostgreSQL, MSSQL, MongoDB, Cassandra and Couchbase are also provided, but require installation and configuration by the user, after the project generation.

Since no data needs to be stored in the proposed system, being only a processing node, no database is required to be selected and configured.

4.2.5 Build tool

JHipster provides two different build tools: **Maven** and **Gradle**. Maven is more stable and Gradle being more flexible and easier to extend. Since extendibility of the project will not be a concern and since Maven is the preferred build tool currently used in Polarising projects, **Maven** was selected.

4.2.6 Client framework

Regarding client-side frameworks, JHipster provides **Angular** and **React** to choose from. Since currently Polarising doesn't develop applications using React, in order to take advantage of the company's knowledge and guidance, **Angular** was chosen as the client-side framework.

4.2.7 Languages

At last, similarly to most Polarising applications, two languages were selected: **Portuguese** and **English**.

4.3 Implementation

4.3.1 Inputs and Pre-processing

First, the system will receive CVs, with a specific structure chosen by Polarising (Figure 4.8 and 4.9), in json [29] through HTTP [33]. This will be done by executing a GET request [33] to http://10.0.1.14:32173/api/cvs/exportCvs, specified in the CVs block. To do so, the RestTemplate's interface [37] provided by Spring was used, mapping the json objects to Java objects. Then, based on the body of the JSON response, all CVs will be extracted and returned (Figure 4.7).

```
getAllCvs() {

    cvsJSONResponse = getRequest("http://10.0.1.14:32173/api/cvs/exportCvs");

    cvs = cvsJSONResponse.getJsonBody();

    return cvs;
}
```

Figure 4.7 - Get all collaborators' CVs pseudocode

A specific structure was chosen to make the process of identifying key features of a candidate easier and more accurate as possible, instead of performing text mining techniques.

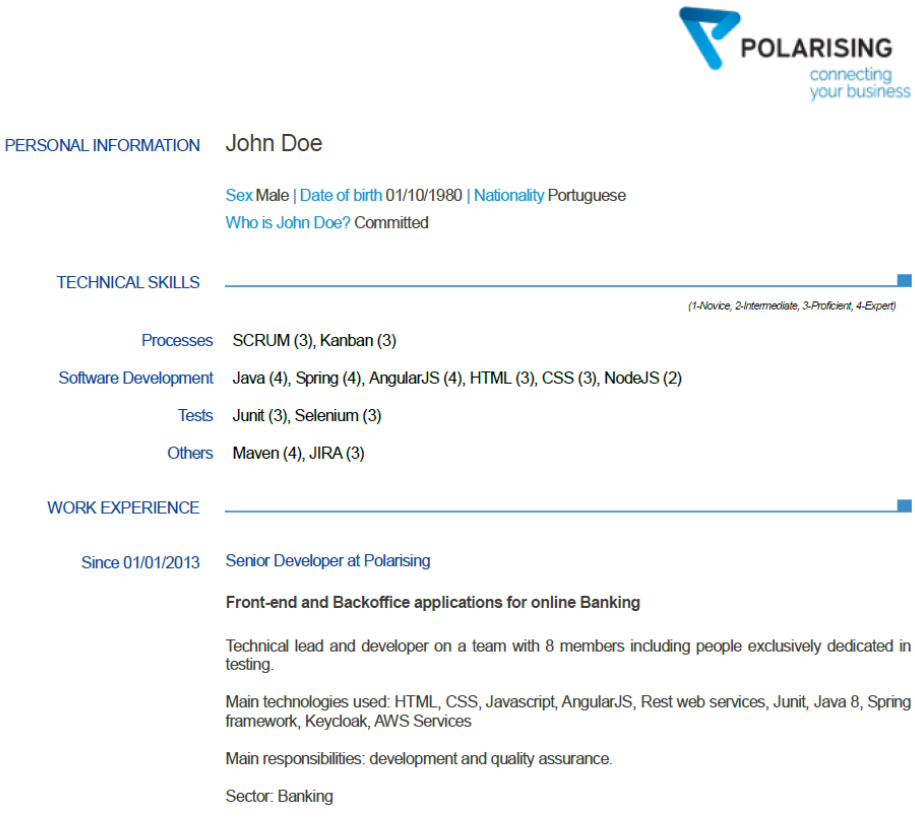
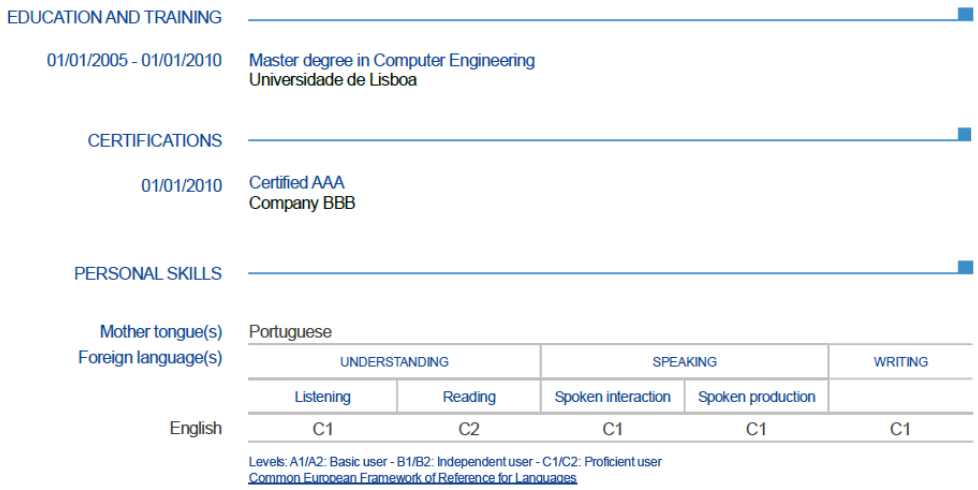


Figure 4.8 - Structure of a Polarising CV I


EDUCATION AND TRAINING

01/01/2005 - 01/01/2010 Master degree in Computer Engineering
Universidade de Lisboa

CERTIFICATIONS

01/01/2010 Certified AAA
Company BBB

PERSONAL SKILLS

Mother tongue(s) Portuguese

Foreign language(s)	UNDERSTANDING		SPEAKING		WRITING
	Listening	Reading	Spoken interaction	Spoken production	
English	C1	C2	C1	C1	C1

Levels: A1/A2: Basic user - B1/B2: Independent user - C1/C2: Proficient user
Common European Framework of Reference for Languages

Figure 4.9 - Structure of a Polarising CV II

Based on this structure and discussing with Polarising's HR, we can select the following features and their range of values, characterizing each candidate:

- Personal Information
  - Age (Any number greater than zero)
- Technical Skills
  - Skills (Array of predefined skills, each skill belong to one of three categories and having a skill level from 0 to 4)
    - Categories:
      - Oracle
      - Software Development
      - TIBCO
    - Levels:
      - 0 - No skill
      - 1 - Novice
      - 2 - Intermediate
      - 3 - Proficient
      - 4 - Expert
- Work Experience
  - Experience (Array of predefined job positions, each position having a value equal or greater than zero corresponding to the years of experience for that job)
    - Job positions:
      - Analyst
      - Consultant
      - Developer
      - Software Architect
      - Team Management

- Education and Training
    - Educational level (Any number between 0 and 4)
        - Levels:
            - 0 - No education
            - 1 - High school
            - 2 - Bachelor's degree
            - 3 - Master's degree
            - 4 - Postgraduate doctoral degree
- Language Skills
    - Languages
        - Portuguese (Any number between 0 and 6)
        - English (Any number between 0 and 6)
        - German (Any number between 0 and 6)
        - Spanish (Any number between 0 and 6)
        - French (Any number between 0 and 6)
            - Levels:
                - 0 - Non-speaker,
                - 1 - Basic User (A1)
                - 2 - Basic User (A2)
                - 3 - Independent user (B1)
                - 4 - Independent user (B2)
                - 5 - Proficient user (C1)
                - 6 - Proficient user (C2)

For each predefined language, each level is applied to three categories: Understanding, Speaking and Writing.

From the json file, all CVs will be parsed, saving only the required information needed for further processing. Regarding this parsing operation, the system obtains all CVs in an internal structure. The internal structure can be represented by the class *CVParsed.java* (Figure 4.10).

```
public class CVParsed {

    private Long id;
    private float age;
    private float[] skills;
    private float[] experience;
    private float educationalLevel;
    private float[] languages;
    private int[] params;

}
```

Figure 4.10 - CV's internal structure class

Regarding the above internal structure, each CVParsed can be described by:

- **id** – Unique CV identifier
- **age** – Candidate's age derived from the CV's birthdate
- **skills** – Array of predefined skills where each value represents the experience level of a specific skill.
- **experience** – Array of predefined work experiences where each value represents the years of experience for a specific job position.
- **educationalLevel** – Number representing the academic level of a candidate
- **languages** – Array of predefined languages where each value represents a language level of a language component (reading, writing, understanding) for a specific language.
- **params** – Array of integers where each value represents the number of components for each attribute, for example: params[0] = 1 (age), params[1] = 73 (number of skills).

The last field (params) is used in order to make the system adaptable to changes in the number of components for each attribute.

Concerning the parsing operation, the CV's **id** is a previously defined number and the **age** attribute is calculated from the CV's birthdate. Both **skills**, **experience** and **languages** are extracted from the CV through the search of predefined keywords in a list of strings, if a certain keyword is present, then the candidate possesses a certain skill/experience/language with the corresponding attribute value. Finally, the **educational level** can be extracted directly from the corresponding CV's educational level field.

At last, from the internally structured CVs, a feature extraction process is performed obtaining a feature vector for each CV with all required attributes to characterize an individual (Figure 4.11). The process runs for each attribute, parsing each attribute values to the feature array. Each attribute is normalized by a given coefficient and has different weights, representing the relevance each attribute has for the recruitment process. For example, work experiences and technical skills are considered much more important than the candidate's age or personal skills, having higher weights.

```
generateFeatureVector(cvParsed) {

    featureArray = cvParsed.getAge().getAgeWeight / ageCoefficient;

    featureArray = cvParsed.getSkills().getSkillsWeight / skillsCoefficient;

    featureArray = cvParsed.getExperience().getExperienceWeight / experienceCoefficient;

    featureArray = cvParsed.getEducationalLevel().getEducationalLevelWeight / educationalLevelCoefficient;

    featureArray = cvParsed.getLanguages().getLanguagesWeight / languagesCoefficient;

    return featureArray;
}
```

Figure 4.11 - Feature Generation pseudocode

4.3.2 Outputs

Firstly, the system will analyze the **overall attributes** of the candidate. To do so, Brain runs through the candidate's feature vector, identifying his strengths and main characteristics. Regarding technical skills, first, the system gets all the predefined categories and corresponding technologies. Then, for each category and based on the candidate's feature array, it calculates the average experience level and the sum of skills in that category. At last, it adds the current category's overall to the categories' overall evaluation list, returning it (Figure 4.12).

```
calculateCategoriesOverallSkillLevel(featureArray) {

    categoriesAndTechsList = getCategoriesAndTechs();

    for(categoryAndTechs : categoriesAndTechsList) {

        candidateAvgExpLvl = calculateExpLvlSum(categoryAndTechs, featureArray) / calculateNumberOfTechs(categoryAndTechs, featureArray);
        candidateNumberOfTechs = calculateNumberOfTechs(categoryAndTechs, featureArray);

        categoriesOverallList.add(categoryAndTechs, candidateAvgExpLvl, candidateNumberOfTechs);
    }

    return categoriesOverallList;
}
```

Figure 4.12 - Technical skills' overall characteristics calculation pseudocode

Concerning work experiences, for each job position, it sums the years of experience on that specific job.

```
calculateWorkExperiencesOverall(featureArray) {

    workExperiencesList = getWorkExperiences();

    for(workExperience : workExperiencesList) {

        yearsOfExperience = calculateYersOfExperience(workExperience, featureArray);

        workExperiencesOverallList.add(workExperience, yearsOfExperience);
    }

    return workExperiencesOverallList
}
```

Figure 4.13 - Work experiences' overall characteristics calculation pseudocode

Then, **similar collaborators** represent the list of collaborators whom the candidate resembles more. In order to obtain the referred list, the system calculates the shortest distances between the candidate and all collaborators during the KNN classification stage.

The distance is calculated as the squared distance between two points in the feature space (11).

$$distance = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \cdots (a_n - b_n)^2},\ a, b - \ feature\ vectors \qquad (11)$$

Concerning **rankings**, the system will rank a candidate regarding the available job positions (12).

$$ranking = \min\left(1, \frac{threshold}{distance(i,j)}\right),\ i - candidate,\ j - job\ position \qquad (12)$$

This ranking will be based on a chosen threshold and the distance between the candidate and the points representing the requirements for the given positions. If the distance between the candidate and a job position is less or equal than the threshold, the ranking is max (ranking = 1), if the distance is greater than the threshold the ranking is equal to the division between the threshold and the given distance. For example, the Software Engineer open position (Figure 4.14) will have the following attributes:

- Technical Skills
    - Intermediate user with SQL, JPA and Hibernate technologies (experience level equal or greater than 2)
    - Proficient user with the Spring framework, Angular JS, batch processing, HTML5 and CSS3 technologies (experience level equal or greater than 3)
- Work Experiences
    - 2+ years of experience as a Developer
- Personal Skills
    - Independent user in English (language level equal or greater than B1/B2)

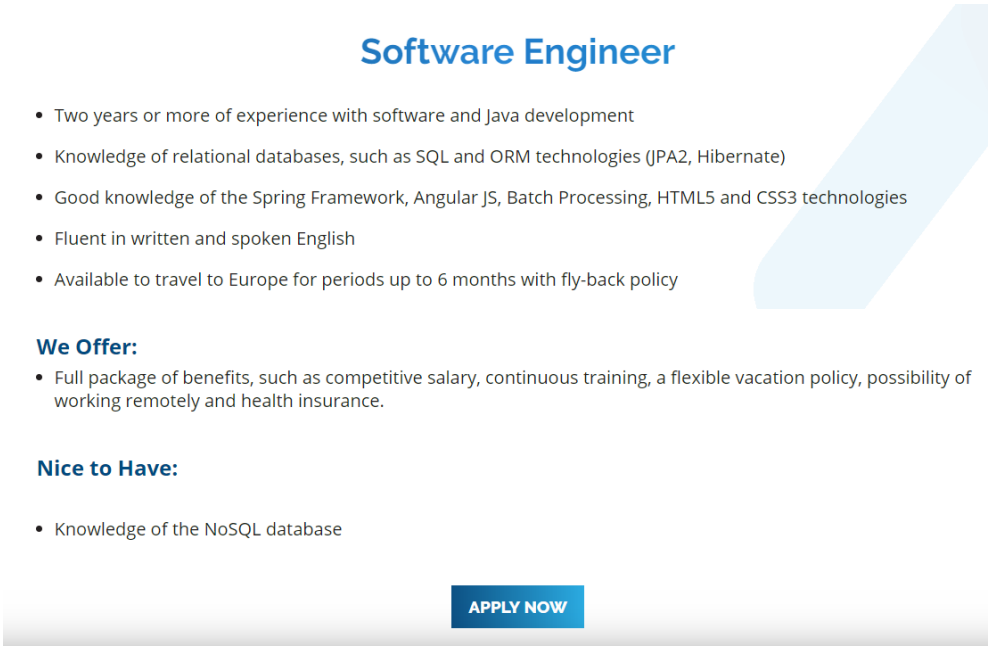


Figure 4.14 - Software Engineer open position [18]

The more attributes the candidate matches with the job position, the distance between the two points decreases and his ranking increases.

Lastly, the system will predict the candidate's class based on the KNN classifier, meaning which cluster is the most adequate for the candidate. Based on that class prediction, the **cluster characteristics** are displayed, similarly to the candidate's overall attributes regarding technical skills and work experiences. To do such analysis, the system instead of processing the candidate's feature vector, it processes the cluster centroid with the same strategy in order to obtain those overall characteristics.

After calculating all the above outputs, a response will be sent to the Candidates' web application, displaying the data to the recruiter.

4.3.3 System approach

4.3.3.1 Clustering

K-Means requires three parameters to be defined: number of clusters k, initial centroids and a convergence epsilon (stop criteria).

The k parameter determines how many clusters the algorithm finds, representing in how many groups the input data will be split. This work studies several choices of k, identifying which is more adequate given the dataset in hands. Based on that value, k initial centroids are chosen.

Regarding K-Means initial centroids there are two different strategies to consider: random centroids vs. chosen centroids. When considering random centroids, is possible to poorly generate initial points that result in a bad cluster distribution and increasing the clustering error. The **clustering error** can be defined as the sum of the distances between each element and the corresponding cluster centroid. Regarding chosen centroids, this approach might be better or worse than random centroids depending on the choice of initial centroids.

After the centroids are generated, K-Means starts the iterative process, checking each iteration if the algorithm converged (Figure 4.15). The class Mean can be described by figure 4.16.

```
kmeans(k, trainingData) {

    means = generateCentroids(k);

    for(i = 0; i < maxIterations; i++) {

        converged = kmeansIteration(means, trainingData);

        if(converged) {
            break;
        }
    }

    return means;
}
```

Figure 4.15 - K-Means Java implementation pseudocode I

```
public static class Mean {

    private float[] mCentroid;
    private ArrayList<float[]> mClosestItems;

}
```

Figure 4.16 - Mean Java class

Each iteration can be represented by the *kmeansIteration()* method (Figure 4.17).

```
kmeansIteration(means, trainingData) {

    for(mean : means) {
        mean.getClosestItems().clear();
    }

    for(instance : trainingData) {
        nearestMean = nearestMean(instance, means);
        nearestMean.getClosestItems().add(instance);
    }

    converged = true;
    for(mean : means) {

        mean.updateCentroid(mean.getClosestItems());

        if(distance(mean.getOldCentroid(), mean.getCentroid) > convergenceEpsilon) {
            converged = false;
        }
    }

    return converged;
}
```

Figure 4.17 - K-Means Java implementation pseudocode II

In each iteration, the algorithm clears the previous state before any computation. Then, each data point is added to their nearest mean. Afterwards, is computed the new centroids for each mean based on the average coordinates of each point in the corresponding cluster. At last, the algorithm calculates the squared distance between each new centroid and the previous iteration centroid, converging if all the distances are smaller than a convergence epsilon.

If one or more distances are higher than the stopping criteria value, the algorithm continues iterating until reaching convergence or the maximum number of iterations predefined.

Based on the feature vectors obtained from the inputs and pre-processing stage, K-Means finds the optimal distribution of Polarising's collaborators, labelling each instance to the identified cluster.

After the algorithm converges, a method of organizing the clusters' information is performed. This method consists of identifying, for each cluster, the overall characteristics as referred in section 4.3.2.

```
public class Cluster {

    private int id;
    private List<CategoryAndRating> categoryAndRatings;
    private float averageAge;
    private List<LanguageAndRating> languageAndRatings;
    private List<CV> cvs;
    private List<WorkExperienceRating> workExperienceRatings;

}
```

Figure 4.18 - Cluster Java class

For each mean, a new cluster with the above information is created. Then each field is calculated based on its centroid coordinates' values. For example, since each centroid coordinate is the average value of each point in the cluster, the average age of a cluster will be centroid's corresponding coordinate value.

```
arrangeClusters() {

    means = getMeans();

    for(mean : means) {
        cluster.setId(mean.getId());
        cluster.setAverageAge(calculateAverageAge(mean.getCentroid());
        cluster.setCategoriesAndRatingsca(calculateCategoriesOverallSkillLevel(mean.getCentroid()));
        cluster.setWorkExperienceRatings(calculateWorkExperiencesOverall(mean.getCentroid()));
        cluster.setCvs(calculateCvs(mean));

        clusters.add(cluster)
    }

    return clusters;
}
```

Figure 4.19 - Method of arranging each cluster's information

The arranged clusters' list and corresponding information will be later displayed on the system's web application.

After the clustering task is done and all CVs are labelled, the training data set is constructed and used to train the classifier.

4.3.3.2 Classification

First, a POST request [33] will be received by the system at a specific endpoint, representing a pending classification request. The request will contain the candidate's CV in the post body.

Then, the candidate's CV, stored in the POST body, will be parsed into the defined internal structure the same way each collaborator CV is parsed and, based on the parsing result, the system will generate the corresponding feature vector as referred in section 4.3.1.

From the candidate's feature vector and labelled training data obtained in the clustering stage, the system will perform the classification task, calculating all the required outputs.

As referred in chapter 2, K-Nearest Neighbors algorithm calculates the *k* closest points in the feature space to the instance to be tested, predicting its class based on the predominant class from the *k* neighbors. KNN algorithm considers only one user configurable parameter: number of k neighbors. Based on the selection of k, the algorithm finds the k closest points to the instance being evaluated.

```
knn(trainingData, candidate, k) {

    for(instance : trainingData) {
        distance = calculateDistance(candidate, instance);
        distancesByElement.add(instance, distance);
    }

    distances.sort();

    nearestNeighbours = distancesByElement.subList(0,k).getInstances();
    predictedClass = calculateMostFrequentClass(nearestNeighbours);


    output.setPredictedClass(predictedClass);
    output.setNearestNeighbours(nearestNeighbours);

}
```

Figure 4.20 - K-Nearest Neighbors Java implementation pseudocode

First, the system calculates the distances between the candidate and each instance in the training data. Then, the list with the distances and corresponding element is sorted ascendingly (smallest distances first). Based on the parameter *k*, the k nearest neighbors are selected. At last, the predicted class is calculated based on the most frequent class between the k elements (Figure 4.20).

Chapter 5

System Validation

5.1 Evaluation Metrics

In order to assess the quality of each strategy is necessary to define several evaluation metrics for the two system components: **clustering** and **classification**. Regarding the clustering component, each strategy is evaluated by the **elbow method**, **silhouette analysis** and **runtime**. On the other hand, the classification component is evaluated through its outputs, validating the results with several candidates against the company's reality and personnel.

5.1.1 Clustering

5.1.1.1 Elbow Method

The elbow method is useful to validate the number of clusters chosen for the k-means clustering. This method runs k-means on the dataset for a range of k values and, for each value of k, calculates the sum of squared errors (SSE) between each point in a cluster and the corresponding cluster centroid.

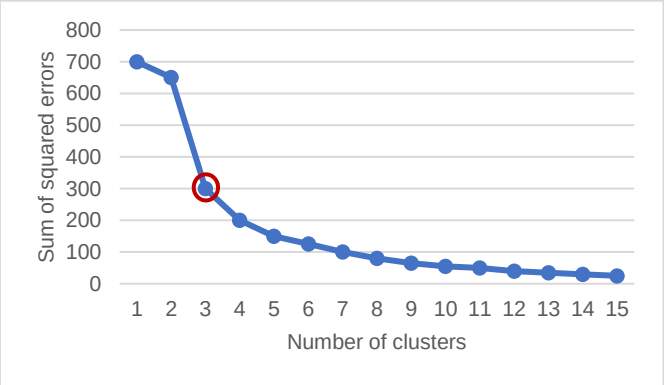


Figure 5.1 - Elbow method example

Then, with a line chart plot of the SSE for each k value we can identify the point that resembles an elbow (Figure 5.1). The goal is to find the smallest number of clusters that give us a low enough SSE. As we increase the number of clusters, the SSE decreases, reaching zero when the number of clusters equals the number of datapoints in the data set. However, the plot won't always show a clear elbow, showing more like a smooth curve. In this scenario, other methods may be needed for a better understanding of the problem, such as the computation of silhouette scores.

Besides helping to decide the number of clusters (K), the SSE for the corresponding K value chosen, can be a metric to compare different implementations.

5.1.1.2 Silhouette Analysis

Silhouette is a method that provides a graphical representation of how well each object has been clustered. This method comprise two notions: **cohesion** and **separation.** Cohesion measures how similar an object is to its own cluster and separation measures how similar an object is to the other clusters. The silhouette score/value ranges from -1 to 1, where a higher value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters. Similarly to the elbow method, the silhouette score for the corresponding K value chosen can be a metric to compare different implementations.

5.1.1.3 Runtime

At last, the clustering algorithm running time is evaluated, testing for different scenarios and input datasets size how the system responds.

5.1.2 Classification

5.1.2.1 Outputs Validation

Based on several candidates' scenarios created and the comparison against real company data, is possible to evaluate the classification stage correctness through the outputs obtained for each instance tested. This stage evaluation will be tested through the validity of the open position rankings, the similar list of collaborators and the predicted cluster obtained, considering the overall attributes of the candidate.

For the open position rankings, is expected that the candidate scores higher in positions where he matches the requirements better, and scores lower in positions where he possesses less required skills by the job offer. For the list of collaborators, is expected that, based on the candidate's attributes, it contains individuals with the same main characteristics regarding technical skills and work experiences. Lastly, similarly to the list of similar collaborators, is expected that the predicted cluster possesses the major characteristics the candidate has, obtaining an adequate classification for each candidate.

5.2 Case Studies

5.2.1 Case Study I – Impact of Feature, Weights and Centroids Selection Strategy on the Clustering Stage

Every case study showcased in this sub-chapter regarding the clustering stage considers all available Polarising collaborators' CVs (~130 CVs), testing the system with real data in order to evaluate the system in a real world scenario.

In order to fully evaluate the clustering component and establish a baseline for the system to start, several parameters need to be evaluated and compared relatively to the metrics used. The parameters are **features**, **weights, centroids selection** and **number of clusters**.

a) Features

As listed in the section 4.3.1, the base selected features are age, technical skills, work experiences, educational level and languages. Several combinations of features selection were performed, varying the number of clusters chosen and selecting random and defined initial centroids.

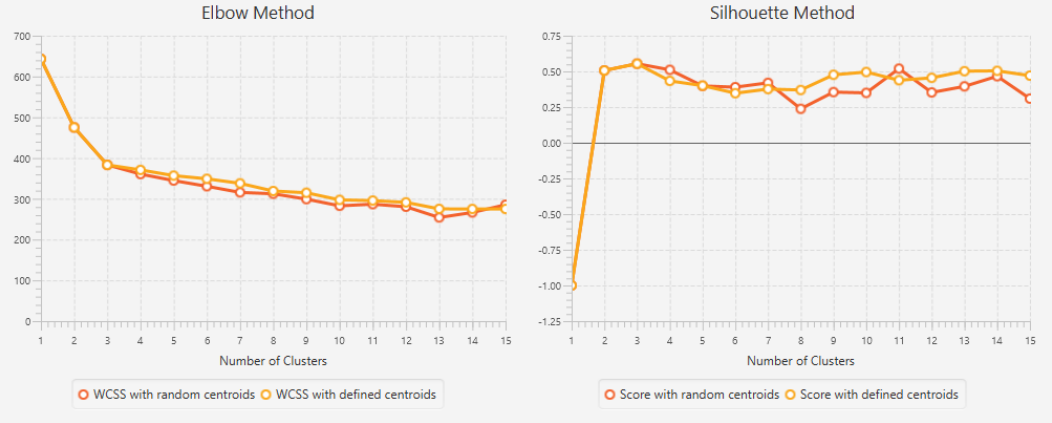


Figure 5.2 - Elbow and Silhouette methods selecting all attributes

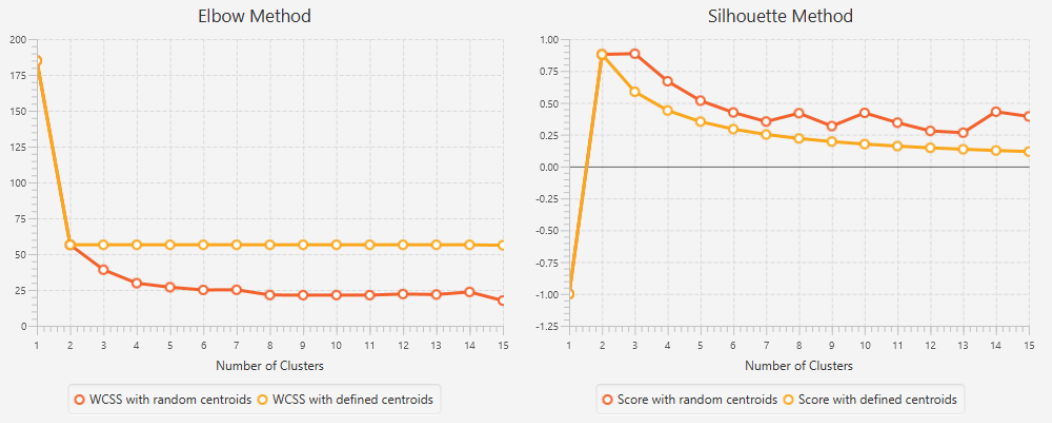


Figure 5.3 - Elbow and Silhouette methods selecting age, educational level and languages

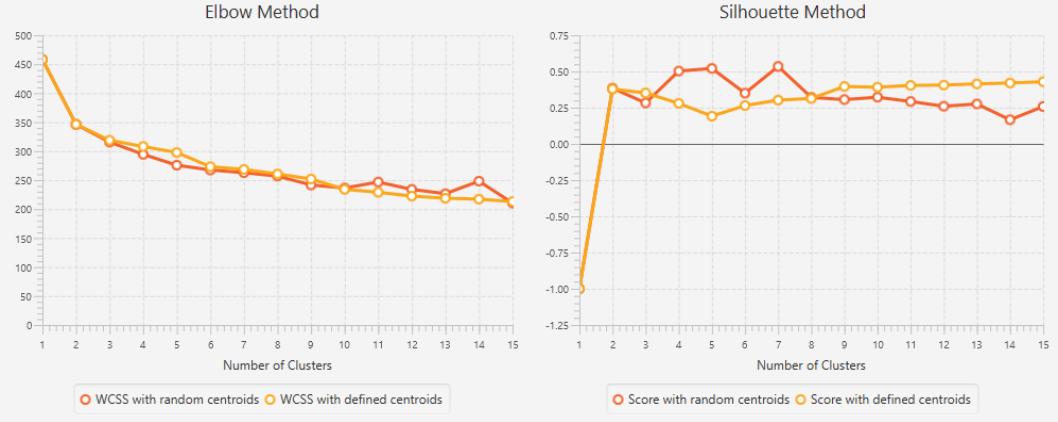


Figure 5.4 - Elbow and Silhouette methods selecting technical skills and work experiences

The results obtained show that selecting fewer attributes, the within cluster sum of squared errors (WCSS) decreases and generally the silhouette score is higher due to the decrease of the complexity of the problem. Moreover, the selection of technical skills and work experiences seems to be the reason behind higher clustering errors. Even though selecting the age, educational level and languages attributes obtains the smallest clustering errors, it provides no useful information since the focus is to understand the company's distribution regarding skills and experience.

b) Weights

Regarding the weights' selection tests, all base attributes are considered, varying only the importance of each feature in order to extract useful information of each configuration.

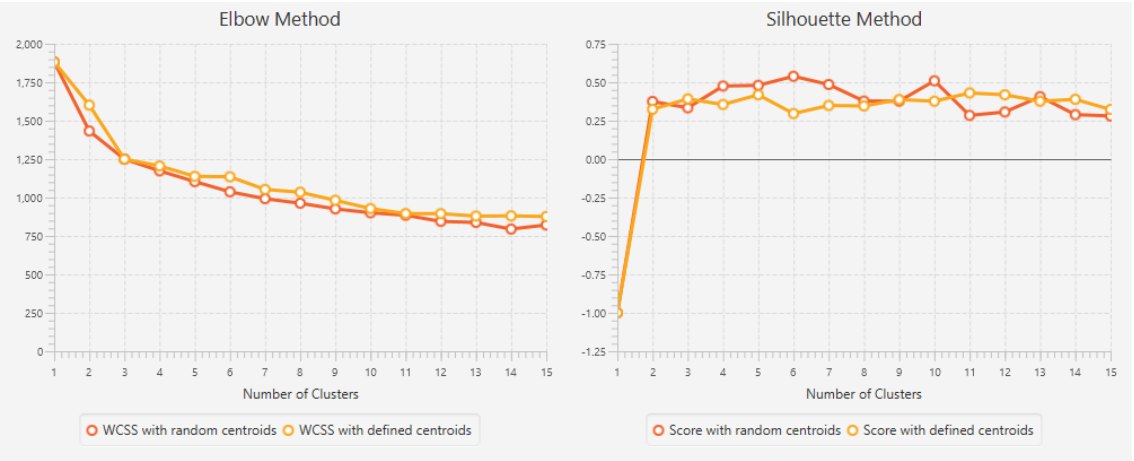


Figure 5.5 - Elbow and Silhouette methods with technical skills having twice the weight of the remaining attributes

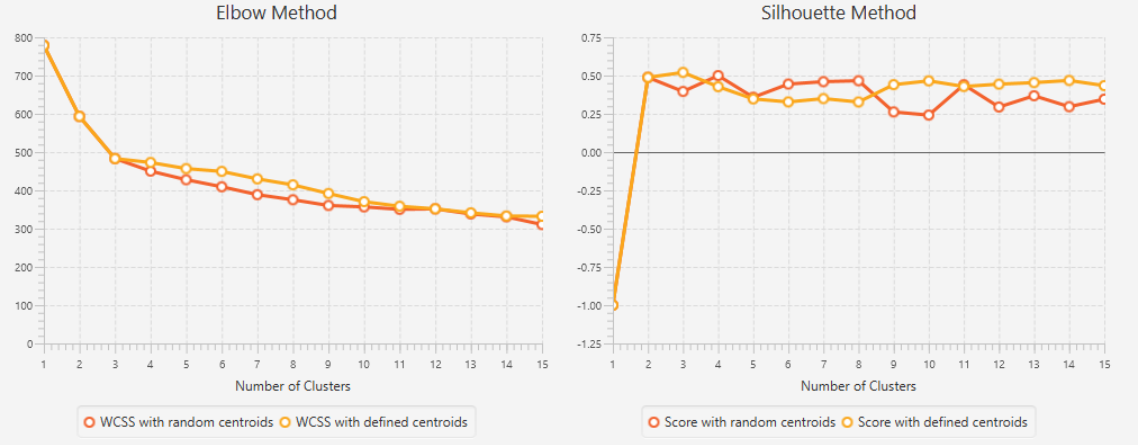


Figure 5.6 - Elbow and Silhouette methods with work experiences having twice the weight of the remaining attributes

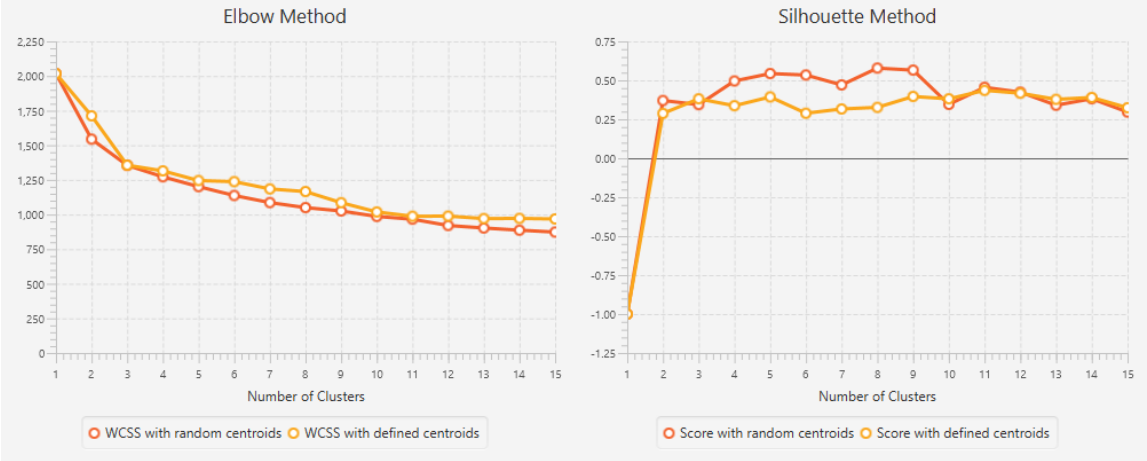


Figure 5.7 - Elbow and Silhouette methods with technical skills and work experiences having twice the weight of the remaining attributes

Based on the results above, increasing the importance given to certain attributes increases the within cluster sum of squared errors, not having a significant effect on the silhouette scores. Specifically, increasing the weight of technical skills, dramatically increases the WCSS due to the high number of attributes (~70) associated with this feature, increasing the dimension and the complexity of the problem.

c) Discussion

Overall, random initial centroids obtain lower WCSS values and higher silhouette scores since choosing adequate initial centroids is a hard problem.

Regarding the selection of the number of clusters to consider in K-Means, based on both Elbow and Silhouette methods, K = 4 or 5 seems to be the best choice.

However, it might be useful from a company's perspective to select a higher value of K in order to emphasize the distinction between collaborators in the corresponding K clusters. As observed by the results, increasing the weights of the attributes increases the within cluster sum of squared errors, however, it might be useful from a company's perspective to perform such evaluation.

5.2.2 Case Study II – Evaluation of the Classification Stage through the Testing of Fabricated Candidates

In this case study is chosen the more adequate configuration for Polarising needs, also considering the results obtained from the previous case study - selecting only the technical skills and work experiences' attributes, both having the same weights. Regarding the KNN algorithm chosen parameter K, is selected to return the five nearest neighbors of the evaluated instance.

In order to validate the system's classification stage, several candidates were created, some based on certain Polarising's collaborators and other based on Polarising's open positions. The outputs will be evaluated regarding the similar collaborators, open position rankings and predicted cluster technical skills and work experiences.

Candidate I represents a real candidate that was subsequently hired. Candidates' II and III represent two real Polarising collaborators, whereas Candidates' IV and V were created to mimic two different job open positions and mainly evaluate the calculated rankings. Every list of similar collaborators is hidden in order to prevent the disclosure of private data regarding Polarising's personnel.

a) Candidate I

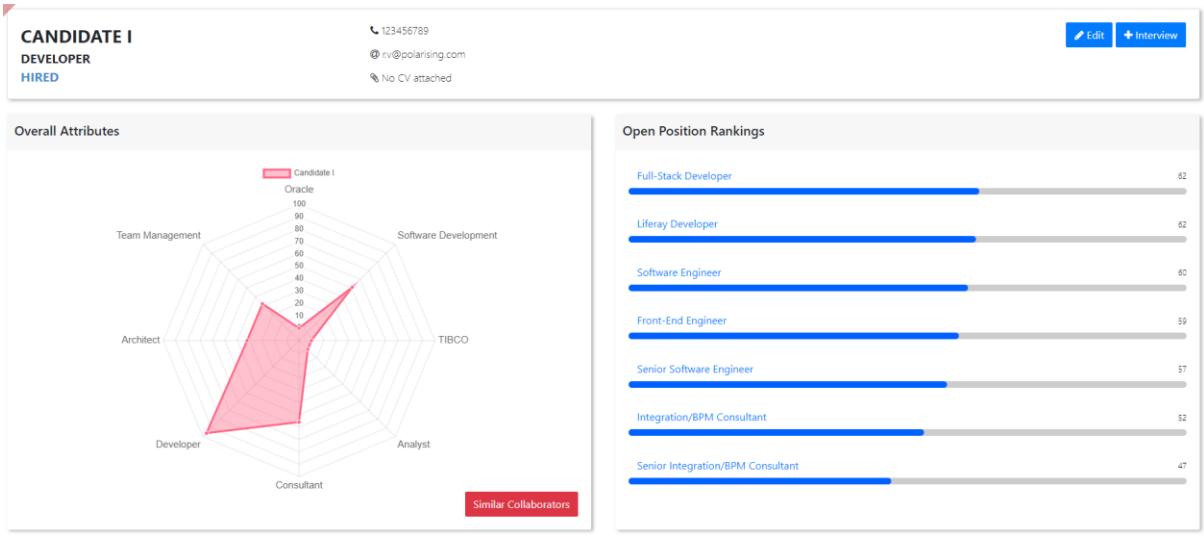


Figure 5.8 - Candidate I's detail page with Brain outputs

Candidate I is mainly characterized for having many skills in software development and a vast experience as a developer and consultant. Regarding similar collaborators, all individuals are adequate since they belong to the same areas of expertise and have identical years of experience.

Concerning the open positions, candidate I matches better with positions having mainly skills in software development, such as full-stack developer and software engineer, having lower scores in positions related to integration, such as integration consultant and senior integration consultant. However, based on the radar chart's observation, is clear that Candidate I has significantly more skills than required for the Full-stack developer job position, achieving an average score on that position.

Besides the rankings, calculated from the distance between the candidate and the open positions, the same can be observed through the radar charts (Figure 5.9). The more accurate the match is between the candidate and the open position; more likely he will be adequate for that given position.

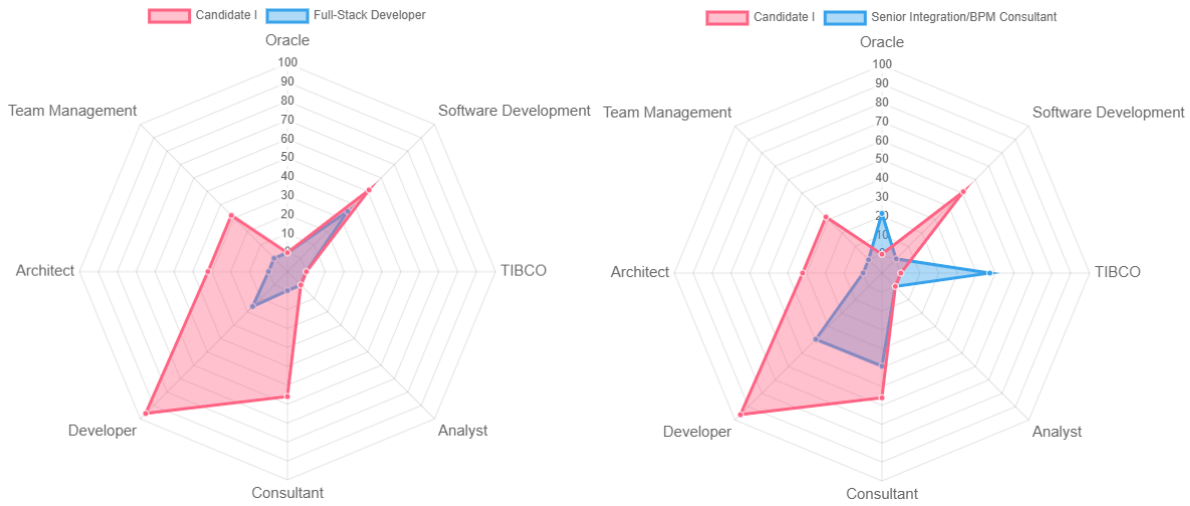


Figure 5.9 - Candidate I's comparison against different open positions

At last, regarding the predicted cluster, are displayed both cluster technical skills and work experiences. Comparing the candidate's attributes to the cluster specific information, Candidate I is placed in a group with a higher skill in Software Development and a vast experience as a Consultant and Developer, matching his attributes.

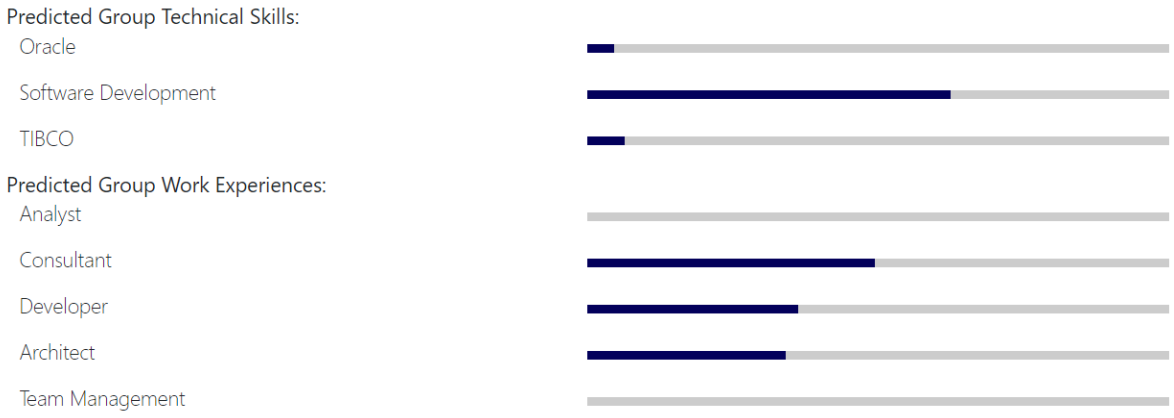


Figure 5.10 - Candidate I's predicted cluster overall characteristics

b) Candidate II

Candidate II has fewer overall skills and work experiences than the previous candidate, standing out his skills in Oracle and TIBCO technologies, along with some years of experience as a Consultant and Developer. Regarding similar collaborators, as said before, all individuals are adequate since they belong to the same areas of expertise and have identical years of experience.

Regarding the open position rankings, Candidate II obtains high scores for multiple job positions, highlighting the Integration/BPM Consultant open position. Comparing both radar charts, we can observe that the match is almost perfect, leading to a higher ranking. On the other hand, the Senior Software Engineer open position represents the lowest ranking score, confirmed by the distinction of both radar charts.

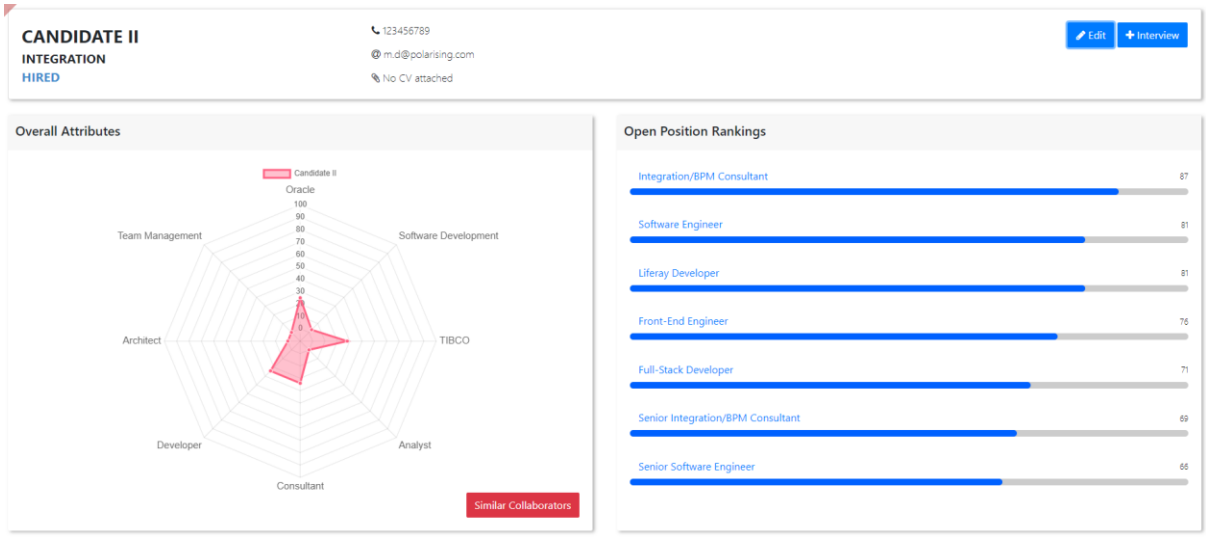


Figure 5.11 - Candidate II's detail page with Brain outputs

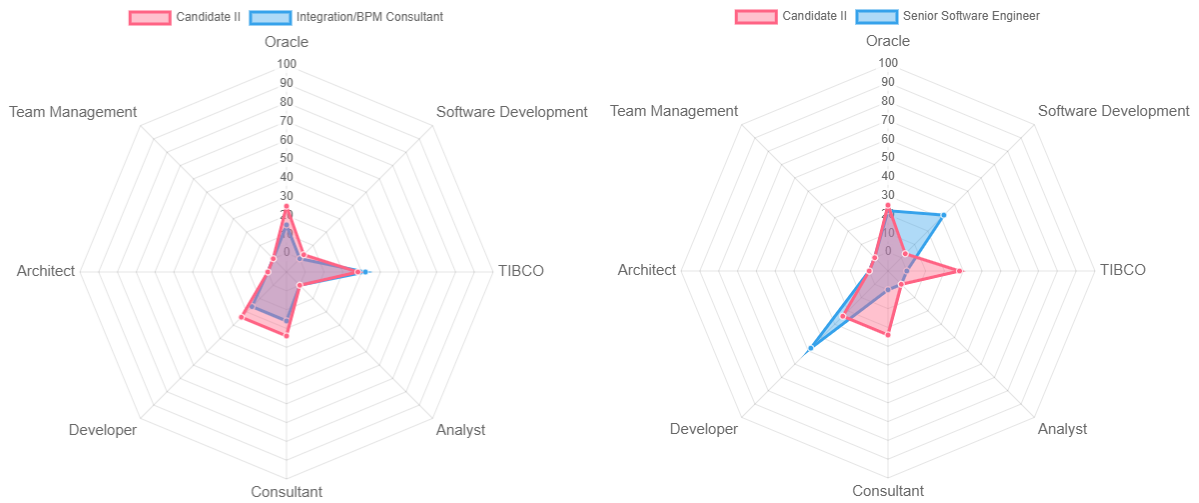


Figure 5.12 - Candidate II's comparison against different open positions

Lastly, Candidate II is placed in a cluster having some skill abroad all technical skills, highlithnig TIBCO technology, and some experience as a Consultant, Developer and Analyst. This cluster doesn't match all the Candidate's attributes, however, possesses his main characteristics, such as some skills in Oracle and TIBCO, along with some experience as a Developer and a Consultant.

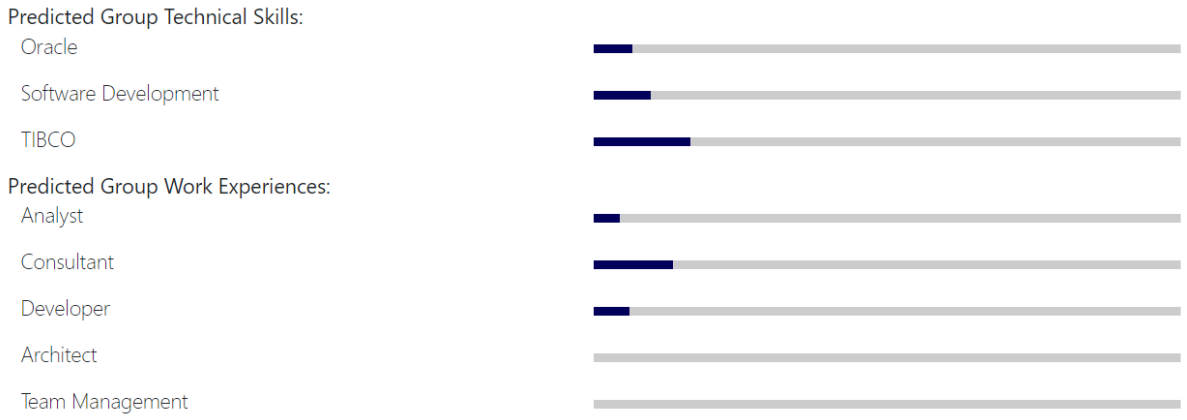


Figure 5.13 - Candidate II's predicted cluster overall characteristics

c) Candidate III

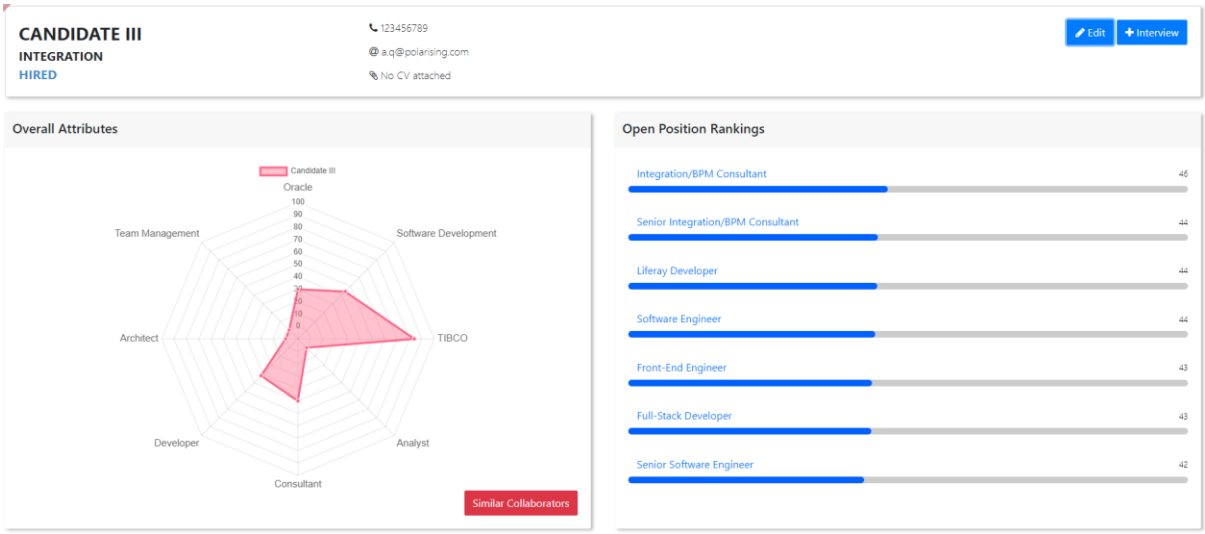


Figure 5.14 - Candidate III's detail page with Brain outputs

Candidate III is, among all fabricated candidates, the strongest in TIBCO technology, alongside with many years of experience as a Consultant and Developer. Regarding similar collaborators, all individuals are adequate since they belong to the same areas of expertise, such as Integration teams, and have similar years of experience.

Concerning the open positions, Candidate III stands out in jobs related to Integration, being mainly characterized by requiring knowledge in many different TIBCO technical skills. However, is clear that Candidate III possesses far more skills than required for any given position, justifying the lower open position rankings.

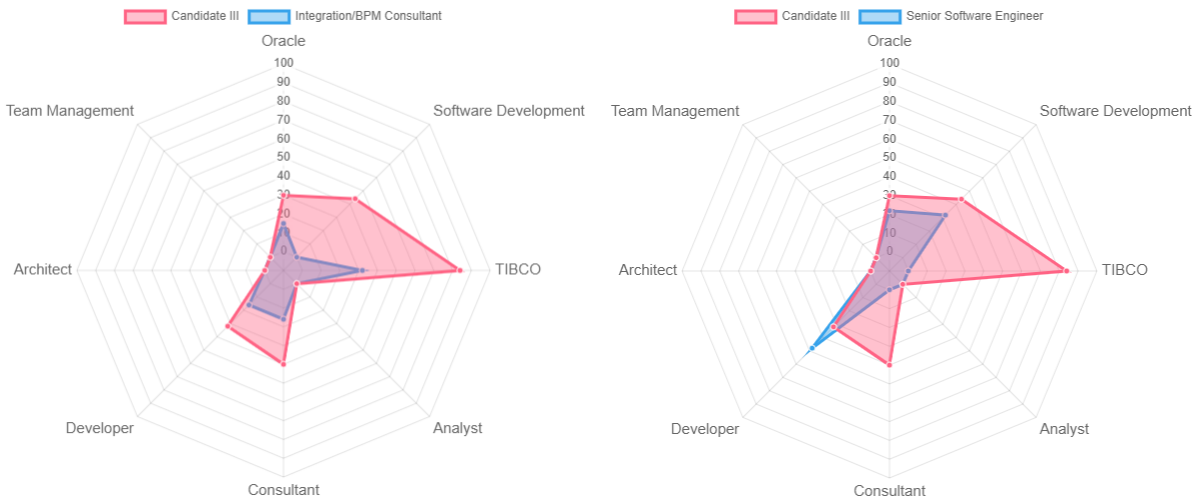


Figure 5.15 - Candidate III's comparison against different open positions

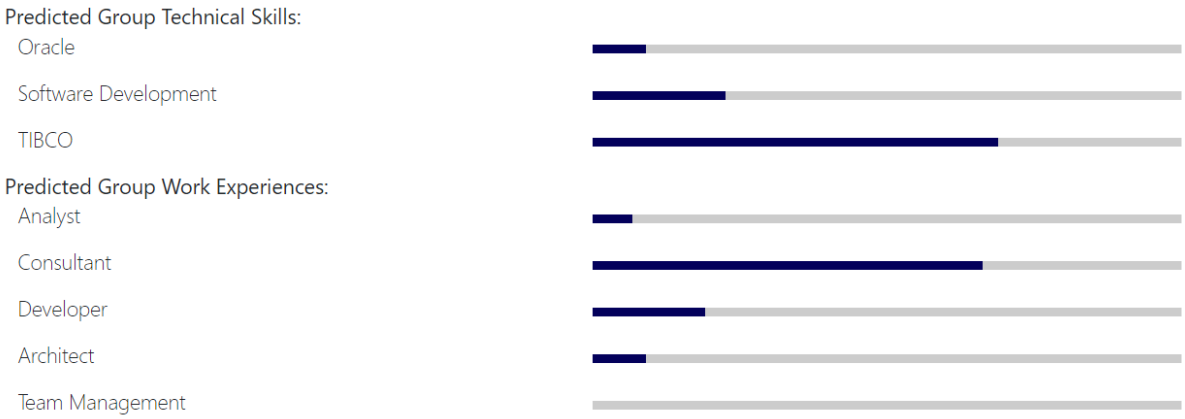


Figure 5.16 - Candidate III's predicted cluster overall characteristics

Observing the predicted cluster characteristics, it matches the candidate's overall attributes, such as predominant skills in TIBCO technology and Software Development, and experience as Consultant and Developer.

d) Candidate IV

Candidate IV is based on Software Engineer job descriptions, in order to validate the obtained open position rankings. This candidate has more skills in the Software Development category, a few in Oracle and a few years of experience as a Software Developer.

Regarding the open positions, as expected, Senior Software Engineer and Software Engineer achieved the highest rankings. This information being easily validated through the respective radar charts, displaying almost a perfect match between the candidate and the open positions.

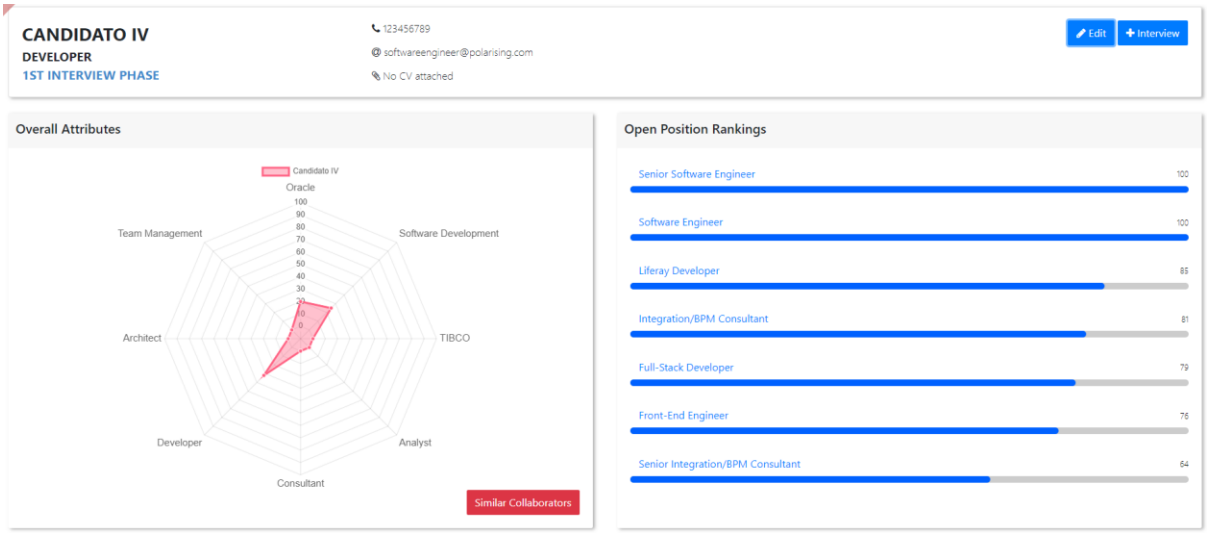


Figure 5.17 - Candidate IV's detail page with Brain outputs

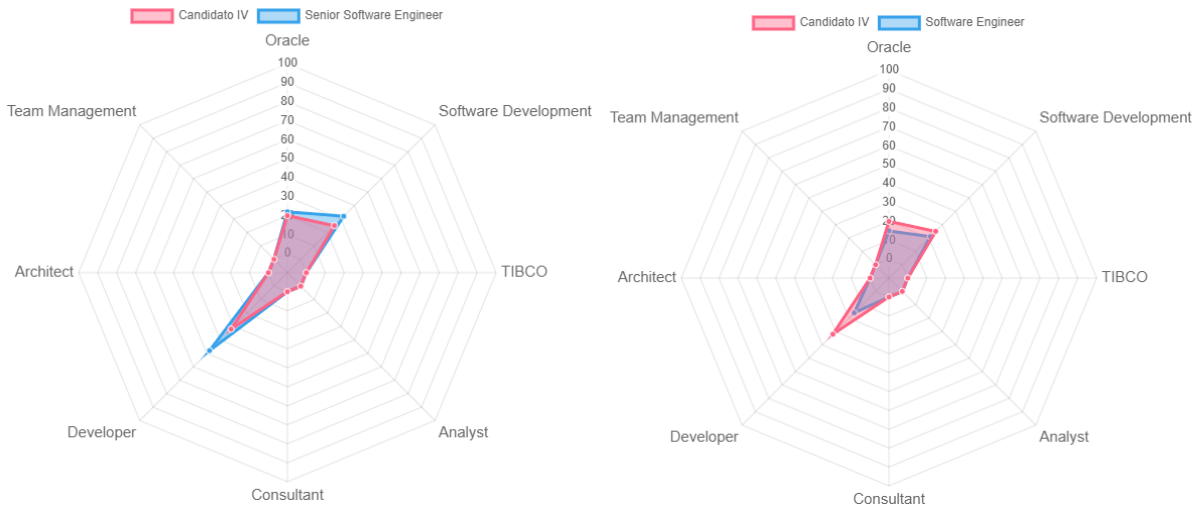


Figure 5.18 - Candidate IV's comparison against different open positions

e) Candidate V

At last, Candidate V is based on Integration job descriptions, requiring mainly TIBCO related skills.

Concerning the open positions, Senior Integration/BPM Consultant and Integration/BPM Consultant are the highest scoring positions. Based on the comparison between the candidate and the mentioned job positions, the match is almost perfect, validating the system analysis performed.

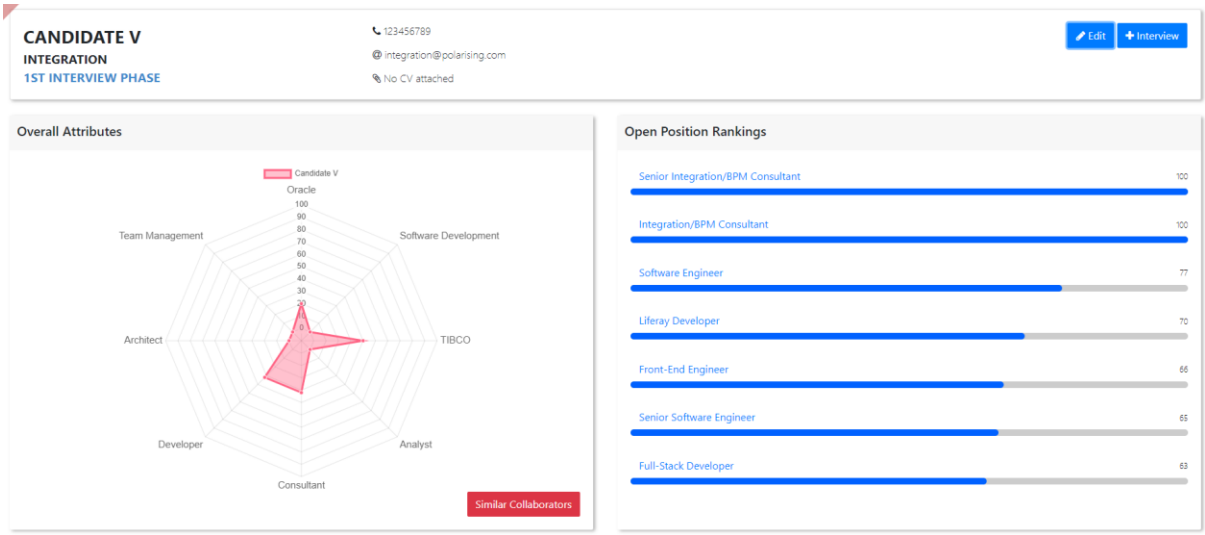


Figure 5.19 - Candidate V's detail page with Brain outputs

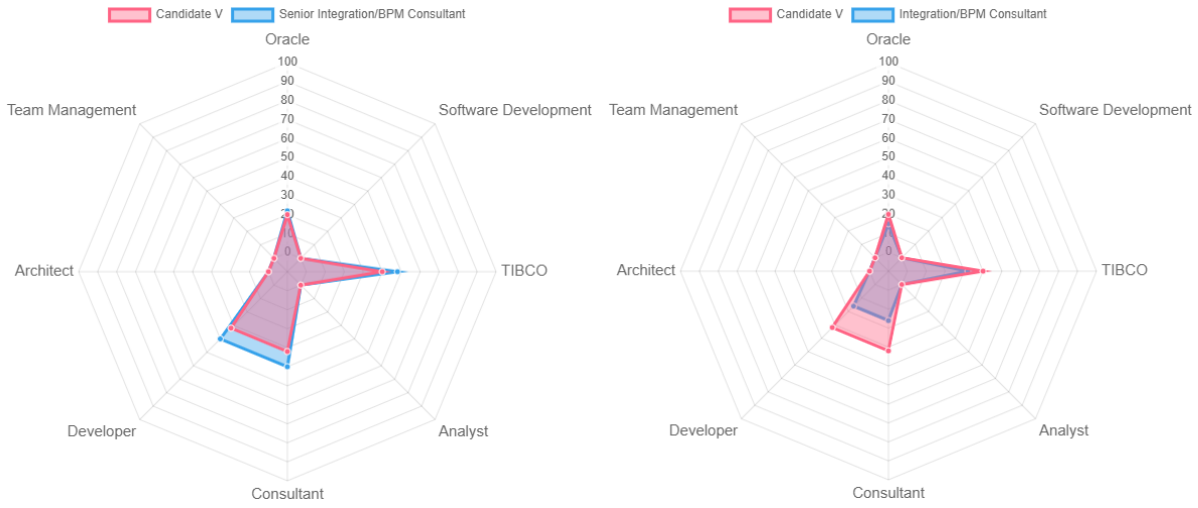


Figure 5.20 - Candidate V's comparison against different open positions

f) Discussion

Through the several tests performed, different system components were displayed and validated. First, the radar chart describing the main characteristics and attributes of a candidate provides a clear, quick and visual understanding of an individual. Then, the open position rankings explains how adequate a candidate is for each given job description, which can be also validated through the radar charts comparison between the candidate and the job. The similar collaborators gives insight on how a candidate compares himself with the rest of the company and the predicted cluster overall characteristics details the company's group where the candidate is more adequate to fit in. With all this information the recruiters have at their disposal tools that can significantly boost their process and increase its efficiency.

5.2.3 Case Study III – System Scaling

The proposed system is projected to process around 130 CVs as its input, representing the approximate number of Polarising's collaborators. However, is important to test the system to a bigger scale, analyzing the effects on the clustering task with a larger CV dataset. The classification task will not be tested since there is no need to request evaluation for more than one candidate at each time.

Regarding the classification task with random initial centroids, in order to prevent a bad initial state, the system runs the algorithm several times, each run with different randomly generated initial centroids, saving the scenario with the lowest clustering error. Doing so, obtains a better solution in regards of the company's distribution, however, has a significant time cost for larger input datasets, namely bigger companies.

Several tests were performed varying the input dataset size and observing the resulting clustering runtimes.

a) Runtime variation with random initial centroids

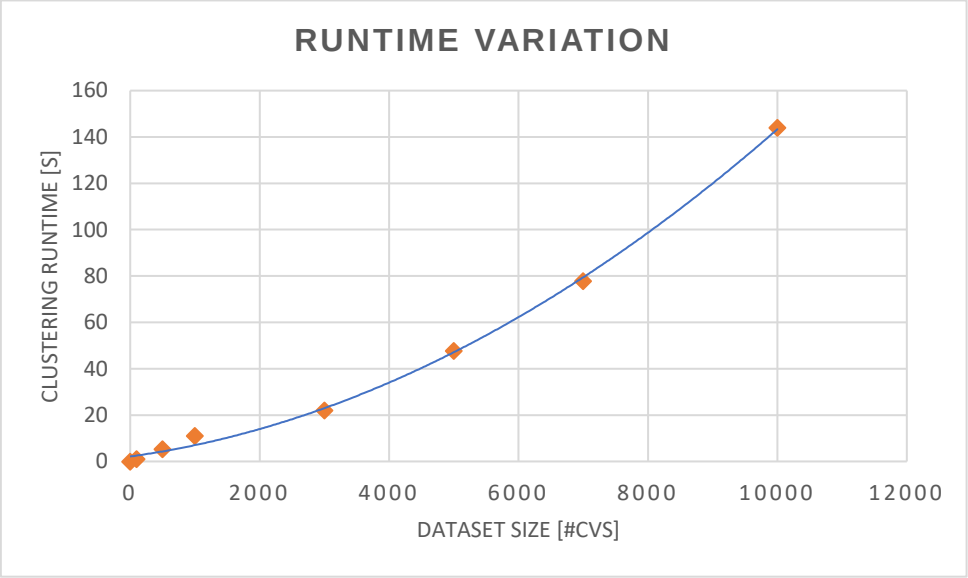


Figure 5.21 - Clustering runtime variation with random initial centroids

Based on the results obtained, the clustering runtime displays a quadratic curve in function of the dataset size. However, for a dataset of 10000 CVs, the clustering runtime is still acceptable with a period of approximately 2min20s.

b) Runtime variation with defined initial centroids

On the other hand, if the system considers defined initial centroids, there is no need to run the algorithm several times since the initial state will be the same. This way, the runtimes decrease drastically that, increasing the dataset size, the clustering runtime doesn't surpass 1s. Therefore, for a larger dataset, choosing the initial centroids is suggested.

c) Discussion

This case study shows that regarding Polarising's specifications and needs, the implementation with randomly generated initial centroids is better than the implementation with defined initial centroids because, as stated before, obtains lower clustering errors and similar clustering runtimes for a small scale. However, for larger input datasets, choosing random initial centroids may not be viable due to the quadratic increase of the clustering runtimes. In this scenario, defined initial centroids would be a more adequate approach.

At last, independently from the scale of the input dataset, the quality of classification task and the overall solution remains intact, considering valid input data.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

Considering the emerging difficulties that HR departments are facing nowadays with the overload of data they receive every day, machine learning techniques seem to be the most promising bet in order to handle this phenomenon. Hereby, the proposed solution intends to handle the immense number of CV's reaching Polarising everyday by building a decision support system based on machine learning techniques.

First, an historical overview was detailed, providing the contextualization of the evolution of HR and the recruitment process throughout the years. Then, several related works were studied and analyzed, describing the most common approaches and the background knowledge needed to fully comprehend the subject. Most systems can be categorized into two different strategies: clustering and classification. The clustering approach focused on understanding the underlying distribution of data, whereas the classification approach aimed at predicting if an individual is adequate or not for a job position.

Based on the related work, a system was proposed and implemented comprising two stages: A clustering stage where the system tries to structure, organize and label a set of data (the company), and a classification stage where the system, based on the labelled data obtained from the clustering stage, predicts the cluster of the instance (candidate) and several other valuable outputs. The clustering stage comprises a K-Means clustering algorithm while the classification stage contains a KNN classifier. Several tests were performed, testing the different components of the system with different initial assumptions, evaluating runtime and correctness. The optimal solution (minimal clustering error for useful outputs) regarding the clustering component was obtained by choosing random initial centroids and selecting only technical skills and work experiences as attributes, considering the size of company involved. Regarding the classification component, the proposed solution is based on KNN outputs. Considering K equal to five (five nearest neighbors) and the class prediction for a candidate, alongside

with two other outputs: a visual representation of the candidate's main characteristics and rankings relatively to the company's open positions.

The system outputs proved that the recruitment process has much to gain from machine learning, as it improves the process efficiency and the overall candidates' analysis by performing a more deterministic procedure.

Facing the solution obtained to other similar systems, this work stands out by performing a deeper analysis on each candidate, having user-friendly interfaces and providing the tools to achieve a clearer idea of the company's underlying structure and personnel.

6.2 Future Work

As stated, the integration of machine learning techniques into the companies' HR departments has a huge potential to significantly improve the recruitment process. However, there still some changes and tweaks that the proposed system can suffer to positively change the global recruitment process:

- **Generalization of the input format.** Instead of selecting a specific input format for the system to process, the system input may be specified with a generic text format, having a text data mining stage to derive high-quality information from the text and identifying the previously selected candidate attributes. Thus, the system's abstraction is significantly higher, increasing its adaptability. Yet, the text data mining stage will raise the system's complexity, as deriving useful information from text is way harder than extracting from a predefined structure.
- **Testing different clustering and classification algorithms.** Although the algorithms implemented already provide good results and the answers that Polarising was looking for, there is still room for improvement, with the testing of different clustering algorithms, such as Fuzzy C-Means or Mean-shift, and different classifiers, such as Naïve Bayes or C4.5.
- **Improvement of the outputs.** Although there are already several system outputs, with the addition of some new features, such as candidates' comparison, and with the improvement of the existing ones, the web application can still enrich the recruiters' everyday job.
- **Website candidate application.** At this stage, new incoming candidates are put up for evaluation through the candidate's web application by an HR recruiter. So, to automate the whole process, an online form available at Polarising's website [18] in each open position could trigger the whole evaluation process, not requiring human input.
- **Source of candidates.** In order to keep a predominant stance in the run for the best available candidates in the market, in the future the system could be improved by actively obtain data from other job offering/seeking websites such as LinkedIn, helping the recruitment process not only on the evaluation stage of a candidate, but also on the searching stage of new possible candidates.

References

[1] Dr. Ruth Tubey, Kipkemboi Jacob Rotich and Dr. Alice Kurgat, “History, Evolution and Development of Human Resource Management: A Contemporary Perspective”.

[2] Sophie Deering, “The Evolution of the Recruiter Throughout the History” [Online]. Available: https://theundercoverrecruiter.com/evolution-of-recruiter. [Accessed 22 October 2018].

[3] Marium-E-Jannat, Sayma Sultana Chowdhury and Munira Akther, “A Probabilistic Machine Learning Approach for Eligible Candidate Selection,” International Journal of Computer Applications, June 2016.

[4] N. Sivaram and K. Ramar, “Applicability of Clustering and Classification Algorithms for Recruitment Data Mining,” International Journal of Computer Applications, pp. 23-28, July 2010.

[5] Evanthia Faliagka, Kostas Ramantas, Athanasios Tsakalidis and Giannis Tzimas, “Application of Machine Learning Algorithms to an online Recruitment System,” presented at the seventh international conference on internet and web applications and services. 2012. pp. 215-220.

[6] Mayuri Verma, “Cluster based Ranking Index for Enhancing Recruitment Process using Text Mining and Machine Learning,” International Journal of Computer Applications, January 2017.

[7] Hamidah Jantan, Abdul Razak Hamdan and Zulaiha Ali Othman, “Human Talent Prediction in HRM using C4.5 Classification Algorithm,” International Journal on Computer Science and Engineering, pp. 2526-2534, 2010.

[8] Anca Apatean and Magnolia Tilca, “Machine-learning based application for staff recruiting,” Acta Technica Napocensis – Electronics and Telecommunications, vol. 58, no. 04, 2017.

[9] S.B. Kotsiantis, “Supervised Machine Learning: A Review of Classification Techniques,” Informatica 31, 2007

[10] Jiawei Han, Micheline Kamber and Jian Pei, “Data Mining Concepts and Techniques,” Third Edition, 2012

[11] Chen-Fu Chien and Li-Fei Chen, “Data mining to improve personnel selection and enhance human capital: A case study in high-technology industry,” Expert Systems with Applications 34, 2008

[12] M. Saidi Mehrabad and M. Fathian Brojeny, “The development of an expert system for effective selection and appointment of the jobs applicants in human resource management,” Computers & Industrial Engineering 53, 2007

[13] Evanthia Faliagka, Athanasios Tsakalidis and Giannis Tzimas, “An Integrated E-Recruitment System for Automated Personality Mining and Application Ranking,” Internet Research, October 2012

[14] Evanthia Faliagka, Athanasios Tsakalidis, Lefteris Kozanidis and Sofia Stamou, “A Personality Mining System for Automated Applicant Ranking in Online Recruitment Systems,” June 2011

[15] Vasudha Sarda, Prasham Sakaria and Sindhu Nair, “Relevance Ranking Algorithm for Job Portals,” International Journal of Current Engineering and Technology, October 2014

[16] Peter Keenan, Séan McGarraghy, Conor McNamara and Michael Phelan, “Human Resource Management DSS,” 2004

[17] Kevin Potter, “What is a GPA and Why Is It So Important?,” [Online]. Available: https://www.mastersportal.com/articles/2126/what-is-a-gpa-and-why-is-it-so-important.html [Accessed 4 January 2019].

[18] Polarising, [Online]. Available: https://www.polarising.com [Accessed 4 January 2019].

[19] Indeed, [Online]. Available: https://www.indeed.pt [Accessed 4 January 2019].

[20] LinkedIn, [Online]. Available: https://www.linkedin.com [Accessed 4 January 2019].

[21] “The K-means Clustering Algorithm,” [Online]. Available: http://kom.aau.dk/group/04gr742/pdf/kmeans_worksheet.pdf [Accessed 4 January 2019].

[22] R.Suganya, R.Shanthi, “Fuzzy C-Means Algorithm – A Review,” International Journal of Scientific and Research Publications, November 2012

[23] Badr Hssina, Abdelkarim Merbouha, Hanane Ezzikouri and Mohammed Erritali, “A comparative study of decision tree ID3 and C4.5,” International Journal of Advanced Computer Science and Applications, pp. 13-19, 2014.

[24] “Naïve Bayes Classification Algorithm”, [Online]. Available: http://software.ucv.ro/~cmihaescu/ro/teaching/AIR/docs/Lab4-NaiveBayes.pdf [Accessed 4 January 2019].

[25] Sergios Theodoridis and Konstantinos Koutroumbas, “Pattern Recognition,” October 2008

[26] B. Ustun, W.J. Melssen and L.M.C. Buydens, “Facilitating the application of Support Vector Regression by using a universal Pearson VII function based kernel,” Chemometrics and Intelligent Laboratory Systems 81, 2006

[27] Ryan J. Urbanowicz, Melissa Meeker, William LaCava, Randal S. Olson and Jason H. Moore, “Relief-Based Feature Selection: Introduction and Review,” Journal of Machine Learning Research, pp.189-203, November 2017

[28] Google, “Angular”, [Online]. Available: https://angular.io [Accessed 4 January 2019].

[29] Douglas Crockford, “JSON”, [Online]. Available: https://www.json.org [Accessed 4 January 2019].

[30] Purnima Bholowalia and Arvind Kumar, “EBK-Means: A Clustering Technique based on Elbow Method and K-Means in WSN,” International Journal of Computer Applications, pp.17-24, November 2014

[31] “Pearson’s Correlation Coefficient”, [Online]. Available: https://www.spss-tutorials.com/pearson-correlation-coefficient [Accessed 6 January 2019].

[32] Pivotal Software, “Spring Framework”, [Online]. Available: https://spring.io [Accessed 6 January 2019].

[33] Timothy John Berners-Lee, “HTTP”, [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview [Accessed 27 March 2019].

[34] Pivotal Software, “Spring Modules”, [Online]. Available: https://docs.spring.io/spring-framework/docs/3.0.0.RC3/spring-framework-reference/html/ch01s02.html [Accessed 28 March 2019]

[35] Julie Dubois, “JHipster”, [Online]. Available: https://www.jhipster.tech/ [Accessed 29 March 2019]

[36] Polarising, [Online]. Available: https://www.polarising.com/careers/ [Accessed 1 April 2019]

[37] Pivotal Software, “Rest Template”, [Online]. Available: https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/web/client/RestTemplate.html [Accessed 3 April 2019]