Universidade de Aveiro
2021

Gabriel
Silva

Algoritmos de Análise de Alinhamento de Ofertas do Mercado de Trabalho

Universidade de Aveiro
2021

Gabriel
Silva

Algoritmos de Análise de Alinhamento de Ofertas do Mercado de Trabalho

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Informática, realizada sob a orientação científica de Doutor Mário Rodrigues, Professor Adjunto da Universidade de Aveiro e de Doutor Armando Pinho, Professor Catedrático do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

o júri / the jury

presidente / president	**Doutora Ana Maria Perfeito Tomé** Professora Associada, Universidade de Aveiro
vogais / examiners committee	**Doutora Liliana da Silva Ferreira** Professora Catedrática Convidada, Faculdade de Engenharia da Universidade do Porto
	Doutor Mário Jorge Ferreira Rodrigues Professor Adjunto, Universidade de Aveiro (Orientador)

agradecimentos / acknowledgements

Gostaria primeiramente de agradecer à minha família por me ter apoiado durante todo este ciclo de ensino superior e em segundo lugar à minha namorada por me ter aturado durante os altos e baixos que se foram apresentando durante este percurso.

Um especial agradecimento também aos Professores Mário Rodrigues e Armando Pinho pela excelente orientação que deram durante este ano inteiro.

Palavras Chave Mercado de Trabalho, Ontologia, ESCO, Random Forest, Probabilidade Estacionária, NLP

Resumo

A adoção de tecnologias digitais tem prometido a aceleração e agilidade das atividades laborais, processos e modelos de negócio. No entanto, as promessas de ganho estão associados a uma forte necessidade de profissionais qualificados que possuam a capacidade de aplicar o potencial da tecnologia de forma eficiente.Os contextos de trabalhos estão a ser remodelados à medida que novos modelos de interação e integração humana e tecnológica evoluem. Para ser possível aumentar a prontidão do mercado de trabalho em contextos de rápida mudança, é importante que os intervenientes nas empresas, profissionais e responsáveis por políticas públicas estejam cientes da dinâmica e das necessidades do mercado de trabalho. Isto pode ser observado pela lista de anúncios de emprego, contudo, o seu elevado número, requer ferramentas eficientes com o objetivo de os analisar e simplificá-los para que, consequentemente, seja possível retirar conclusões atempadamente e corretamente.

Como as propostas de emprego possuem formulações distintas para postos semelhantes, dependendo das empresas que estão a contratar, surge o desafio de estabelecer um equilíbrio para a comparação de propostas de emprego. Neste trabalho, foi feita uma tentativa de mapeamento das propostas de emprego nas ocupações da ESCO (European Skills/Competences, qualifications and Occupations). ESCO é uma ontologia publicada pela União Europeia e as suas ocupações traduzem-se em cargos num determinado emprego e têm associadas as competências essenciais e facultativas ao exercício da ocupação. A ELK (Elasticsearch, Logstash, Kibana) stack foi usada para lidar com o grande volume de propostas de emprego. ELK é uma ferramenta estável que gere avultadas quantidades de dados e, a camada do Kibana, possibilita a rápida exploração dos dados e a criação de painéis de visualização. Os resultados mostram que a ELK stack é uma ferramenta adequada para providenciar uma interpretação visual das dinâmicas do mercado de trabalho.

Foram testadas várias formas de alinhamento entre ofertas reais de emprego e as ocupações ESCO. Os melhores resultados revelaram um f1-score de mais de 0.8 no mapeamento de ofertas de emprego de ocupação de nível 1 da ESCO e uma exatidão de 63.75% quando houve a tentativa de prever a ocupação de nível 5. Estes resultados estão alinhados com o estado da arte presente na literatura e são bastante promissores, especialmente quando comparado ao patamar inicial de 40%, e mostra que a ESCO é um bom candidato para estabelecer esse equilíbrio em que permite a comparação das dinâmicas no mercado de trabalho para ambientes distintos.

Keywords Labor market, Ontology, ESCO, Random Forest, Steady State Probability, NLP

Abstract

The adoption of digital technologies promises to accelerate the transformation and the agility of processes, work activities and revenue models. Yet, the promised gains come together with dramatic needs for qualified professionals who can effectively leverage the technology potential. Job contexts are being reshaped as new models for the interaction and integration of humans and technologies take shape. To increase the readiness of the job market in fast-changing contexts all stakeholders–companies, professionals, policymakers must be aware of the job market dynamics and needs. These dynamics can be observed from the collection of job announcements, but its high volume requires effective tools for analyzing and simplifying it to draw timely and correct conclusions.

As job announcements have distinct formulations for similar roles, depending on the hiring company, this raises the necessity of establishing a common ground for comparing the job offers. In this work, an attempt at mapping job offers to ESCO (European Skills/Competences, qualifications and Occupations) occupations is made. ESCO is an ontology published by the European Union and its occupations are job positions with the mandatory and optional skills associated. ELK (Elasticsearch, Logstash, Kibana) stack was used for dealing with the high volume of job announcements. ELK is a stable tool that can manage large quantities of data and has an effective text search algorithm, the Kibana layer enables the rapid exploration of data and creation of visualization dashboards. Results show that the ELK stack is a suitable tool for providing a visual interpretation of the job market dynamics.

Several strategies were tested to align real job offerings with ESCO occupations and the best one revealed an f1 score of over 0.8 in mapping job offers to level 1 ESCO occupations and an accuracy of 63.75% when trying to predict the level 5 Occupation. These results are comparable to the state-of-the-art and are very promising, especially when compared to the baseline of 40%, and shows that ESCO is a good candidate as common ground to enable the comparison of job market dynamics for distinct environments.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Digitalization is one of the most popular topics in the agendas of companies and policymakers. For companies, the adoption of digital technologies promises to accelerate the interconnectedness and the agility of processes, work activities and revenue models [34]. Yet, the promised gains come together with dramatic needs for qualified professionals who can effectively leverage the technology potential. Job contexts are being reshaped as new models for interaction and integration of human talent and technologies take shape, and as enormous volumes of data are generated at every operation [14].

The demand for qualified professionals that are able to make sense of such accelerated and interconnected work systems is already evident, notably in the explosive growth of job announcements in social media and online platforms. The challenge lies in the fact that many of the professional profiles are in growing demand and require new qualifications and unique combinations of technical and behavioural competences that are hard to find in the current job market [6, 19]. Several of these new jobs, such as app developers, cloud computing specialists, data scientists, drone operators, data brokers, cyber security specialists, among many, were nonexistent in the last decade. For both companies and professionals, this is a critical moment to make sense of the new professional profiles and demands that are emerging, in order to adjust recruitment and qualification and career strategies.

The popularity of online media for recruitment, and the abundance of job postings announced every day, offers a rich setting to learn about the characteristics of the new professional profiles that are in demand to meet the digitalization across all business sectors [37, 15]. Moreover, the understanding of the changes taking place in workforce demand is critical for the society as a whole, in order to develop detailed knowledge about the forthcoming skills requirements and support the development of adequate education and training programs in a timely manner. This study offers a contribution in such a direction. We developed tools that allow for the extraction and the analysis of large volumes of data from online job announcements, with the purpose of advancing in the understanding about the emerging demands for competences and skills. Hence, it brings together the rich information concerning the emerging demand for professional profiles that is scattered across new job announcements, and the existing structured knowledge about skills, competences, qualifications and occupations relevant for the EU labour market that is organized in ESCO [7], the European Skills, Competences and Occupations classification system.

The creation of a tool that automatically collects data and qualifies them to a given framework, in this case ESCO, would allow for comparison between regions or even countries all across Europe. This was the main goal of this study, i.e., to create a tool that offers the capability to detect trends of where the industry is moving towards and help managers build better job offers for candidates.

1.1 Objectives

The main objective of this work is the development of a tool to collect and automatically classify job offers, according to the ESCO framework in order to rapidly understand the most required skills at any moment.

The system should be able to fetch job offers from different regions and countries and automatically attribute them an ESCO occupation based on an algorithm adaptable to unforeseen job posts. It should also allow the creation of varied dashboards depicting distinct dimensions, analyse trends in jobs as of late, historically analyze how a certain industry is doing, among other options.

There is also another part of the system that should be able to, given a query, attribute it an ESCO occupation. This would be helpful to, for example, recruiters trying to build job offers for potential candidates. Having a tool like this, would allow managers and recruiters to create more accurate job offers when it comes both to the name of the position wanted for the company as well as the skills needed to fill out that position.

The development of such system makes use of many techniques, such as using different job offer sources API (Application Programming Interfaces), as well as creating its own, web scrapping and document-storage databases for the data collection system. For the development of the classification algorithm, when it comes to engineering the feature vector, knowledge of ontologies and how to navigate them (both vertically as well as horizontally) is needed. Some natural language processing techniques are also used, such as the semantic similarity, and the usual pre-processing pipeline of removing stopwords, tokenization and lemmatization of words. For the classification part of the algorithm, knowledge of machine learning algorithms, as well as Markov chains, is necessary.

For an user to have to test and visualize the work done there will be a visualization tool for the dashboards, where an user can access and create their own visualizations, as well as aggregate all of the created visualizations in a dashboard. For the sole purpose of testing the algorithm there, will also be a website where an user is allowed to enter queries and see the results according to the classification algorithm.

1.2 Dissertation Structure

This document will feature six chapters, where each chapter will feature a brief introduction and present a relevant part of this work. It provides the background knowledge needed to understand this system, introduces the state of the art, as well as related work done in this area of work showcasing their results and achievements, showcase how the work was developed and which steps were took to ensure the best possible results, talk about the results obtained as well as the comparing them to the state of the art, lastly, present the conclusions reached as well as talk about the future work to better develop the work here shown.

The chapters following this initial one will be the following:

- Chapter 2 - State of the Art and Background Knowledge

- Chapter 3 - Resources and Tools used
- Chapter 4 - Developed Work
- Chapter 5 - Results
- Chapter 6 - Conclusion

The second chapter, as the title implies, will present the state of the art for this area, as well as show the essential base concepts necessary to understand this work. This chapter will start off by talking about these base concepts and move one to a collection of studies done in the past that approach this topic of automatically classifying job offers or even curriculum's into frameworks.

The third chapter will focus on describing the several tools used when developing the work and the decision process behind using these instead of existing alternatives. A description of how these are used and their purpose when it comes to this work will be given.

On the fourth chapter, all the developed work will be presented. Starting from building up the data collection system, explaining the data model, where the data comes from, how it is transformed until the algorithm and the all that was tried when it comes to machine learning and to tuning the final model.

The fifth chapter will showcase the results achieved for both the data collection and visualization, as well as the algorithm created. Examples of dashboards will be shown for the "Aveiro" region of Portugal, as well as the best results the algorithms achieved when classifying the job offers into ESCO occupations, while comparing them to the state of the art.

To end this work, the "Conclusion" chapter will give a brief summary of the work developed, show the main results achieved and discuss the future work so that all that was here shown could be improved upon.

Chapter 2

State of the Art and Background Knowledge

The focus of this section will be to discuss the background knowledge needed to understand the work that will be presented further ahead in the document as, well as give a summary of the current state of the a art.

The chapter will start with the background knowledge and concepts about the semantic web and machine learning followed, by the state of the art and the related works.

2.1 Background Knowledge

The background knowledge section will mention ways of representing information and general information needed to understand the work that will be presented further ahead. Topics such as ontologies, machine learning algorithms, the framework used will be displayed in this section.

2.1.1 Semantic Web

The Semantic Web could be considered an extension of the existing, and widely used, World Wide Web. The Semantic Web concept means that content in the Web should be structured in such a way that both humans and machines can easily understand [2]. This is done by adding machine-interpretable metadata of the already published information and data, so that the machines can make similar connections that humans are already able to do.

Connected to the semantic web there is the linked data concept. Linked open data is a structured data modeled as a graph and distributed in such a way that allows for connections between different servers and databases. This concept enables people and machines to access data from different servers and make sense of it more easily. This takes the semantic web to another level, where instead of simply having linked documents, now there is linked information. Creating this type of data allows both humans and machines to interpret data in similar ways which is exactly the goal [8, 36].

Linked open data, usually, uses a defined structured data type to publish the data called RDF (Resource Description Framework). This standard offers a way to share graph data with both people and machines. Once this standard was recognized by W3C (World Wide Web Consortium), a large collection of tools started being built around it [4, 31].

The RDF is a standard used to describe web resources and data interchange. It works by defining relationships between data and objects through triples. These triples consist of three linked data pieces: a subject, a predicate and an object. Having such a simple and uniform structures that allows information to be express using simple statements, connecting pieces of information together, lets any resource be identifiable, disambiguated and interlinked [2, 31]. It allows resources to not be ambiguous, by treating everything as a resource and attributing it a unique identifier called an URI (Universal Resource Identifier). These work similarly to URL (Universal Resource Locators) which are, in fact, a form of URI, but for uniquely identifying webpages. Now, instead of identifying web pages, the goal of URIs is to have resources be uniquely identifiable by machines and serve as the building of data structures. Also, it helps knowing exactly which resource the machine is looking at and how it relates to other pieces of data.

Ontologies

Ontology has varied definitions in different fields, however, when it comes to the Semantic Web meaning of ontology, it is a document that contains the description of concepts and their relationships. They are meant to model the structure of a given system [18].

Ontologies usually have a taxonomy and inference rules. Taxonomies are what defines the concepts and their relationships, it is what gives the ontology structure and terminology, so while an ontology identifies concepts and relationships, the taxonomy formalizes these hierarchical relationships among the concepts and how they should be referred to. It allows "lower-level" concepts to obtain the properties of the "higher-level" concepts. The inference rules work to find these relationships and make ontologies a more powerful tool, for example, the author of [2] gives the following example: "If a city code is associated with a state code, and an address has a city code, the address will also have the state code associated with it".

Triples

RDF files are structured in triples. Each triples must have a subject, a predicate and an object and can be represented using XML (eXtensive Markup Language) tags. On the RDF file the triples make specific affirmations, such as the ones shown in Figure 2.1. In this Figure, the subject would be Tim, the predicate both "birthPlace" and "birthDate", with the objects being "1995-06-08" and London. However, in this case, London also contains an URI to some other value and is not just a literal value like the birth date.

This structure offers great simplicity and a natural way of describing things and, more importantly, a way that machines can understand and help them process these large amounts of data. Anyone can define new concepts and create new triples, as long as they create the appropriate URI for the subject, object and predicate.

Synsets

A synset is a group containing many words that can be used to describe the same concept. These are synonymous words that can freely be interchanged with one another between contexts [11]. One word can belong to several synsets for varied reasons.

When it comes to lexical terms these can have several relationships between themselves.

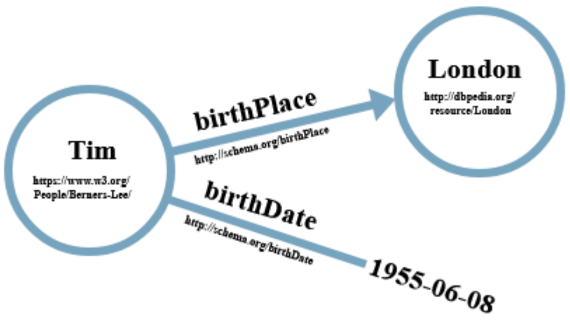

London
http://dbpedia.org/resource/London
birthPlace
http://schema.org/birthPlace
Tim
https://www.w3.org/People/Berners-Lee/
birthDate
http://schema.org/birthDate
1955-06-08

Figure 2.1: Example of two triples [21].

- Synonyms - Two different words that have the same meaning. For example: fine and good are two different words that have the same meaning.
- Hypernyms - Describe more abastract terms. For example: Flower is a hypernim of rose since flower can encompass many species.
- Hyponyms - Describe more specific terms. For example: Rose is a hyponym of flower since a rose is a type of flower.
- Polysemy - The capability of the same word to have one or more meanings. For example: Newspaper can refer to both the object we read or the company that prints it.

synsets take all these into account when building their structure making them a very powerful lexical tool.

2.1.2 ESCO

Creating an algorithm that can map occupations to job offers was set as a goal for this project. In order to do this, there was a need to find a framework that contained descriptions, titles and skills needed for each occupation, so that these could be mapped to job offers.

European Skills, Competences, Qualifications and Occupations (ESCO) is a multilingual classification of skills, competences, qualifications and occupations. It describes, identifies and classifies professional occupations, skills and qualifications inside the European Union labour market. ESCO describes 2942 occupations and 13485 skills linked to these occupations translated into 27 languages. The ontology works by having different levels of occupations from more "general" concepts such as "Professionals" to finer-grained occupations on the last levels like "Software Developer". Figure 2.2 shows an example of the ESCO structure. This is given as an ontology that is free to download and work with by everyone [7].

Besides this, once the algorithm is finely tuned, ESCO allows for easy comparisons between different countries or regions due to being a base model that classifies occupations. It also allows for the clustering industries and see how they are developing per country or per region.

With all this in consideration, ESCO seemed like the ideal framework to use when trying to match job offers to a certain occupation and it would also allow us to develop a more robust work and expand which directions we can follow.

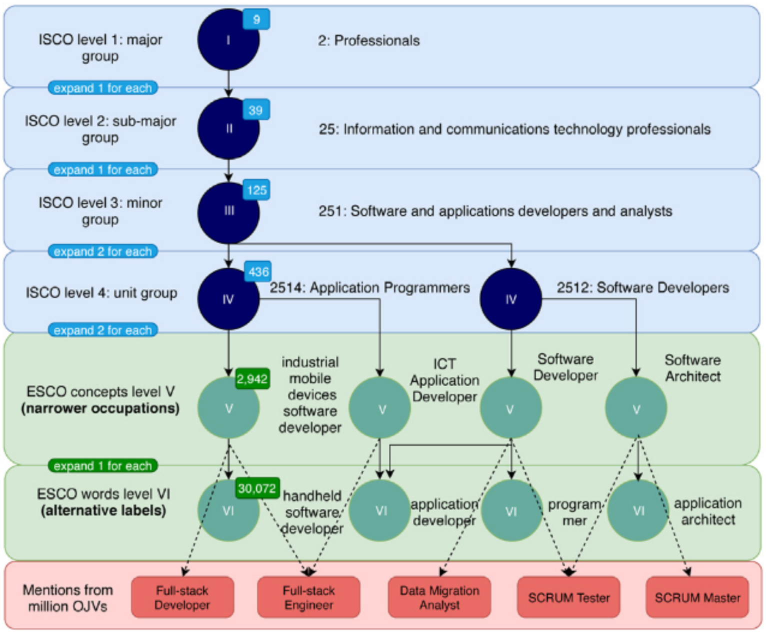

ISCO level 1: major group
9
I
2: Professionals
expand 1 for each
ISCO level 2: sub-major group
39
II
25: Information and communications technology professionals
expand 1 for each
ISCO level 3: minor group
125
III
251: Software and applications developers and analysts
expand 2 for each
ISCO level 4: unit group
436
IV
2514: Application Programmers
IV
2512: Software Developers
expand 2 for each
ESCO concepts level V (narrower occupations)
2,942
V
industrial mobile devices software developer
V
ICT Application Developer
V
Software Developer
V
Software Architect
expand 1 for each
ESCO words level VI (alternative labels)
30,072
VI
handheld software developer
VI
application developer
VI
programmer
VI
application architect
Mentions from million OJVs
Full-stack Developer
Full-stack Engineer
Data Migration Analyst
SCRUM Tester
SCRUM Master

Figure 2.2: Example of ESCO Structure [20].

In Figure 2.3 it is possible to see an example taken from the ESCO website of what an occupation looks like and the fields that come with the ontology, such as essential skills and competences, alternate names, etc. On the left side there is the path taken to reach "software developer" which is the fifth level occupation. The path taken to get to software developer is the following, in order from first to fifth: Professional, Information and communications technology professionals, Software and applications developer and analysts, Software developers and lastly Software developer.

2.1.3 Job Offers

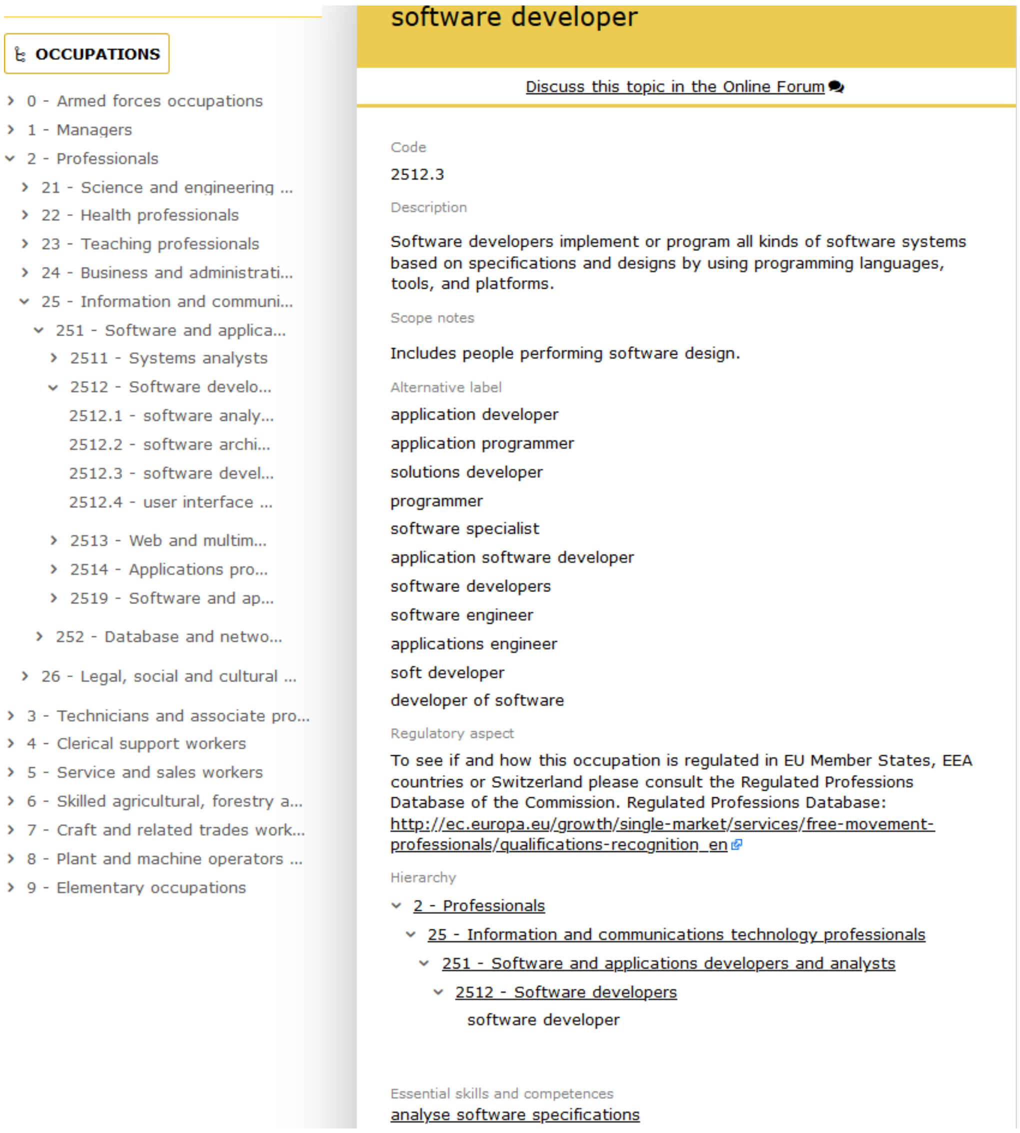
In order to develop a system that can collect and categorize data from job offers into different ESCO occupations, as well as create different dashboards, first there is a need to look into job offers and see what can be extracted from them.

Looking at Figure 2.4 as an example taken from the "CareerJet" website, job offer postings contain many fields that can be extracted. The job title, company offering the job, location, contract status, description and date when the offer was made are the first ones that pop out to the end user.

Most of this information can be obtained using the different companies public API as well as more information that is not right away displayed on their page.

For the dashboard creation part of this work, the information here seen is relevant. It allows the creation of several different dashboards that could be useful to analysing a country labour force or to compare regions of the same country. Things such as having more permanent contracts as opposed to timed ones, or number of jobs posting in the past X days, are useful for this purpose.

For the classification part, the decision was to try and use only the job offer title, in


OCCUPATIONS

- 0 - Armed forces occupations
- 1 - Managers
- 2 - Professionals
  - 21 - Science and engineering ...
  - 22 - Health professionals
  - 23 - Teaching professionals
  - 24 - Business and administrati...
  - 25 - Information and communi...
    - 251 - Software and applica...
      - 2511 - Systems analysts
      - 2512 - Software develo...
        - 2512.1 - software analy...
        - 2512.2 - software archi...
        - 2512.3 - software devel...
        - 2512.4 - user interface ...
      - 2513 - Web and multim...
      - 2514 - Applications pro...
      - 2519 - Software and ap...
    - 252 - Database and netwo...
  - 26 - Legal, social and cultural ...
- 3 - Technicians and associate pro...
- 4 - Clerical support workers
- 5 - Service and sales workers
- 6 - Skilled agricultural, forestry a...
- 7 - Craft and related trades work...
- 8 - Plant and machine operators ...
- 9 - Elementary occupations

software developer

Discuss this topic in the Online Forum

Code

2512.3

Description

Software developers implement or program all kinds of software systems based on specifications and designs by using programming languages, tools, and platforms.

Scope notes

Includes people performing software design.

Alternative label

application developer
application programmer
solutions developer
programmer
software specialist
application software developer
software developers
software engineer
applications engineer
soft developer
developer of software

Regulatory aspect

To see if and how this occupation is regulated in EU Member States, EEA countries or Switzerland please consult the Regulated Professions Database of the Commission. Regulated Professions Database: http://ec.europa.eu/growth/single-market/services/free-movement-professionals/qualifications-recognition_en

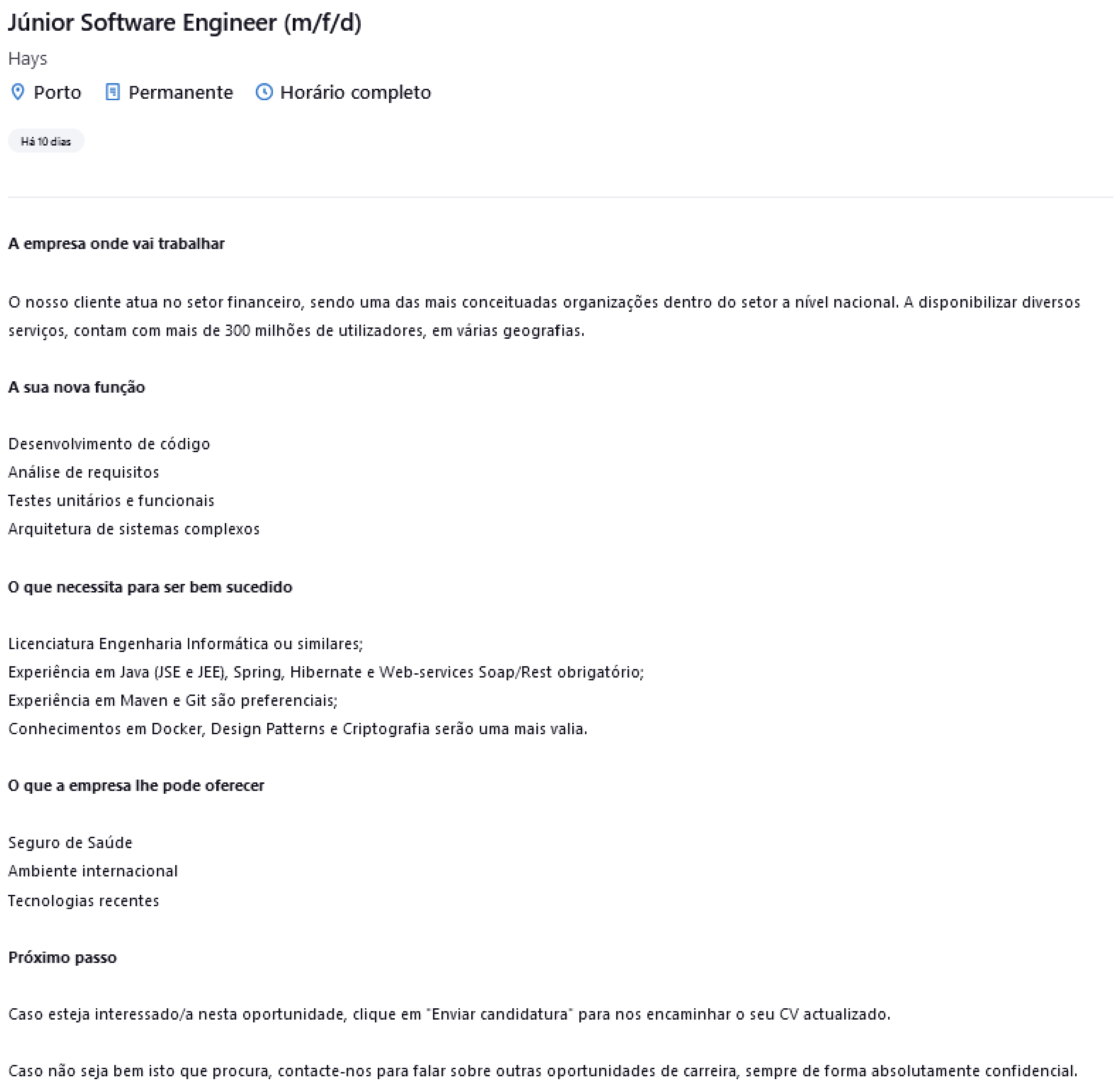
Hierarchy

- 2 - Professionals
  - 25 - Information and communications technology professionals
    - 251 - Software and applications developers and analysts
      - 2512 - Software developers
        - software developer

Essential skills and competences

analyse software specifications

Figure 2.3: Software Engineer occupation from the ESCO website [7].



Júnior Software Engineer (m/f/d)

Hays

Porto Permanente Horário completo

Há 10 dias

A empresa onde vai trabalhar

O nosso cliente atua no setor financeiro, sendo uma das mais conceituadas organizações dentro do setor a nível nacional. A disponibilizar diversos serviços, contam com mais de 300 milhões de utilizadores, em várias geografias.

A sua nova função

Desenvolvimento de código
Análise de requisitos
Testes unitários e funcionais
Arquitetura de sistemas complexos

O que necessita para ser bem sucedido

Licenciatura Engenharia Informática ou similares;
Experiência em Java (JSE e JEE), Spring, Hibernate e Web-services Soap/Rest obrigatório;
Experiência em Maven e Git são preferenciais;
Conhecimentos em Docker, Design Patterns e Criptografia serão uma mais valia.

O que a empresa lhe pode oferecer

Seguro de Saúde
Ambiente internacional
Tecnologias recentes

Próximo passo

Caso esteja interessado/a nesta oportunidade, clique em "Enviar candidatura" para nos encaminhar o seu CV actualizado.

Caso não seja bem isto que procura, contacte-nos para falar sobre outras oportunidades de carreira, sempre de forma absolutamente confidencial.

Figure 2.4: Job offer taken from CareerJet for a "Júnior Software Engineer (m/f/d)".

the case of the example ”Júnior Software Engineer (m/f/d)”. However, this can prove to be difficult seeing as the real world job offers even the titles contain information that is not relevant to the classification and there are, sometimes, misspellings.

API

An API is a piece of software that allows two applications to talk to each other. The purpose of an API is to answer a request made by a machine and return what was asked for, provided the correct parameters were given and it has authorization to do so.

These follow certain standards. Nowadays API are typically REST (Representational State Transfer), and are easily accessible and understood. The most common use for an API is to provide some type of data, given some parameters it was designed to take. These can be used internally or be exposed to the world for anyone to query and fetch data.

Web Scraping

Since some API do not return the full information of the job offer postings, for example, CareerJet truncates their descriptions past a certain number of characters, web scraping methods were used to extract the rest of the information from the job offer postings.

Web scraping refers to the practice of automatically extracting information and gathering data by any means that are not having a program interact with an API [25]. The most common form of web scraping is to fetch a web page HTML (HyperText Markup Language) code and write a program that will parse it and extract the relevant bits of information the user wants. This can be automated by having the program navigate between web pages automatically, requesting the HTML from the server, and parsing the files it gets back. Parsing this data can encompass many techniques such as, for example, NLP (Natural Language Processing), data analysis, information security, etc.

In general, if we can view the data on the web, it can be extracted using web scraping methods. These methods also offer the advantage of being able to cross-reference information from different websites and create new data from the extracted one.

2.1.4 Machine Learning

Machine Learning is a subset of Artificial Intelligence and it was defined by Arthur Samuel in 1959 as a field of study that gives computers the ability to learn without being explicitly programmed [29]. In today’s world, machine learning is becoming more and more relevant each day as the amount of data available increases tremendously and it would be very hard to analyze all of it without such techniques.

This is a very vast field that includes supervised learning (learning through pre-labelled inputs which act as targets) and unsupervised learning (training set does not contain any targets and the algorithm tries to group similar examples).

We used the following supervised learning algorithms:

- Random Forest
- Neural Network
- Decision Tree

- K-Nearest Neighbours (KNN)
- ADA Boost
- Support Vector Machines (SVM)

Although some of these were not used as extensively as others, they were all, at some point, tested and trained with. The metrics used to evaluate these models were their accuracy, precision, recall, and f1-score.

2.1.5 Document-Oriented Databases

Document-Oriented databases are part of the growing technology that is NoSQL Databases. These are seeing more common use cases everyday, especially when it comes to big data analytics. These have seen a rise due to data becoming more complex when it comes to both volume and structure (unstructured data is becoming more common) and the need for speed, the need for all of these makes NoSQL databases become more attarctive than more traditional Relational Database Management Systems (RDBMS) [26].

Document-Oriented databases saw their niche carved out in the NoSQL world by being able to store and retrieve semi-structured data. These are a subset of key-value store databases with the key difference being how data is processed. In Document-oriented databases the document structure metadata is used by the database engine to perform optimizations when it comes to search and retrieve speeds.

As the name suggests, these revolve around documents. These assume that data is encoded in a standard format or encoding, such as XML or JSON (JavaScript Object Notation). Not every document needs to have every field filed and some engines even allow for mix and matching of encodings, like uploading both XML and JSON files to the same database.

2.1.6 Markov Chain

The idea behind using Markov chains is to take advantage of the probabilities for each class given by the machine learning models. By using a stationary Markov model, it is possible to get a probability that each class is the correct one based on the matrix shown in Figure 2.6.

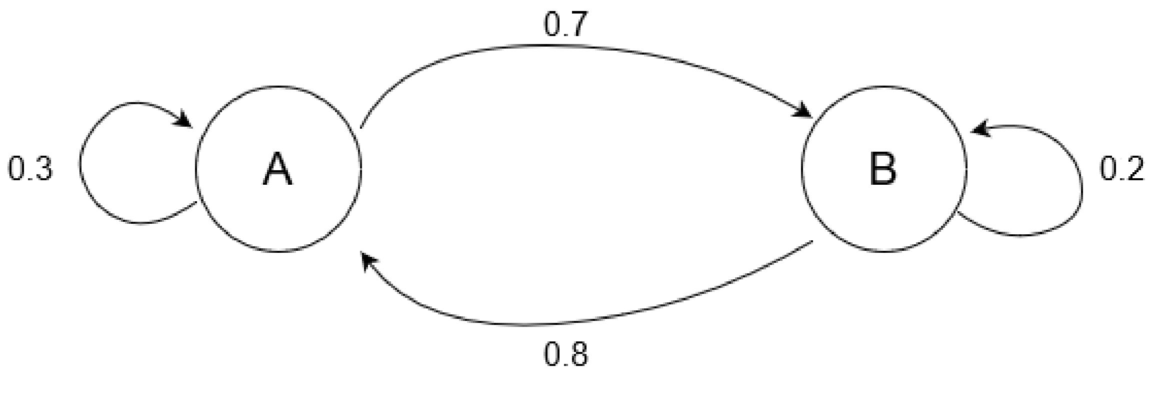


Figure 2.5: Example of a probability graph.

$$\begin{array}{c|cc} & A & B \\ A & 0.3 & 0.7 \\ B & 0.2 & 0.8 \end{array}$$

Figure 2.6: Probabily matrix of the graph shown in Figure 2.5.

With the objective of calculating the stationary probability of a Markov chain, a system of linear equations needs to be solved. Figure 2.5 shows an example of a probability graph to calculate the stationary probabilities. The first step is to convert the graph into a probability matrix, such as the one shown in Figure 2.6

Once we have obtained the probability matrix, the way to attain the stationary probabilities for this model is through matrix multiplication:

$$[p_A \, p_B] \begin{bmatrix} 0.3 & 0.7 \\ 0.2 & 0.8 \end{bmatrix} = [p_A \, p_B]$$

where, in the end, p_A and p_B are the stationary probabilities for this model. The way to solve this is to convert it into three different linear equations such as the ones following:

$$0.3\,p_A + 0.2\,p_B = p_A$$

$$0.7\,p_A + 0.8\,p_B = p_B$$

$$p_A + p_B = 1$$

Solving these equations will give us the stationary probabilities for both A and B.

When it comes to the problem at hand, the first step is to build the probability matrix as shown in Figure 2.6.

After getting the full probability matrix, the stationary probabilities are calculated, which ends up being a vector with the probability of each occupation being the "correct" one. Mixing the two models, machine learning and Markov, yielded better results, that will be presented in Chapter 5.

2.2 State of the art

In this section, literature related to either extracting job information from postings or resumes is presented, as well as articles using frameworks such as ESCO.

In [12], the authors develop a framework called eSkills Match. The framework attempts to align the existing work done by the European Union with the European e-Competence Framework (e-CF), ESCO and the Information and Communication (ICT) Body of Knowledge (BoK) into a single model. This model uses e-CF as the reference framework, tries to relate the BoK directly with it. The authors also use ESCO and relate it with the e-CF framework through skills and knowledges. In the end, the authors had a set of 10 ICT job profiles that were used to test the matching method with only 4 of these being present in the article. The authors identify the difficulties in matching different frameworks, requiring external validation from experts at times as well as the need to include more jobs.

In a second article [5] the authors attempt to extract information from resumes. The authors identify two steps needed to extract information from resumes which are not standardized neither in format nor layout. The first step takes as input a resume where the raw text is extracted followed by identifying the composition of each line and using adaptive segmentation in order to output semi-structured data comprised of several key-value pairs. To help generate this data, the authors design a novel feature called “Writing Style”. The second step is called “Resume Facts Identification”, where the previously generated semi-structured data is taken as an input, passed through a multiple category text classifier and the result is structured output resume data containing facts about the resume that was used as an input. The results are compared with PROSPECT and CHM, with this method beating both in almost every test done with the added benefit of needing less hours of annotations.

A different approach to extracting information from resumes and converting it into structured data can be seen in [38]. An ontology-driven algorithm is used to extract information and match it with concepts. The methodology is based on concept-matching and ontological rules so that semantic analysis and data parsing can be performed on the resumes. Extracting information such as experience, features and business/education information and converting it onto an ontology. The authors propose two algorithms that can be used to accomplish this task and test them on both English and Turkish résumés.

E-Recruitment systems try and give the best possible match between candidates and job descriptions. In [1], the authors propose a new approach to building these systems. The system created extracts the job description text and segments with the help of dictionaries (that are a combination of rules and lists). This segmentation then passes through a Natural Language Processing (NLP) pipeline (sentence splitting, part-of-speech tagging, named entities extraction and compound words extraction) they are then sent to the ”Context Builder” and ”Enrichment” processes before being stored in the Knowledge Base. This approach is compared with other information extraction software, such as OpenCalais and Alchemy, and beats them both by quite a margin in all the metrics used (precision, recall and F1) and tests done (job title and requirement extraction).

In the workshop [22] the author presents his work on several topics related to the job market, including classifying them according to ESCO. This presentation touches on the following topics: Occupation and Skill Discovery, Soft/Digital/Hard Skill rates, New Emerging Occupations, Training Course Design and Taxonomy Extension. It provides several key ideas on how to build these kind of systems and analyze the data.

GraphLMI [16] is an attempt at leveraging the emerging field of study called Labor Market Intelligence. The authors of this paper attempt to create a system that will organize labor market information as a graph and use this information to enrich and expand upon ESCO with emerging new occupations and skills, so that it better fits the labor market. Data was collected between 2018 and 2019 for 3 countries (France, Germany and United Kingdom). The resulting graph can be queried by experts to better understand the labor market dynamics and compare different markets. However, this work focused only on ICT related occupations and is being worked on to expand to other occupations.

Another paper that looks at ICT-related online job vacancies is MEET-LM [20]. In this paper, the authors define and formalise a method to evaluate and select embeddings that try and perserve domain-specific taxonomy. They also create a novel metric to evaluate ”hierarchical semantic relatedness between concepts of taxonomy”. To show the practical results of this work, the authors trained a neural network classifier with the purpose of classifying co-hyponym relations while using the embeddings extracted as features. The results show an accuracy of 99.4%, however, the f1-score is 86.5%.

Going back to skills, Skills2Job [17] is a proposed recommender system that looks at a user skillset and tries to identify the suitable job occupations for that user that are in a large database of online job vacancies. A dataset of over 2 million examples from 3 countries was used to train several embeddings and compute measures of skill importance for each occupation per country. Three models were built, one for each country, which are used to feed a graph database that which will serve as the heart of the recommender system. The final results look promising, achieving high precision values.

The work that is more directly related to the one developed in this study is shown on [23]. While this is meant as an introductory guide to using big data when it comes to labour market intelligence, the authors of this guide trained a model that does what this work aims to do. Match job offers to ESCO occupations. Throughout the guide the authors give an overview of the current state of the field and talk about several key topics when utilizing big data such as architecture components, state-of-the-art, the role of AI and plenty of other topics. For the comparable work done in this document, the authors present a box-plot with their results for classifying occupations into first-level ISCO groups (the same that ESCO uses). This was done by training an SVM on over 60 000 job vacancies and the claim is that it reached an overall accuracy of 90%.

Chapter 3

Resources and tools used

3.1 Introduction

The tools and resources used to develop this work will be presented in this chapter. From the stack used to store and visualize data to the libraries used for the machine learning models are going to be presented.

3.2 Tools Used

This section will focus on programs or stacks of programs used while developing the work. The extensively used ELK Stack will be the first one talked about, followed by GraphDB.

3.2.1 ELK Stack

The ELK stack is a powerful combination of tools namely, Elasticsearch, Logstash and Kibana. These tools allow data consumption from different sources, as well as allowing searching information and creating reports. While Logstash was not used, both Elasticsearch and Kibana were extensively used and now integrate the working prototype developed in this work. The process of processing data and creating meaningful and beautiful visualizations was made simpler due to these tools and what they allow users to do.

Elasticsearch

Elasticsearch serves as a storage for JSON documents and makes use of Lucene [9] for its search engine, allowing both storage and complex search of documents. It provides an HTTP web interface in order to perform searches or store and retrieve documents.

Elasticsearch in this project is used for both the storage potential, in addition to the search. It is used to store the transformed data that is retrieved from the different data sources, and the ability to get results from any field of the document is a key feature while dealing with ESCO. It also powers the visualizations that are made by kibana.

The decision to use Elasticsearch over other more popular document-storage databases was the integration in the ELK stack, making the creation of visualizations a smooth process, as well as being easy to manipulate with Python.

Kibana

Kibana is a visualization tool that can be used to create graphs and dashboards for Elasticsearch. Users can connect to an Elasticsearch cluster and create several visualizations (bar, line and scatter plots for example) on top of data. Kibana also has a "lens" feature that allows suggestions on which graphs would better suit the data that was selected.

Here, Kibana is used to analyze the job posting data from the different sources. It is used to answer several questions and create dashboards that help visualize current status of the job market. More examples of how Kibana is used and the questions it answers will be shown in Section 5.1.

Kibana provides several different modules that can be used to explore data, however, in this work not all of them were used. The modules used were the following:

- Visualize - Used to create different visualizations from the data gathered
- Dashboard - Aggregator of the differente visualizations created, provides an overall overview of the data.
- Discover - Used to check the data in tabular form and export it to .csv when needed.
- Lens - Similar to Visualize, the Lens module allows the creation of visualizations based on the data, however, lens provides suggestions and different types of graphs to choose from.

3.2.2 GraphDB

ESCO provides their data in an RDF (Resource Description Framework) format, which is a standard format for data interchange. In order to access it and pull data from the model, an engine capable of understanding this data format was needed.

The engine chosen was GraphDB. It allows easy querying of data by using SPARQL, which is the usual methodology used when dealing with ontologies, and visualization of the different relationship between attributes. It provides an easy-to-use API that allows data to be pulled from whichever model was imported onto it.

The ease-of-use of their API, as well as the simplicity of GraphDB, were the main considerations taken when choosing it. GraphDB was used initially before making the decision to try and upload ESCO onto Elasticsearch, however, this tool did not suite our needs, since it would not be efficient, because there is no prior knowledge about which classes or paths on the graph are relevant to the queried concept. It was useful to set a different plan on how to approach this problem, since the regular methods of navigating ontologies did not fit the needs of the project.

3.3 Wordnet

Wordnet is a lexical database, first developed in english, but now adapted to many other languages, that groups nouns, verbs, adjectives and adverbs into sets of synonyms also known as synsets (which were talked about before in Section 2.1). These synsets are organized and linked together by their semantic and lexical relations to form a network that can be queried [11]. Wordnet works by grouping up words according to their meanings.

The use of wordnet was done through the library NLTK (Natural Language ToolKit). This library also provides a way to find the similarity between two wordnet synsets, by calculating the Wu-Palmer Similarity between them, or using other alternatives of similarity metrics also supplied, and returning a score between 0 and 1, where a score of 1 means the words have the same meanings and 0 that they are completely unrelated. This similarity takes into consideration the depth of the two synsets in the WordNet taxonomies and the depth of the LCS (Least Common Subsumer).

3.4 NLTK

NLTK [3] is a Python library that supports a wide arrange of standard NLP procedures, such as text classification, tokenization, stemming, tagging, etc. It also includes plenty of datasets for users to test around with and interfaces to interacting with several corpora and lexical resources, such as wordnet and it does so not only using the english language which would better fit the point of this work.

There were two methods used from this library, the previously mentioned wordnet method, that provides an interface to work with wordnet and fetch synsets for different words, as well as allows to calculate the similarity between these by using the Wu-Palmer Similarity, and the stopwords method for removal of these stopwors from titles when needed. The library is thoroughly documented and for such a big and complex library it was easy to use, navigate and understand [3, 28].

3.5 Scikit-Learn

Scikit-learn or sklearn [27], much like the previously presented NLTK, is an open-source Python library that acts as an interface to a lot of methods, more concretely, related to machine-learning and data analysis. It provides resources to do supervised learning, unsupervised learning, model selection and evaluation, dataset transformation among many others.

This project made use of the supervised learning and model selection modules by using their implementation of the machine-learning algorithms, along with the gridsearch that was conducted to find the best parameters for each algorithm. Scikit-learn played an integral part of this work, by providing simple and clean interfaces to the algorithms that were used, a good documentation filled with examples on how to use them and an active, helpful community.

Scikit-learn is an extensive library that contains many methods and there were plenty used in this work. The three main categories of methods used were the following:

- Machine Learning Models - The different interfaces provided by sklearn to all the machine learning models that were tested (Neural network, SVM, ADA Boost, Random Forest, KNN and Decision Trees)
- Model Selection - Interfaces that provide tools to split the dataset. In this case the methods used were train_test_split and gridsearchCV.
- Metrics - Used to create confusion matrices of the models.

Further explaining on the Model Selection part, the train_test_split module was used initially to split the dataset into an 80/20 split of training and testing data respectively while the gridsearchCV allows for an exhaustive search of the parameters of the several different algorithms that were tested while also providing cross validation, in our case 5-fold. Both of these were integral when finding the best models for the algorithms that will be presented further ahead.

3.6 Tree Tagger

Tree Tagger [30] is a tool that was developed in 1994 with the purpose of annotating text with part-of-speech and lemma information. Tree Tagger is available in several languages and adaptable to other as long has there is a lexicon and a tagged training corpus. This tool takes words or sentences and returns individual words and their part-of-speech and the lemma.

The point of using a tool like this is to convert words into their lemma, for example, "is" would be converted into "be". It makes it so that the matching between ESCO and the job offers is easier by converting, what could be, different words representing the same concept into the same one or to a smaller set of forms. However, this implies that ESCO had to pass through TreeTagger and a lemmatized version be uploaded instead of the one provided by default.

While TreeTagger is given as a standalone tool to use through the command line tool with separate parameter files to download, we have used one of the several Python wrappers available for the command tool.

Chapter 4

Developed Work

4.1 Introduction

This chapter will be used to describe the work developed, along with how it was developed. It will showcase the main components and how everything was tested and fitted together.

There will be two main discussions in this Section, one to showcase the work done to collect data and process it, handling duplicates and converting different data sources into a single data model. The second Section will refer to the algorithms developed in order to match an ESCO occupation to every job offer, how the datasets were classified, how to match alternatives that were represented in the feature vectors and the experiments done with different machine learning algorithms.

4.2 Data Collection

Working with different data sources is always a challenging task. Every source has its own schema, attributes, and naming conventions. At the beginning of this project was set the goal of creating an adaptive system that could easily ingest data from any source. Instead of ingesting data from every API and adding the documents as they were onto the database system, the decision was to create a data model that these documents could be converted to. Figure 4.1 shows the flow of data in the system that was created.

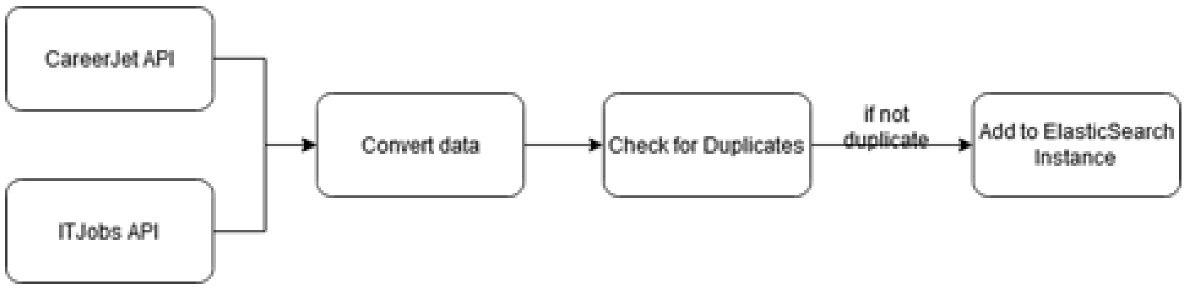


Figure 4.1: Data Information Flowchart [32].

The creation of this shared data model allows for easier data crunching, inclusion of additional sources of data, as well as duplicate detection since the model is the same. Overall, it allows for better control over data and which fields can be used later for analysis.

4.2.1 Preparatory Work

With the intention of developing this work, the first step taken was to find websites that had job offerings and see if there was an API provided so that data could be extracted and stored on our end.

The first websites considered were the following:

- LinkedIn
- CareerJet
- ITJobs
- Net-Empregos
- Sapo Empregos

From these websites, only CareerJet and ITJobs had APIs that provided listings to every job posted on their respective websites. LinkedIn also had an API that could be used, however, this would be against their terms of service, meaning the accounts could be shut down at any time, so it could not be used as a source for data extraction. The last two sources, Net-Empregos and Sapo Empregos, do not have an API that can be used, so the only solution would be using web-scrapping techniques to extract information. While this is an effective technique, if anything changes on those websites the data extraction process will fail and need to be revisited, which is not a desired outcome.

In the early stage of development and testing, only ITJobs and CareerJet are being considered, since they are the ones that offer more direct access to the information displayed on their websites. These websites offer, for example, the following information about a job posting: Posting Date, Company Information, Job Title, Job Description and Job Location.

Unfortunately, only ITJobs gives the complete Job Description which contains relevant information. CareerJet cuts the Job description after a set number of characters making it so that web scrapping had to be used in conjunction with their API as to get full descriptions and automate the whole process.

4.2.2 Data Preparation

Before diving into the different attempts at tackling the problem, the data needed to be prepared and collected. This subsection focus on the steps taken to make sure the data was ready to use.

Data Model

Since every data source gives different fields with different names, it was necessary to create a unified model under which it was possible to convert the results obtained from API calls to both ITJobs and CareerJet rose.

This Data model should be comprehensive and extensive, but also easily expansible, so that everything was in accordance to the needs of the project. This model was worked on with people from a managerial role instead of a technical one to ensure their needs were being met.

Looking at the data gathered by both ITJobs and CareerJet, the data model ended up having the following fields:

- Source - Website it came from
- Location - Location of the job
- Date Posted - Date in which the job was posted in a year, month, day format
- Job Posting - Link to where the job was posted
- Title - Job title
- Description - The job description written by the company
- Company - Name of the recruiting company
- Website - Recruiting company website
- Contact - Recruiting company contact
- Allow Remote - If the work can be done remotely
- Salary - Salary ranged the company posted with the job.
- Type - Whether it's a full time job or part time.
- Contract - If the job is temporary or not
- Number of Words - Number of words the job description had

With this model in mind, it is now needed to convert the results obtained by both ITJobs and CareerJet and standardize the data.

ITJobs Conversion

The response obtained by the ITJobs API did not need much conversion to our proposed data model.

The API managed to return almost every field data structure had, with the exception of Source and Job Posting, however, there were "incomplete" postings that did not contain every field, with salary being the most frequent omission. Besides having to fill in the fields that were not provided, the date also had to be converted.

CareerJet Conversion

CareerJet was the more troublesome API. The job description had a fix number of characters after which the whole description was truncated. As a means to solve this, web scrapping methods were used. The API returned the link to the job posting which is then used to extract the whole job description.

Besides the trucated job descriptions, the API does not return a lot of information that would be useful in our model. CareerJet is lacking the following fields:

- Contact
- Type
- Allow Remote
- Contract

4.2.3 Handling Duplicates

It was crucial that our data extraction algorithm could handle duplicates, since most of the times companies don't post only on a single website, but instead post to several and at times even repeat the job postings on the same website. The first step towards achieving duplicates detection was to make both the job title as well as the company name lower-cased. This was done so that in case the websites handled things differently, for example, some websites like to capitalize the beginning of every word in the title. After standardizing titles and company names, the date a job was posted is also looked at as a possible duplicate detection. The month is checked to account for companies that post jobs at different points in time, which could have a multitude of reasons behind, such as, they need more people to fill that role or because they haven't found anyone within a month time frame which could be significant to identify a lack of specialized employees in that area. Since ElasticSearch uses the Lucene search engine behind, a score is attributed to documents based on their similarity to the query. Having no control over how this similarity score is calculated, in this algorithm, every document returned by the query is looked at as a possible duplicate. In short, the algorithm is as follows:

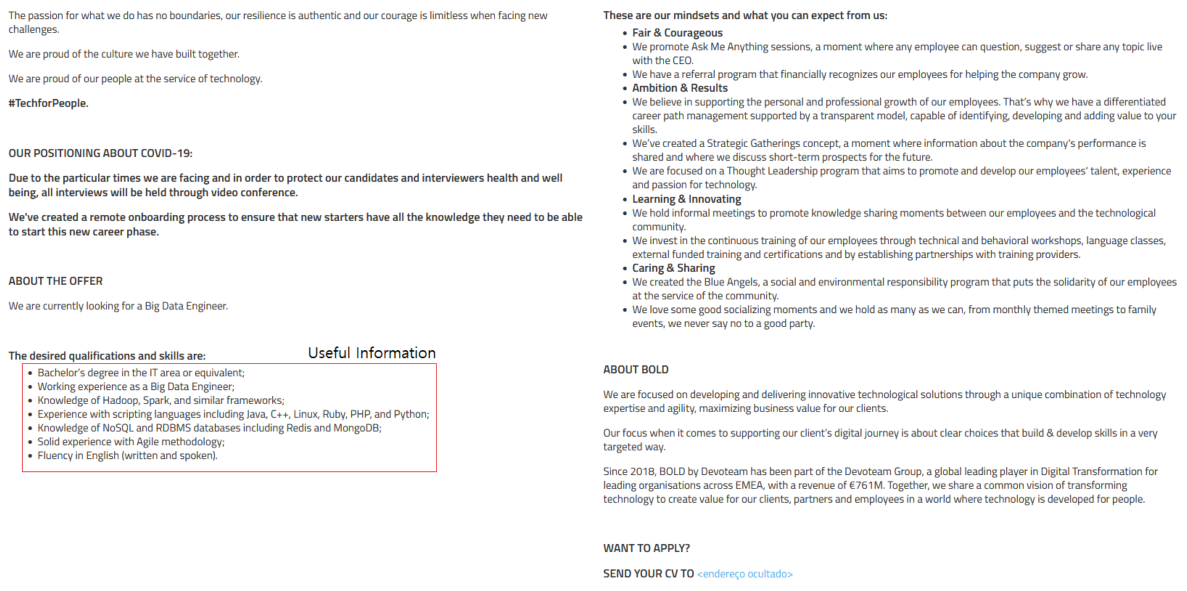
Algorithm 1: Job posting duplicate detection

```
Query ElasticSearch for documents with the same title as the new job
for Every document retrieved from the query do
    if New job posting title and company name MATCH the job in the database then
        if Month the new job was posted MATCHES the month of the job in the
          database then
            Return True
Return False
```

In case the algorithm returns a "True" value this means that there is already a job posting in ElasticSearch with the same exact title, from the same company and posted in the same month, so this document will not be added. However, in case the algorithm returns a "False" value this means that ElasticSearch could not find any document like the new one and it should be added.

4.3 Algorithm

The goal of the algorithms shown in this sections is to classify a given job offer into an ESCO occupation, by using only the job title and no other information about the job offer at hand. The reason for this is that most job descriptions are poor and mainly contain information about the merits of the company that is trying to recruit the person. Figure 4.2 shows an example of a job posting description.


The passion for what we do has no boundaries, our resilience is authentic and our courage is limitless when facing new challenges.

We are proud of the culture we have built together.

We are proud of our people at the service of technology.

#TechforPeople.

OUR POSITIONING ABOUT COVID-19:

Due to the particular times we are facing and in order to protect our candidates and interviewers health and well being, all interviews will be held through video conference.

We've created a remote onboarding process to ensure that new starters have all the knowledge they need to be able to start this new career phase.

ABOUT THE OFFER

We are currently looking for a Big Data Engineer.

The desired qualifications and skills are:

- Bachelor's degree in the IT area or equivalent;
- Working experience as a Big Data Engineer;
- Knowledge of Hadoop, Spark, and similar frameworks;
- Experience with scripting languages including Java, C++, Linux, Ruby, PHP, and Python;
- Knowledge of NoSQL and RDBMS databases including Redis and MongoDB;
- Solid experience with Agile methodology;
- Fluency in English (written and spoken).

These are our mindsets and what you can expect from us:

- **Fair & Courageous**
- We promote Ask Me Anything sessions, a moment where any employee can question, suggest or share any topic live with the CEO.
- We have a referral program that financially recognizes our employees for helping the company grow.
- **Ambition & Results**
- We believe in supporting the personal and professional growth of our employees. That's why we have a differentiated career path management supported by a transparent model, capable of identifying, developing and adding value to your skills.
- We've created a Strategic Gatherings concept, a moment where information about the company's performance is shared and where we discuss short-term prospects for the future.
- We are focused on a Thought Leadership program that aims to promote and develop our employees' talent, experience and passion for technology.
- **Learning & Innovating**
- We hold informal meetings to promote knowledge sharing moments between our employees and the technological community.
- We invest in the continuous training of our employees through technical and behavioral workshops, language classes, external funded training and certifications and by establishing partnerships with training providers.
- **Caring & Sharing**
- We created the Blue Angels, a social and environmental responsibility program that puts the solidarity of our employees at the service of the community.
- We love some good socializing moments and we hold as many as we can, from monthly themed meetings to family events, we never say no to a good party.

ABOUT BOLD

We are focused on developing and delivering innovative technological solutions through a unique combination of technology expertise and agility, maximizing business value for our clients.

Our focus when it comes to supporting our client's digital journey is about clear choices that build & develop skills in a very targeted way.

Since 2018, BOLD by Devoteam has been part of the Devoteam Group, a global leading player in Digital Transformation for leading organisations across EMEA, with a revenue of €761M. Together, we share a common vision of transforming technology to create value for our clients, partners and employees in a world where technology is developed for people.

WANT TO APPLY?

SEND YOUR CV TO <endereço ocultado>

Figure 4.2: Example of a job description.

This is a task that proved to be challenging, seeing as titles are usually short and prone to human errors, such as writing mistakes, or have extra information on them that is not relevant.

4.3.1 Main Components

While developing the algorithms, there were certain components that remained constant throughout all the different algorithms and attempts at solving the problem. The first one are the datasets that were used. These remained the same for all the solutions developed, and the second part are four different components that made up the feature vectors tested, namely:

- Elasticsearch Scoring
- Link Navigation
- Wordnet Scoring
- Occupation Similarity

Dataset

There were two datasets used when testing the different algorithms.

The first one was a dataset collected by us as shown in Section 4.1. This dataset had 100 entries that were independently classified into ESCO occupations by two different human classifiers to be used as the Gold Standard. The options given to the classifiers were the ones returned by the initial Elasticsearch query, so the search space was already narrowed down to a maximum of 15 ESCO occupations per job offer, instead of the 2500+ that comprise ESCO. This option was made since ESCO already retrieves good results and often the best

Job Offer Title	Matching ESCO Occupations		
Receptionist/ Secretary	Receptionist	Secretary	
Graphic Designer and Animator/ Web Developer	web developer	graphic designer	animator
Designer	industrial designer	set designer	graphic designer

Table 4.1: Example of ambiguous offers that match more than a single ESCO occupation.

hipothesis was already in this set, although not always ranked in the first place(s). In case where the human classifiers did not agree on one classification, a third person had to untie it and pick one. If none of the options given seemed suitable this job offer would be taken out of the dataset completely. In the end there were 79 job offers classified and ready to be used.

The second dataset was taken from kaggle [35] and it contains 19 000 job posting taken from the Armenian human resource portal CareerCenter which was the only quality job posting dataset found publicly available.. From these 19 000 a subset of 1 000 was randomly picked and classified by a single individual, using the same process as before of querying Elasticsearch to narrow the search space, according to ESCO occupations. Of these 1 000 chosen occupations, 169 had no match from our initial query leaving us with 831 examples that had a corresponding ESCO occupation match returned from the initial Elasticsearch query. This dataset was classified so that the same job offer could have different ESCO occupations associated with it. This was a change from the previous classification process due to the fact companies often want one person to have different responsibilities and this would many times translate into one job offer having several corresponding ESCO occupations associated with it making it impossible to just pick one. In Table 4.1 there are a few examples of these job offer titles that show the problem at hand well. These match several ESCO occupations to the point where a human couldn't decide between them, since the job offer title is ambiguous.

Another common problem found within both datasets was the misspelling of words. Cases such as recruiters writing "mobilar developer" instead of "mobile developer" have a negative impact on our first query to Elasticsearch, which would end up impacting the results produced. However, considering that misspelling is a common mistake, even if any were found on the datasets, they would not be corrected due to the goal of having as much real-life like data as possible so the algorithms would encode such situations.

The algorithms developed were tested with these datasets of 79 and 824 examples.

Elasticsearch Scoring

As was shown in Section 2.1.2 ESCO is a framework comprised of over 2500 occupations. This framework is provided as an ontology for free and instead of going over the traditional methods of querying an ontology, for example, using SPARQL, the approach was to upload it into Elasticsearch.

Uploading the ontology onto Elasticsearch allows for an initial filtering by querying the title of the occupation and getting back a reduced number of ESCO occupations which words were more similar to the query just made.

The Elasticsearch Scoring function uses the BM-25 algorithm to match and score documents [10]. It attributes a ranking to each document, which in this case represents an ESCO occupation, of how likely they are to match the given query. This allows us to narrow our search space from 2500+ occupations to only a few and the score attributed to each document will be a feature for our feature vector in both models.

There was also another upload of this ESCO into Elasticsearch, but this was a processed version of it that was run through TreeTagger [30] in order to lemmatize the words, so that the matching done between the job offers title and the ESCO occupation was made easier, as the queries would be the lemmas and not the whole words. This works by reducing the quantity of word forms, meaning that every word sharing the same lemma would be represented by that lemma. An example would be the different grammatical conjugations there are for the same word.

Graph Centroids

While Elasticsearch attributes a score to each returned document, these could be one of two types, ESCO skills or ESCO occupations. The goal of this component is to search the ontology through the skills and their connections to each occupation.

Each skill has occupations associated with it, these occupations are extracted and matched against the ones returned by the initial Elasticsearch query. If there are any matches, a score is attributed to this metric, otherwise it will be 0.

The scoring is done as follows:

R = number of relevant occupations that came from the skills

N = Number of times an occupation showed up on the skills

$$Score = N/R$$

N represents the number of times a specific occupation showed up throughout the skills that were in the query and R represents the total number of occupations that showed up on these same skills.

This link navigation allows us to transverse horizontally through the ontology. The goal is to try and promote singular occupations that are relevant in not only the occupations category of ESCO but also in the skills category.

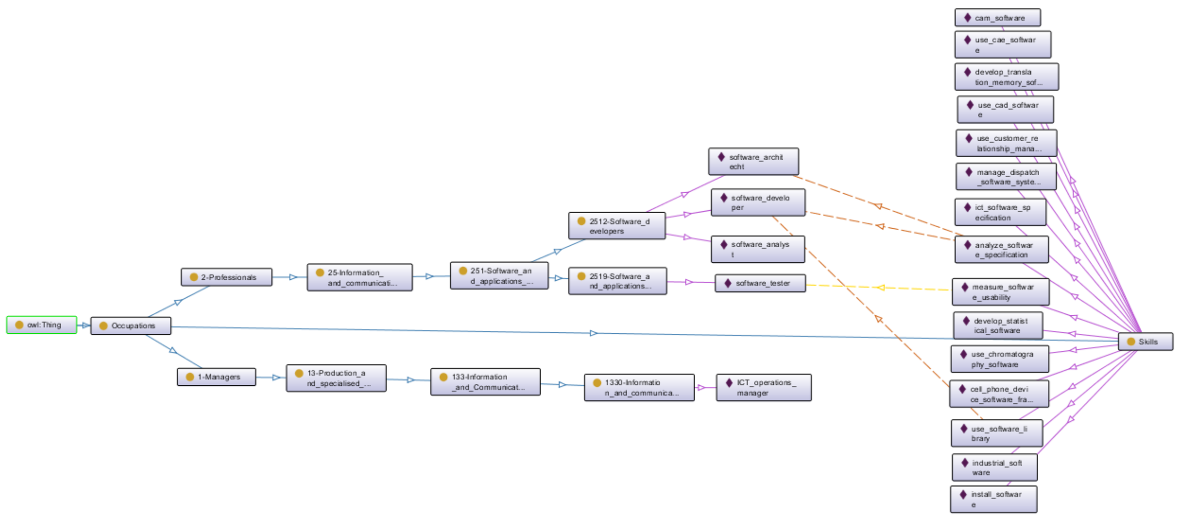

owl:Thing
Occupations
2-Professionals
25-Information_and_communicati...
251-Software_and_applications_...
2512-Software_developers
2519-Software_and_applications...
software_archit echt
software_developer
software_analyst
software_tester
1-Managers
13-Production_and_specialised_...
133-Information_and_Communicat...
1330-Information_and_communica...
ICT_operations_manager
cam_software
use_cae_software
develop_translation_memory_sof...
use_cad_software
use_customer_relationship_mana...
manage_dispatch_software_syste...
ict_software_specification
analyze_software_specification
measure_software_usability
develop_statistical_software
use_chromatography_software
cell_phone_device_software_fra...
use_software_library
industrial_software
install_software
Skills

Figure 4.3: Example of an ontology graph with the nodes returned by the query ”Software Engineers”.

For example, looking at Figure 4.3, the query for "Software Engineers" returns these results with "software tester" being the highest score on Elasticsearch query, however, this was labeled as belonging to "software developers". It is possible to see that there are nodes coming from the skills returned that connect to the occupations "software developer", "software architect" and "software tester". The dashed arrows from right to left show that the software developer occupation is mentioned in more skills, it should be rewarded since it is more relevant to the query made even if it did not come first in Elasticsearch score.

Wordnet

The third feature in our feature vector makes use of Wordnet to find the semantic similarity between the query title and the occupation title. The score increases with the semantic similarity of words.

A simplified version of how this is done can be found in Algorithm 2.

Algorithm 2: Wordnet Similarity Score

```
while There are documents do
    Extract the occupation title from the document
    Split the title into individual words
    Fetch the synsets for each word (document synsets);
    while There are document synsets do
        while There are query synsets do
            Calculate the wup similarity between the synsets
            Save the highest similarity result for each word.
Similarity Score = sum of max similarities for each word / length of the query title
```

These scores indicate how semantically similar the query title and the occupation title are. A similarity score of 1 would indicate that both titles are the same, while a score of 0 would indicate no correlation between them.

Taxonomical Similarity

The fourth and last metric that is used by the feature vectors was called Occupation Similarity.

Each ESCO occupation has a code associated with it and each digit of this code represents a level. The goal of this similarity is to attribute a score based on how similar the occupation codes are between the ones returned by the initial ElasticSearch query and the ones found on skills. The more digits they have in common the higher the score attributed is so that deeper levels of similarity are promoted. However, since this is a generic approach to navigating ontologies, due to the structure ESCO has some occupations might seem similar on a surface level but their respective codes will differ starting on the first digit.

Figure 4.4 shows the path for 5 different occupations obtained by the "Software Engineers" query.

This navigation allows us to go through the ontology vertically and find similarities between occupations even if, on the last level, these occupations are different.

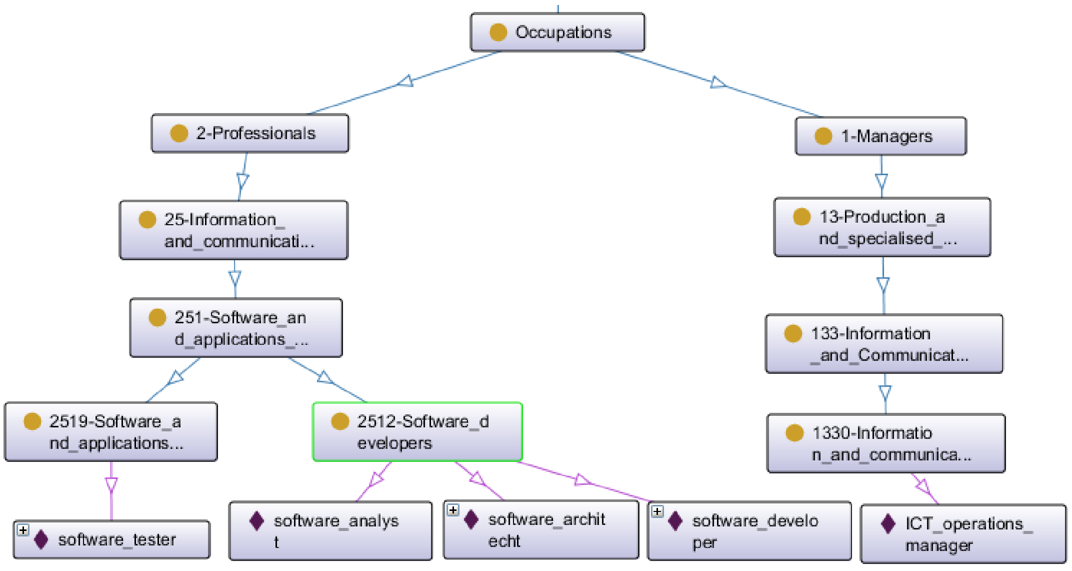

Occupations
2-Professionals
1-Managers
25-Information_ and_communicati...
13-Production_a nd_specialised_...
251-Software_an d_applications_...
133-Information _and_Communicat...
2519-Software_a nd_applications...
2512-Software_d evelopers
1330-Informatio n_and_communica...
software_tester
software_analys t
software_archit echt
software_develo per
ICT_operations_ manager

Figure 4.4: Example of the path for each occupation in the "Software Engineers" query.

4.3.2 Occupation match as a multi-class problem

The first attempt at this problem was a multi-class classification problem with 15 classes, the number of hypotehsis returned by ESCO, in which the classification label would discriminate which hypothesis is the best match.

Feature Vector

After having the dataset ready and classified, the feature vector needed to be constructed. This feature vector would make use of the features as described in Section 4.3.1.

Each feature vector is made up of 60 features and 1 target. The 60 features come in sets of four, each representing the Elasticsearch score, link navigation score, wordnet score and occupation based score for each ESCO occupation for one job offer. In case Elasticsearch does not give 15 different options for a given job offer, we would have a vector filled with zeroes until the 60 features were met.

This feature vector construction has a lot of features, however, a lot of the last features are filled with zeroes, due to Elasticsearch not always returning the same number of documents for each query (in case there aren't 15 relevant documents it will only return the appropriate number of relevant documents to a given query).

The target is a number between 1 and 15, corresponding to the correct ranking of the document returned by Elasticsearch, this is, with the documents ordered by Elasticsearch score a target of 4 would mean that the correct ESCO occupation would be the 4th document that was returned by this query.

Since the datasets used in this body of work did not have that many examples (79 and 824) having a feature vector with 60 features and 15 possible different targets might be a problem due to the lack of data. It was observed that there were even classes that had no representation and plenty vectors were sparse leading to a very imbalanced dataset. Having a machine learning algorithm work this data out could be difficult due to the nature of it.

Testing

A machine learning approach was taken when it came to trying to find an algorithm that could classify this data into the respective categories. The objective that was set for this approach was to beat the initial Elasticsearch score of 40% accuracy, which came from comparing the first Elasticsearch query result with what was defined as the Gold Standard, and see if the machine learning approach was suitable for this problem despite all the dataset limitations previously presented.

The algorithms tested were the following:

- Random Forest
- Support Vector Machine
- Neural Network
- ADA Boost
- Decision Tree

With the goal of finding the best parameters, a grid search was conducted to find the model that would best fit our data, the models were tested with a 5-fold cross-validation and the best results were the Random Forest, that achieved an accuracy of 58.6% and the Neural Network with a result of 52.7% which showed an improvement of about 10 percentual points over the baseline of Elasticsearch.

4.3.3 Occupation match as multiple binary classification problems

While the multi-class approach managed to slightly improve the results from 40% to 58.6%, there was another approach that could be tried. In this approach instead of having multiple classes at once and have the machine learning algorithms try and learn which class is the correct one, despite the obvious issues with the feature vector, there are only two classes at a time (one against the other) and the result is binary to predict which one is more likely to be the more relevant one to the job offer in question.

Feature Vector

The feature vector for this model follows the same pattern as the last one, the features that were created in Section 4.3.1 are still used, however, instead of having 60 features the base model for this approach has only 12 features, 8 describing both alternatives (2x4 features) plus 4 representing the differential of scores between both alternatives. This implied the expansion of our dataset as each multiple class vector needed to be represented by a set of binary class vectors.

Since this model puts ESCO occupations returned by the initial Elasticsearch query against one another the feature vector consists of:

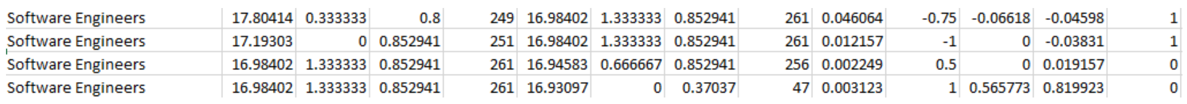
- 4 Score results (Elasticsearch scoring, link navigation, Wordnet, occupation similarity) for the first occupation

- 4 Score results (Elasticsearch scoring, link navigation, Wordnet, occupation similarity) for the second occupation

- Normalized difference between the scores of the same category divided by the max value of the two.

$$difference = \frac{Score1 - Score2}{\max(Score1, Score2)}$$

This feature vector would then have a target of 0 or 1 depending on which occupation won the "duel". In Figure 4.5 there is the example of a feature vector for a job offer that was labeled "Software Engineers".



Software Engineers	17.80414	0.333333	0.8	249	16.98402	1.333333	0.852941	261	0.046064	-0.75	-0.06618	-0.04598	1
Software Engineers	17.19303	0	0.852941	251	16.98402	1.333333	0.852941	261	0.012157	-1	0	-0.03831	1
Software Engineers	16.98402	1.333333	0.852941	261	16.94583	0.666667	0.852941	256	0.002249	0.5	0	0.019157	0
Software Engineers	16.98402	1.333333	0.852941	261	16.93097	0	0.37037	47	0.003123	1	0.565773	0.819923	0

Figure 4.5: Example of the feature vector for the job offer titled "Software Engineers".

In an attempt to generate the dataset, we take the labeled data, extract the name of the correct ESCO occupations and start putting the other occupations returned by the query against the correct ones. The way the dataset generation is setup makes it so that if there are occupations with a higher Elasticsearch score than the labeled one, the scores corresponding to these will show up in the first four slots of the feature vector while the "correct" one will be the second four slots with a target of 1. When the correct one has a higher Elasticsearch score than the remaining query results it will become the first four results and the target will promptly be 0 since it is supposed to "win the duels".

Seeing as Elasticsearch does not return a fixed number of occupations per query, only a maximum number, this approach already seems more robust. There are no sparse matrices, the target is binary (there is no need to have an example for each of the classes like in the one tested on Section 4.3.2) and the dataset could be extended from the 824 examples for the previous attempt to over 6000.

At later stages and so as to try and improve the results several other feature vectors were tried with small changes into this baseline. For example, trying to incorporate more information about the occupation levels into this vector. However, in the end, the baseline was the model that performed better.

Machine Learning

Just as in the previous attempt, the first action was to use machine learning on the new dataset. In order to train the model we split the data into 80% training data and 20% test and made sure every model used the same training data. The models were also trained using a 5-fold cross-validation on the 80% training data with the remaining 20% being used purely for testing.

The different combinations of parameters tested can be found on Table 4.2.

As a means to better differentiate between the "winners" and "losers" of the duels, there was also a rule applied to the dataset. The rule was in case the winner was the second occupation at least one of the score differences had to be negative, meaning that the second

Algorithm	**Parameters**	
Random Forest	Number Estimators	50, 100, 200, 500
	Minimum Samples Split	2, 3, 4, 5
	Criterion	Gini, Entropy
	Max Features	Auto, Sqrt, Log2
	Max Depth	Auto, 5, 10 ,20
SVM	C	0.06, 0.125, 0.25, 0.5, 1
	Gamma	scale, auto
	Kernel	linear, poly, rbf, sigmoid
Neural Network	Hidden Layer Size	5, . . . , 25
	Max Iterations	1000, 1200, 1400, 1600, 1800, 2000
	Solver	lbfgs, sgd, adam
ADA Boost	Algorithm	SAMME, SAMME.R
	Learning Rate	0.5, 1, 1.5
	Number Estimators	50, 100, 150, 200
Decision Tree	Criterion	Gini, Entropy
	Max Depth	Auto, 5, 10 ,20
	Max Features	Auto, Sqrt, Log2
	Minimum Samples Split	2, 3, 4, 5
	Splitter	best, random
KNN	Neighbors	2,...,30
	Weights	Uniform, Distance
	algorithm	Auto, Ball tree, Kd tree, Brute force

Table 4.2: Parameters tested for each algorithm.

option needed to have a higher score in at least one of the features therefore justifying it being a best fit.

Using only these machine learning algorithms and finding the best parameters was not enough to get a significant increase in results when compared to previous models. It was observed that the algorithms had to give a result of 0 or 1 no matter how close the decision had to be, for example, one occupation being selected with 99% confidence would have the same meaning of being selected with 51% confidence, so the next step was to look at the probabilities of each ESCO occupation winning against each other rather than dealing in absolutes (it either wins or loses). In Figure 4.6 we can observe the probabilities of each class winning against one another. It is possible to observe that some occupations are really close to one another, for example, occupation 2 and 4 while others are clear winners.

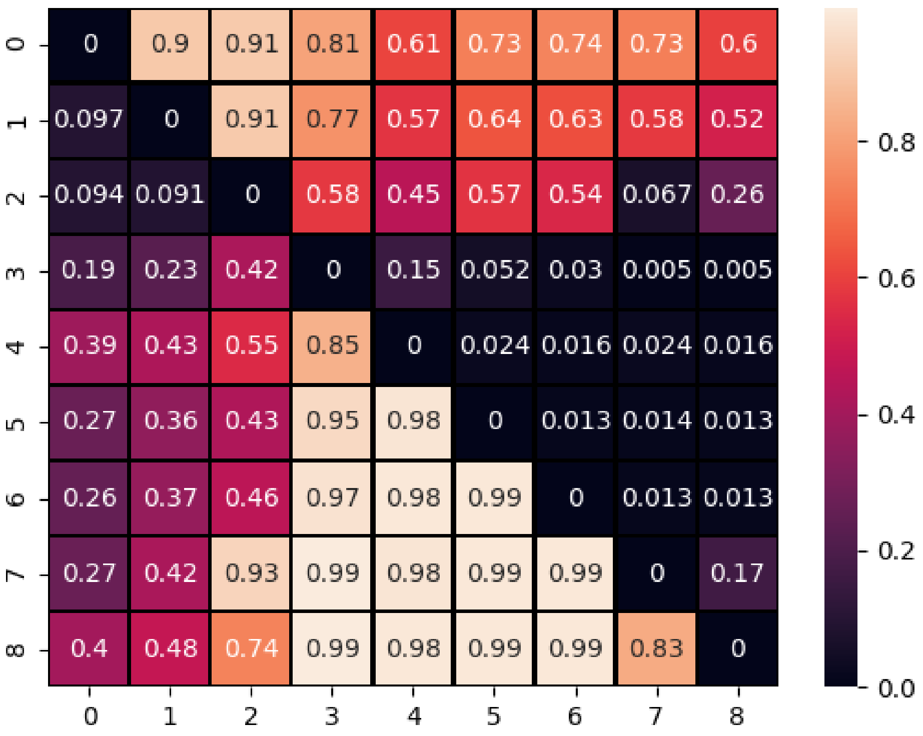


Figure 4.6: Probability of each class defeating one another according to a Random Forest model. This example was for the ”” hilti ” subdivision salesman” query.

With this type of model, however, there is not a single index that will give us the result since it will the only result obtainable is whether or not an occupation ”won” against another. The way these algorithms were tested will be explained further in Section 4.4.

4.4 Testing

While the initial attempt at testing the machine learning algorithm was easy and straightforward, with 15 classes each representing the index of a different occupation, testing the machine learning algorithm in the 1v1 model is different.

As it was mentioned before, the data was split into 80/20, training and testing respectively. On top of the training split the models were also trained using a 5-fold cross-validation in order to try and not overfit the models and get as good of a generalized model as possible.

4.4.1 Binary Classification

The machine learning model has two ways of being tested. The first one is the how good it is at evaluating the duels, how consistently does it pick the right winner. This is a straightforward way since there is only the need to look at targets and the output of the algorithm. The second way is to test how good it is at predicting the correct occupation.

For this second option, and before implementing an algorithm that interprets these probabilities as transition probabilities in Markov Models thus allowing to compute steady state probabilities, the way to test the accuracy of the model was to put up the occupations against one another and calculate the ratio of how many "fights" they won. The occupation with the better ratio would be the picked one. In case there was a tie the occupation with the highest BM-25 score would be the picked one.

4.4.2 Binary classification results as transition probabilities

If we consider each occupation alternative as a state in a Markov Model, then the probability of an occupation against the other would represent the steady state probabilities between those states. So, based on these probabilities, a Markov Transition matrix is built using those probabilities as transition between the two states, and the states' self-transitions were estimated by two distinct ways:

- As the intersection of the complementary transition probabilities.
- As the complementary of the sum of the transition probabilities.

Figure 4.7 helps understanding how this Markov Transition matrix is built. When conducting a binary classification between states A and B, the machine learning algorithm estimates the probability of being the state A or the state B. As the evaluation universe is A plus B, then the values obtained are estimates of $\frac{P_A}{(P_A+P_B)}$ and $\frac{P_B}{(P_A+P_B)}$ respectively. As such, a transition matrix can be built as depicted in Figure 4.7. Then is necessary to estimate the self state transitions and to normalize the rows for its values add up to 1.

$$\begin{array}{cccc}
 & A & B & C \\
A & P_{AA} & \frac{P_B}{P_A+P_B} & \frac{P_C}{P_A+P_C} \\
B & \frac{P_A}{P_A+P_B} & P_{BB} & \frac{P_C}{P_B+P_C} \\
C & \frac{P_A}{P_A+P_C} & \frac{P_B}{P_B+P_C} & P_{CC}
\end{array}$$

Figure 4.7: Matrix example to help explain how the probability matrix was calculated.

With this in mind, the data could not just be split into 80/20 as it was for the multi-class model. Because now the same job offer had more than one feature vector associated with it, the split had to be done by job offer title instead of randomly picking vectors out of the dataset. The risk associated with randomly picking vectors of the dataset and training with a regular 80/20 and cross-validation is that, while training the model won't see every duel for a single job offer title, it will see part of them making it so that the testing results can not be trusted due to the algorithm "already knowing" some results for some duels. So the dataset is instead splint into 5, an 80/20 split still, by job offer titles, removing all the respective feature vectors from the training dataset.

In order to test the model a query is made to Elasticsearch where all options for a given job offer are presented. The feature vector is built using the features presented in Section 4.3.1.

These feature vectors go through a machine learning algorithm, in the case of our testing it goes through 5 different algorithms, where instead of getting a single result for the duel the probabilities that one wins over the other are extracted. These probabilities will then build the initial probability matrix (As an example, see Figure 4.6). After going through the Markov chain process the result will be a probability vector with a probability for each occupation returned by the initial query. The vector is ordered the same way the occupations were returned by the elasticquery, that is, if the highest probability occupation is the 3rd index in our vector this directly relates to the 3rd occupation returned by the initial query.

For example, taking into consideration the example for the initial query into Elasticsearch for "Software Engineers" would return the occupations "software tester", "software analyst", "software developer", "software architect" and "ICT operations manager" ordered by their BM-25 score. This would mean that "software tester" would be the index 0. The final probability vector is:

$$[0.2043, 0.1943, 0.3491, 0.1559, 0.0964]$$

With the highest probability being the 3rd index this would mean that the occupation picked by the Markov model is "software developer".

Knowing this, it is now possible to extract all the details needed from the ESCO occupation and compare it to our targets and evaluate whether it hit or missed. Figure 4.8 has a flowchart of how the testing works for this model.

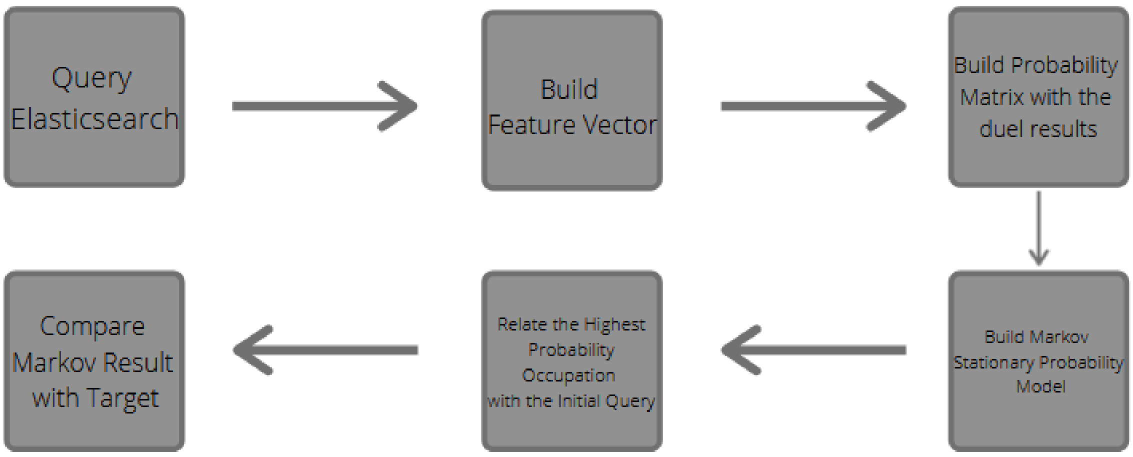

Query Elasticsearch
Build Feature Vector
Build Probability Matrix with the duel results
Build Markov Stationary Probability Model
Relate the Highest Probability Occupation with the Initial Query
Compare Markov Result with Target

Figure 4.8: Markov Training flowchart.

Chapter 5

Results

This chapter will serve to discuss the results obtained from both the data collection, and the visualizations that were made with it, as well as the results from the algorithms to classify the job offers into ESCO Occupations.

5.1 Results

5.1.1 Data Collection and visualization

Ever since the system was developed it has been collecting data on job postings for the Aveiro region of Portugal. As of the writing of this document, the system has collected over 12 000 job offers that have all been categorized as ESCO occupations by the algorithms explained here. Besides being classified as ESCO Occupations, the necessary and optional skills for each one of them has also been extracted.

What this data classification and extraction permits us to do is create fully dynamic dashboards about specific regions of the world, for example, to compare how regions are doing and get an idea of the needed occupations or skills at each moment, or to try to forecast jobs that are on the rise and allow education institutions to adapt their curriculum based on the skills/knowledge identified in these dashboards. From a recruitment perspective it allows recruiters to better name their job postings and identify the necessary skills and competences to look for in the job market. These dashboards are updated in real time as soon as data gets added on Elasticsearch making it a very powerful analysis tool.

In Figures 5.1and 5.2 there are two examples of such dashboards. Despite the algorithm used classifying occupations up to ESCO fifth level, the standard to display, according to the state of the art [23], is the third level, hence these dashboards only show the first three levels. As said before, these dashboards are fully dynamic and it is possible to dig deeper, by simply selecting the desired occupation/skill to look deeper into it will automatically update the other graphs too, Figure 5.3 showcases an example of this by selecting "Technicians and associate professionals" on the "1 Digit Occupations" graph. The goal of these graphs were to showcase that dashboards like the ones made by the European Centre for the Development of Vocational Training (CEDEFOP) [13] can easily be made for the data gathered by the system made.

Besides this, the creation of graphs that use any of the fields defined on the data model is possible, since these are all indexed when new data is added onto Elasticsearch, making these dashboards a very powerful managerial tool.

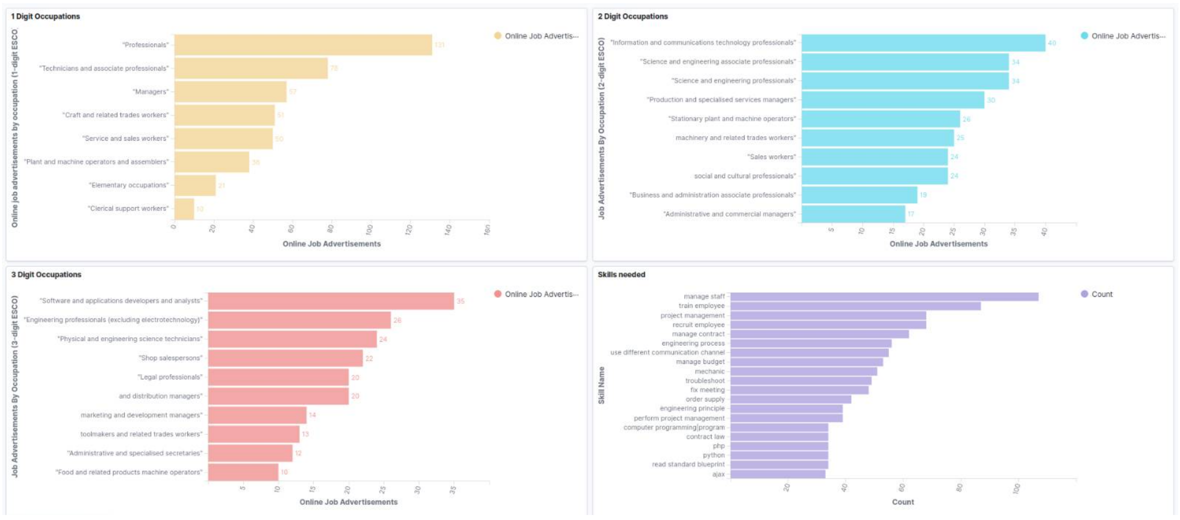


Figure 5.1: Dashboard with occupations (until the 3rd level) and skills (4th level) for the Aveiro region for the past 90 days.

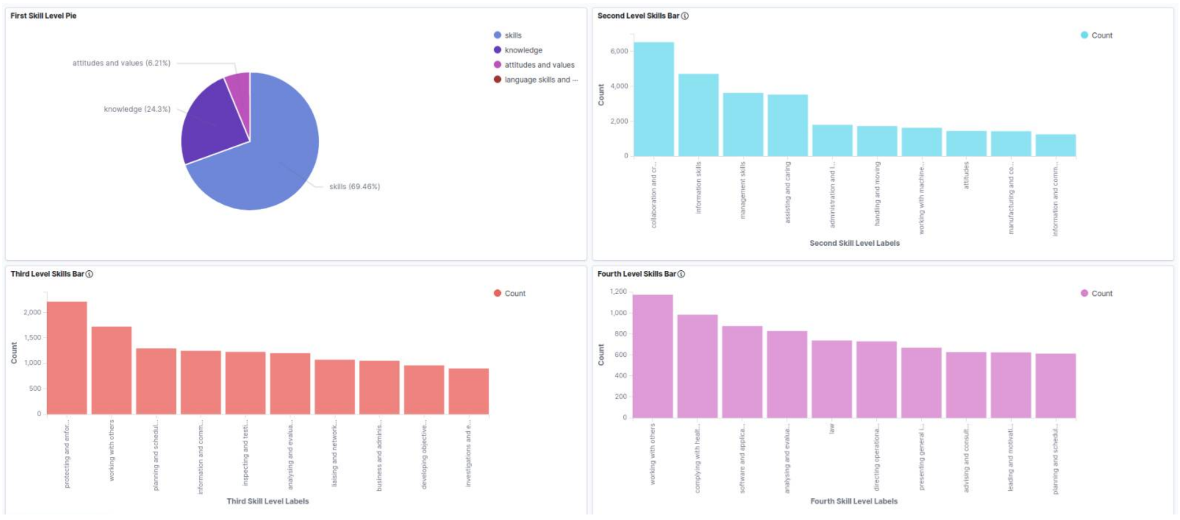


Figure 5.2: Dashboard with skills (until the 4th level) for the Aveiro region for the past 90 days.

5.1.2 Machine Learning

This section will focus on the machine learning results for the best model, that is the one versus one model previously described in Section 4.3.3.

As was mentioned previously, on Table 4.2 there are several algorithms and combinations of parameters tested for each algorithm. The first few rounds of testing used all of the algorithms shown, however, with time a pattern started forming, where both the SVM and the ADA Boost algorithms did not perform well compared to the rest so, they were dropped in the final stages of testing. Dropping these left us with the following algorithms:

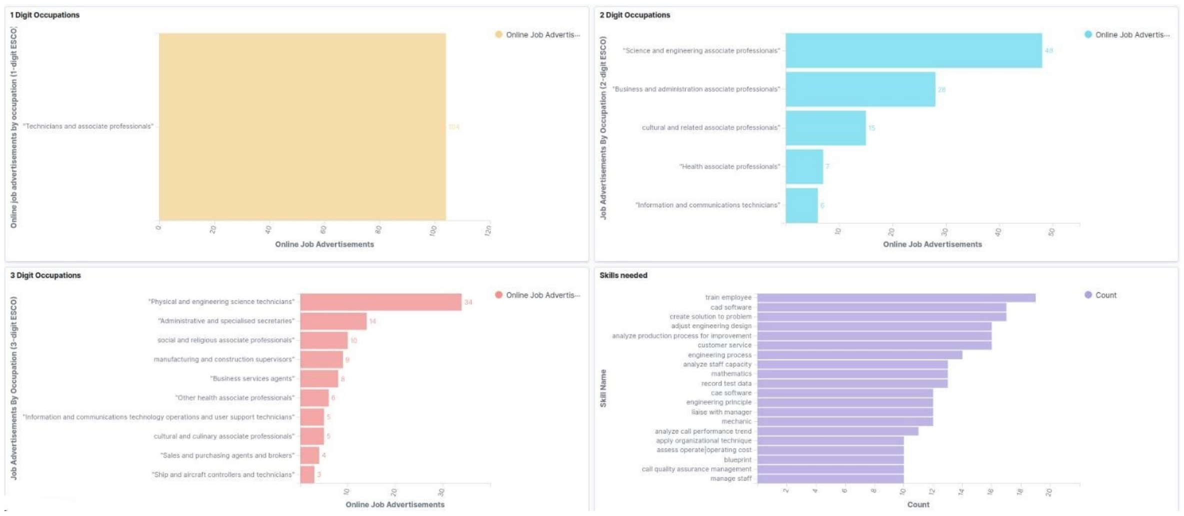


Figure 5.3: Example of dashboard updating by clicking one specific occupation.

Algorithm	Max Score	Min Score	Avg Score	Std Deviation
Random Forest	**0.829**	0.801	0.819	0.008
Neural Network	0.794	0.699	0.774	0.017
Decision Tree	0.791	0.737	0.765	0.011
KNN	0.779	0.728	0.763	0.011

Table 5.1: Grid search results for the different machine learning algorithms.

- Random Forest
- Neural Network
- Decision Tree
- KNN

As a reminder from Section 4.4 the models were training using 80% of the data with a 5-fold cross-validation and all the models were trained using the same data in order to have consistency. The results here are for the binary classification portion of the algorithm and not for the occupation identification classification since that is not the point of these machine learning algorithms. For this portion of the work the dataset was also balanced to ensure there was no bias occurring.

In Table 5.1 the best results for the training phase are shown.

Looking at the results obtained, every model is relatively stable, not having big differences between the best and the worst results. The Random Forest came in first with almost a 4% lead over the next best algorithm, the Neural Network. The Random Forest winning was also better in the next part of the algorithm. In Table 5.2 there are best parameters found for each one of these algorithms.

The algorithms were tested using the remaining 20% of the data that was not eeven seen during the training phase. Tables 5.3 5.4 5.5 5.6 have the confusion matrices results using

Algorithms	**Parameters**	
Random Forest	Criterion	Gini
	Max Depth	20
	Max Features	Log2
	Min Samples Split	5
	Number of Estimators	200
Neural Network	Hidden Layer Size	29
	Max Iterations	1400
	Solver	adam
Decision Tree	Criterion	Entropy
	Max Depth	5
	Max Features	Square Root
	Min Samples Split	4
	Splitter	Best
KNN	Algorithm	auto
	Number of Neighbors	21
	Weights	Distance

Table 5.2: Best parameters found for each algorithm.

Random Forest		
	0	1
0	1455	97
1	183	285

Table 5.3: Confusion Matrix for the best scoring random forest model for the first fold.

the testing set for the best parameters of each algorithm. These values were obtained for the first fold during the algorithm training.

The initial training tests show that this method could be a potential good approach to the problem so the results were further explored. Testing the machine learning prediction results using the best performing algorithm (Random Forest) yielded the results shown on Table 5.7 and the corresponding confusion matrix for each 1st ESCO level can be seen on Table 5.8.

From this confusion matrix it is possible to calculate the precision, recall and F-Score. For the 1st ESCO level the model attained a precision, recall and F-Score of 0.7649. In Table 5.9 the F-Score for each class is shown, however, as seen on the confusion matrix the latter examples are not as represented as the first classes so any error or hit on these could drastically change these numbers.

While these results may seem promising, they were comparable to the ones previously

Neural Network		
	0	1
0	1441	111
1	257	211

Table 5.4: Confusion Matrix for the best scoring neural network model for the first fold.

Decision Tree		
	0	1
0	1511	41
1	281	187

Table 5.5: Confusion Matrix for the best scoring decision tree model for the first fold.

KNN		
	0	1
0	1423	129
1	280	188

Table 5.6: Confusion Matrix for the best scoring KNN model for the first fold.

	Level 1	**Level 2**	**Level 3**	**Level 4**	**Level 5**
Random Forest	0.764855	0.668774	0.620733	0.572693	0.551201

Table 5.7: Accuracy for each ESCO level.

		Prediction								
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Chosen (Truth)	**1**	91	26	8	0	2	1	1	0	1
	2	24	347	39	3	7	1	11	0	0
	3	14	16	104	5	2	0	0	1	0
	4	2	4	2	34	2	0	0	0	0
	5	1	0	3	0	13	1	0	0	0
	6	0	0	0	0	0	2	0	0	0
	7	0	4	0	0	0	0	8	0	0
	8	1	1	1	0	0	0	0	3	1
	9	0	0	0	0	1	0	0	0	3

Table 5.8: Random Forest confusion matrix for every ESCO first digit Occupation

First-Digit	**F-Score**
1	0.6920
2	0.8361
3	0.6957
4	0.7907
5	0.5778
6	0.5714
7	0.5000
8	0.5455
9	0.6667
Global	0.7649

Table 5.9: Random Forest F-Score for each individual ESCO first-digit occupation.

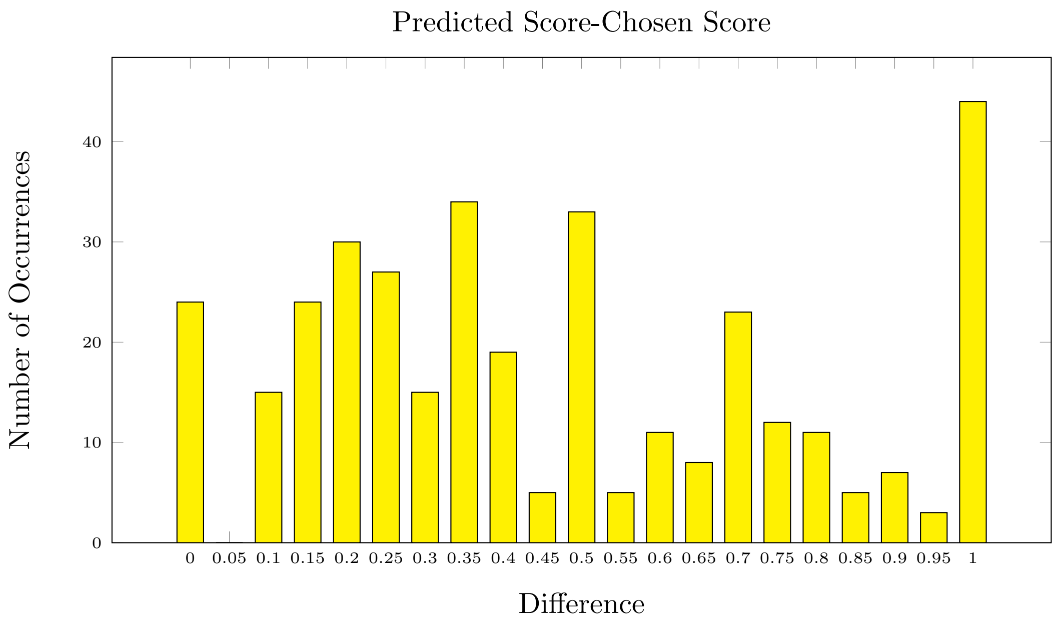


Figure 5.4: Difference between the correct occupation score and the predicted score using the Random Forest algorithm.

attained by the multi-class problem. Looking at the difference in scores between the predicted occupation and the correct occupation gives the results on Figure 5.4 which does not bode well. In 44 cases, which is the highest count, the correct option did not even stand a chance to be the correctly picked one.

The next step taken to improve the algorithm was to build a state transition matrix on top of these results. Instead of looking at the algorithm decision as binary, the goal is to fetch the percentage of one winning over the other and build a state transition matrix. While doing this there was also an interesting observation where the random forest model had some close decisions on certain "duels" while the other algorithms were much more certain of their decision, which ends up working out in favour of what is trying to be accomplished with the state transition matrix and probabilities of each occupation being the correct one for a given job offer title.

5.1.3 Occupation Transition Matrix

Building the transition matrix after machine learning is where matching job offer titles with ESCO occupations comes to fruition. After having a model that can decide between two occupations based on the features described on Section 4.3.3 the probability matrix had to be built.

Referencing back the matrix used in the previous Chapter, Figure 5.6 will be used to help explain how the probability matrix was built.

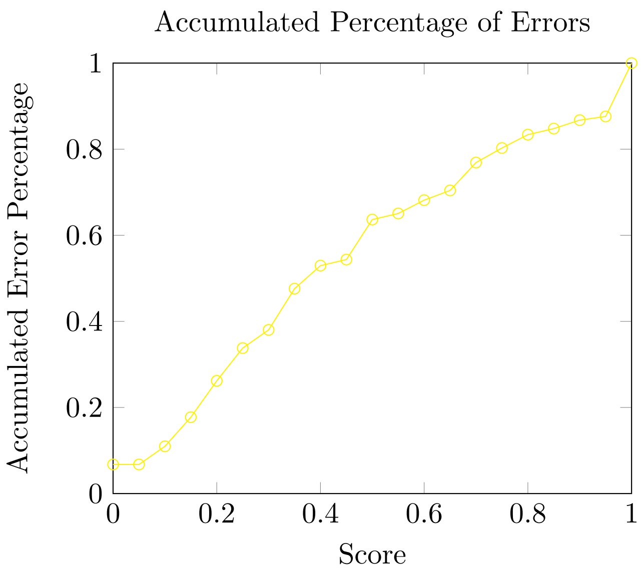


Figure 5.5: Accumulated errors for the Random Forest Model.

$$\begin{array}{cccc} & A & B & C \\ A & P_{AA} & \frac{P_B}{P_A+P_B} & \frac{P_C}{P_A+P_C} \\ B & \frac{P_A}{P_A+P_B} & P_{BB} & \frac{P_C}{P_B+P_C} \\ C & \frac{P_A}{P_A+P_C} & \frac{P_B}{P_B+P_C} & P_{CC} \end{array}$$

Figure 5.6: Matrix example to help explain how the probability matrix was calculated.

The probability matrix was built using comparisons between two alternatives, where $\frac{P_{X_i}}{P_{X_i}+P_{X_j}}$ is obtained directly from the comparison between alternatives X_i and X_j, and $P_{X_iX_i}$ is estimated by the joint probability of the complimentary events:

$$P_{X_iX_i} = \prod_{j;j\neq i} \left(1 - \frac{P_{X_j}}{P_{X_i} + P_{X_j}}\right)$$

After building the probability matrix using this methodology, the end result is shown in Figure 4.6, the end result will be a probability vector of each occupation being the "correct" one. Figure 4.8 shows what the testing process looks like for a single occupation. This process is done for each of the four algorithms presented above so that finding the best fit is possible.

Looking at examples for each of the algorithms probability matrices, there are some conclusions to be taken already. Figures 5.7, 5.8, 5.9 and 5.10 have examples for the query "Art Editor".

In these Figures, the Random Forest and Neural Network algorithms have much closer percentages for the "duels" than the KNN or the Decision Tree algorithms, meaning that they are more certain that the decision they made is the correct one, while in the neural network and random forest models there is some doubt in their decisions which means that the latter two models would, in theory, be a better fit for generating the Markov model.

The testing was done by querying 800 (160 at a time for different folds) different job offer titles and trying to match them to a given ESCO occupation. The results were split into

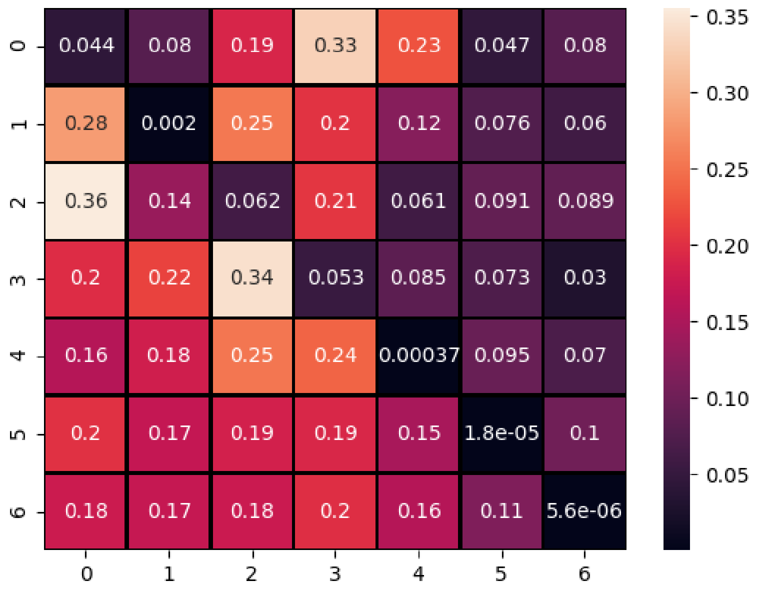


Figure 5.7: "Art Editor" probability matrix for the Random Forest model.

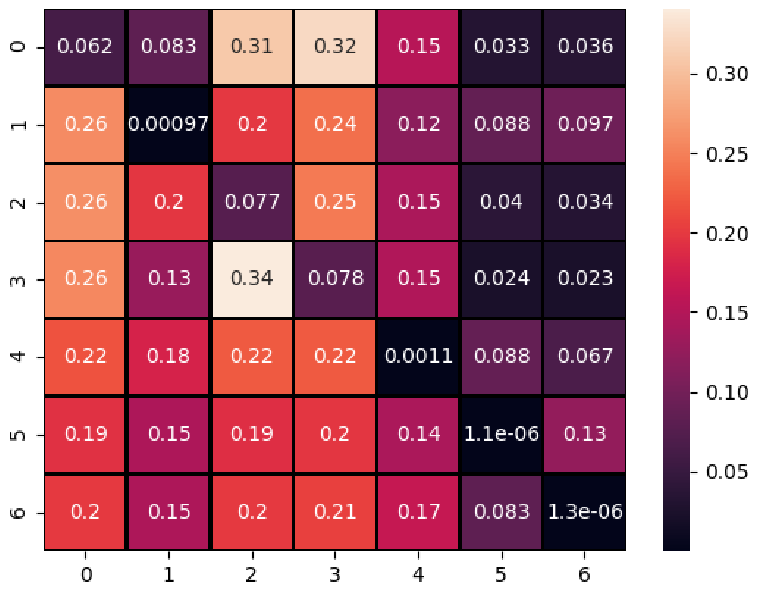


Figure 5.8: "Art Editor" probability matrix for the Neural Network model.

the different ESCO levels. The goal is to predict the level 5 correctly, however, looking at

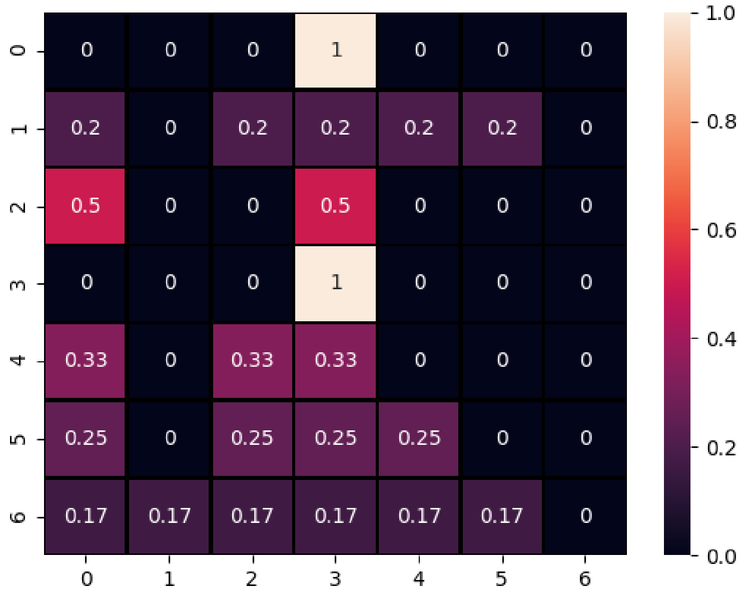


Figure 5.9: "Art Editor" probability matrix for the Decision Tree model.

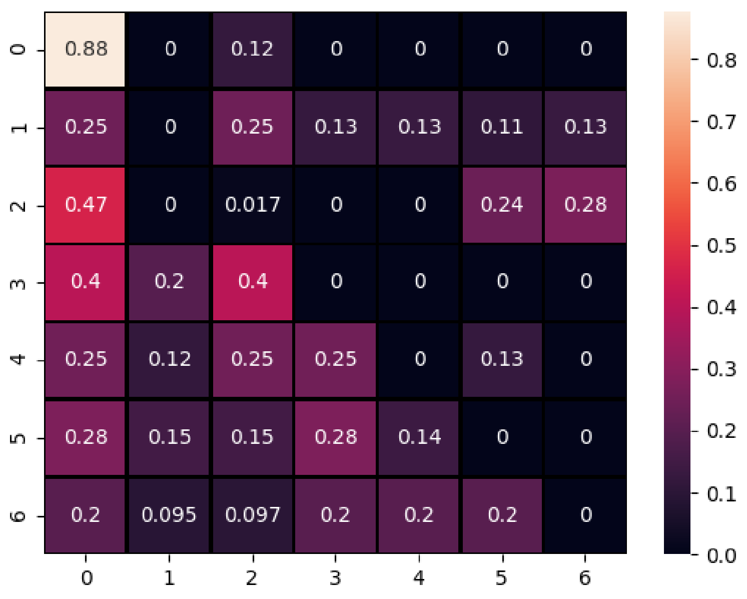


Figure 5.10: "Art Editor" probability matrix for the KNN model.

---

what the authors of [23], which use only the first ESCO level on their published results, it is

Algorithm	**Level 1**	**Level 2**	**Level 3**	**Level 4**	**Level 5**
Random Forest	0.8850	0.8100	0.7725	0.7375	0.7150
Neural Network	0.7675	0.6738	0.6338	0.5775	0.5413
Decision Tree	0.7825	0.6938	0.6413	0.6088	0.5813
KNN	0.7825	0.6725	0.6213	0.5713	0.5463

Table 5.10: Accuracy for each ESCO level using the different algorithms.

important to get these numbers out and compare them to the current state of the art. Besides this, other questions arose: In cases where the algorithm fails the classification how far from first was the correct answer ?

Would it be in the top 3 choices the algorithm would make?

This analysis allows the creation of tools for continuous improvement of the algorithms and to have defined cutoff points for when the algorithm is absolutely certain it got the occupation correctly versus when it needs human help deciding. In these cases it is possible, for example, to display the best three decisions the algorithm made.

The start of this analysis will focus on the four algorithms and finding the best one of the four, then doing a deeper analysis of the results for the best performing algorithm. This deeper analysis will focus on the aspects mentioned before. Table 5.10 shows the accuracy for each ESCO level using the different algorithms.

Despite the neural network having closer percentages when it came to the duels, adapting them to the transition matrix it did not work out well since it was the worst performing one. Much like in the previous section, the random forest also came out on top by a close to a 6% margin over the next best algorithm, the Decision Tree. With that being said, the model that will be explored more is the Random Forest, since it achieved the best results.

The first thing to take a look at is, when the algorithm failed, how badly did it fail? Since the model returns percentages, failing when the algorithm is 100% sure it picked the correct option is a lot worse than the model failing when the algorithm was only 25% sure it picked the correct option. In Figure 5.11 there is a stacked chart showing the percentage with which the algorithm made the choice, both the correct and the incorrect decisions can be seen.

Looking at the stacked chart, we can observe that when the algorithm picks a wrong answer these come out as being a low-probability choice which is the best case scenario. In Figure 5.12 the accumulated percentage of errors is seen. Around 80% of errors occur when the algorithm picks a response with a score of 0.35 while a score of 0.5 or lower represents about 90% of the errors the algorithm makes. Knowing this is useful to know when to display more than one ESCO occupation. Of the 160 decisions the algorithm made with a score above 0.7 it got 149 correct and 11 wrong for an accuracy of 93.125%. The higher the score the more likely the algorithm is to be correct by a good margin.

It is also important to figure out when the algorithm makes mistakes, how well did the correct option do? Was its score too far off from the one that was picked? Figure 5.13 has a bar chart showing the difference between the predicted score and the chosen score and Figure 5.14 has the accumulated percentage for this difference.

Most of the time when an error occurs the difference in scores between the picked option and the correct option is really small, 80% of the time it is lower than 0.35 meaning that, probably, the correct answer is among the top 3 best rated choices the algorithm could have picked. Picking a cutoff where the system decides to show more than one answer ends up

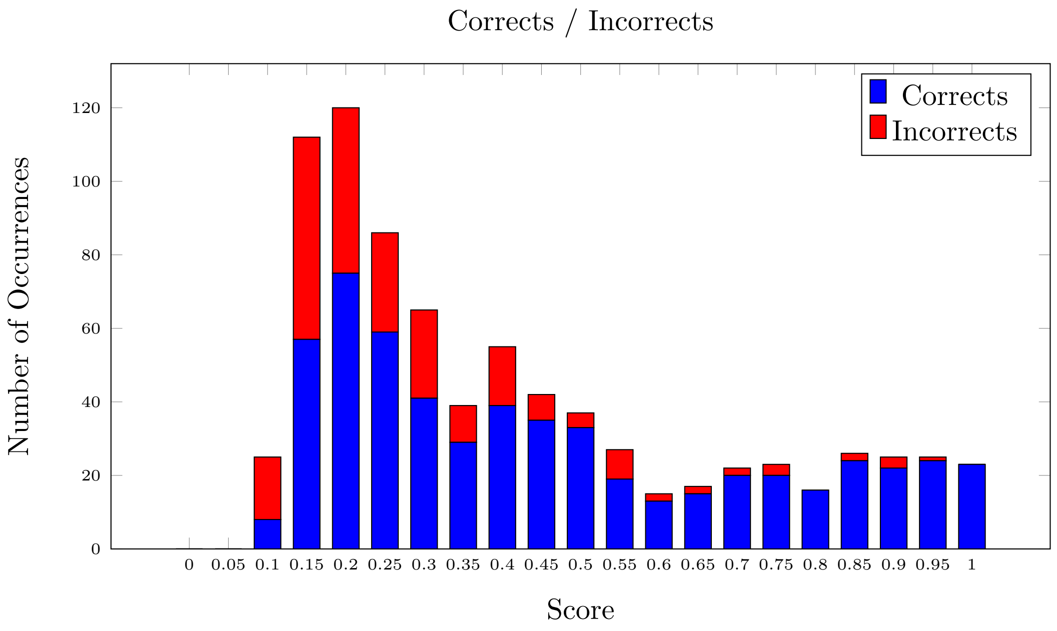


Figure 5.11: Markov decisions per percentage.

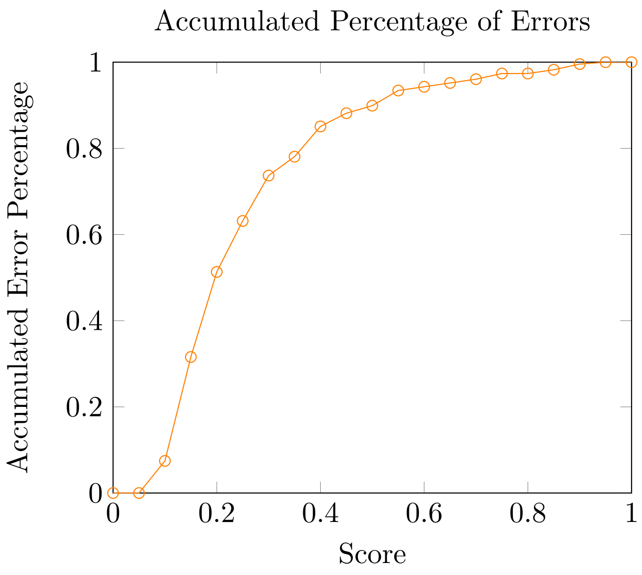


Figure 5.12: Accumulated percentage of errors when computing the steady state probability based on the transition matrix.

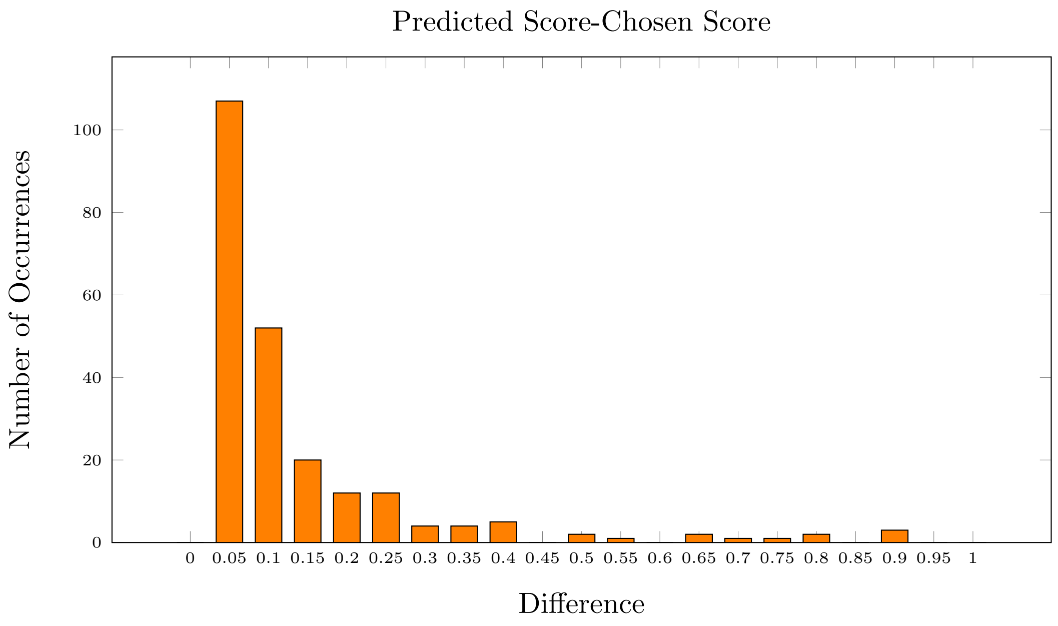


Figure 5.13: Difference between the correct occupation score and the predicted score using the transition matrix.

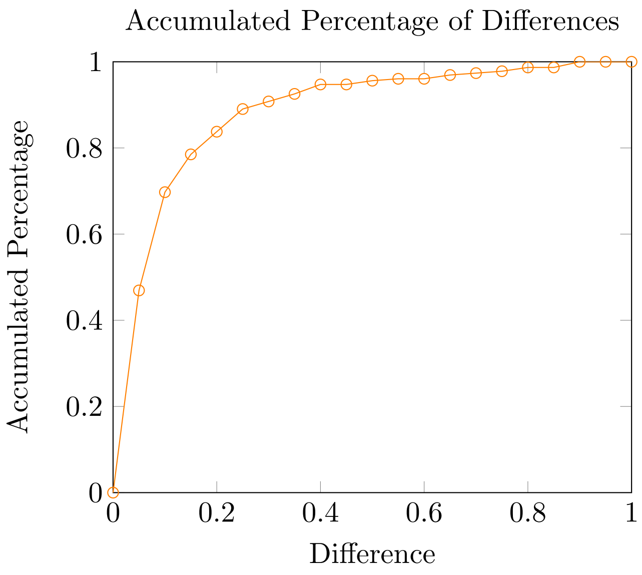


Figure 5.14: Accumulated percentage of differences for the transition matrix.

	Top2	Top3	Top4	Top5
Random Forest	0.8225	0.87	0.9025	0.93

Table 5.11: Accuracy for the decision being on the Top 2, 3, 4 or 5 decisions made by the algorithm.

		Prediction								
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Chosen (Truth)	**1**	**103**	19	7	0	3	1	1	0	0
	2	15	**382**	28	5	3	1	4	0	0
	3	9	11	**113**	1	4	0	0	1	0
	4	1	1	3	**41**	2	0	0	0	0
	5	1	0	1	0	**13**	1	0	0	0
	6	0	0	0	0	0	**1**	0	0	0
	7	0	2	0	0	0	0	**10**	0	0
	8	0	1	0	0	0	0	1	**4**	1
	9	1	0	0	0	0	0	0	0	**4**

Table 5.12: Confusion matrix for every ESCO first-digit Occupation obtained when using the state transition matrix.

being very viable if that is the case. Table 5.11 has the accuracy for the correct decision being in the top 2, 3, 4 and 5 decisions being made by the algorithm.

In this Table we see that the accuracy jumps from 71.5% when considering only the first decision made by the algorithm to 82.3% when considering the Top 2 highest percentage job offers and even reaching a maximum of 93% accuracy when dealing with the Top 5. When it comes to absolute numbers this means that for top 2 there were 142 errors in 800 distinct examples that were tested.

Comparing these results to the state of the art would mean going back to the level 1 results since that is what the authors of [23] released for their models. There are no concrete numbers to compare ourselves too, however, there is a box plot with the F-Score for the first-digit level. The latter classes do not contain many examples due to dataset limitations making it so that their precision and recall values can fluctuate very heavily with just the addition of 2 or 3 examples. With a bigger dataset this problem could have been solved.

In Table 5.12 it is shown the confusion matrix relating to the first-digit occupations from ESCO. This confusion matrix shows the clear improvements over the baseline, which would be the results labeled as ”1”. The F-Score for each individual class can be consulted on Table 5.13 as well as a global F-Score for the model. It is worth noting that the models precision and recall were equal and attained values of 0.8388.

The results obtained by ETF (European Training Foundation) are shown in Figure 5.15. They used an SVM with a dataset of 60 000 labeled examples compared to our Random Forest / Markov mixed model with just 800 labeled examples and the results obtained by this work almost reached state of the art levels despite the clear difference in number of labeled examples to train the models between the two. The algorithm here developed also has the advantage of being easily adaptable to new languages. There are also problems with the latter classes not having as much representation on our dataset which could affect the results had these had more examples to look at.

First-Digit	F-Score
1	0.7803
2	0.8946
3	0.7766
4	0.8632
5	0.6341
6	0.4000
7	0.7143
8	0.6667
9	0.8000
Global	0.8388

Table 5.13: F-Scores for individual single-digit ESCO occupations obtained using the state transition matrix

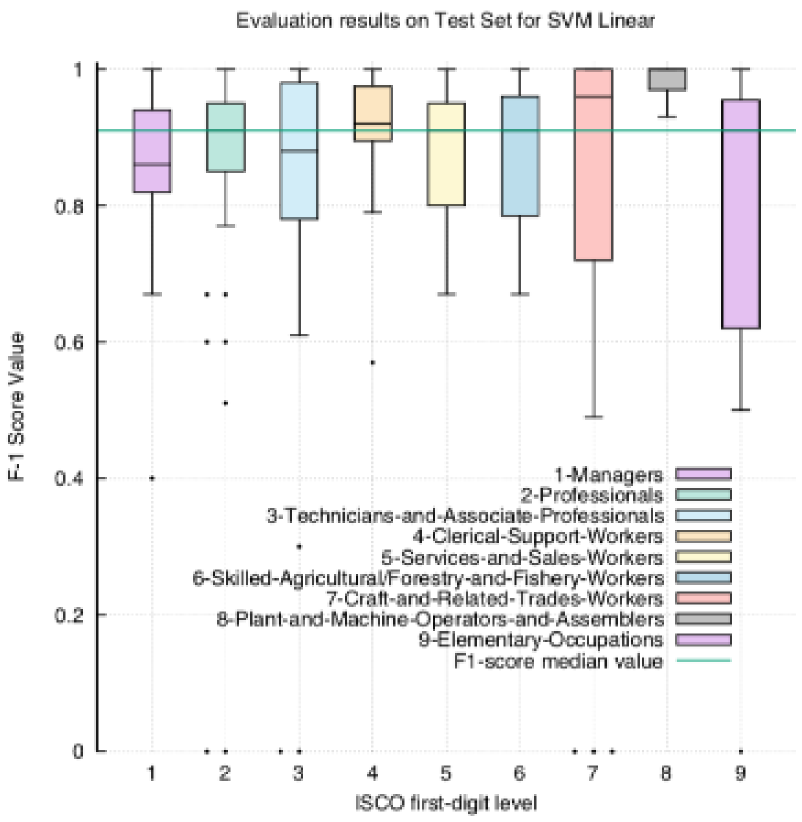


Figure 5.15: Box plot with the state of the art EFT results [23].

It is not possible to compare numbers for each class directly since these were not released in the report. The only results that were officially released are displayed in the box plot and these are only for the first-digit. Training a model to identify first-digit occupations only instead of the existing 2600+ that ESCO has to offer could yield better results.

	Level 1	**Level 2**	**Level 3**	**Level 4**	**Level 5**
Random Forest Accuracy	0.81125	0.73125	0.69000	0.66000	0.63750

Table 5.14: Results for the 2nd version of transition matrix.

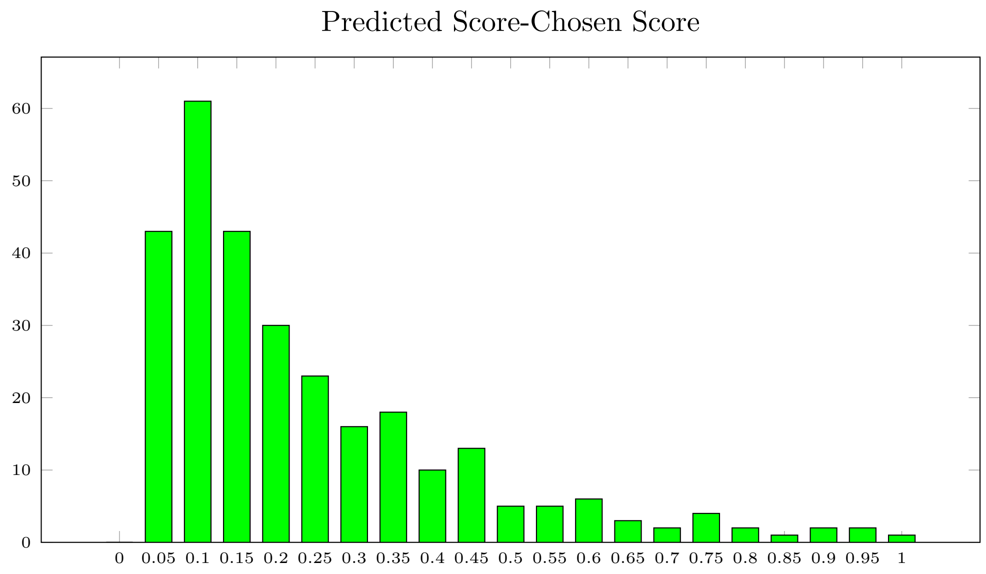


Figure 5.16: Number of occurrences for the difference between the predicted score and the chosen score for Version 2.

Different Diagonal Estimator

There was also another attempt at bettering the results using a different diagonal estimator than the one shown at the start of this section, the estimator used for this section was the following:

$$P_{X_iX_i} = 1 - \sum_{j;j\neq i} \left(\frac{pX_j}{pX_i + pX_j} \right)$$

The use of this estimators makes it so that the algorithm is "greedier" with its classification in an attempt to make it more certain of its decisions since in the previous algorithm most of the fails came from a difference between the picked score and the correct score of less than 0.2. The algorithm was tested using the same data for the previous one.

The results are on Table 5.14 and it shows that trying a "greedier" approach only exacerbated the problem that was previously prevent.

Figure 5.16 shows the graph portraying the difference between the chosen score and the picked score. For this version, this difference between the two is higher than on the other one, while on the first version a difference of 0.2 would mean around 80% of difference between the picked and the correct one, in this version a value of 0.2 only amounts to about 60% of the differences as seen on Figure 5.17.

5.1.4 API

Currently, there is an API to test the developed algorithm, as well as a website that makes use of this API to display the results in a very crude way. The website is available

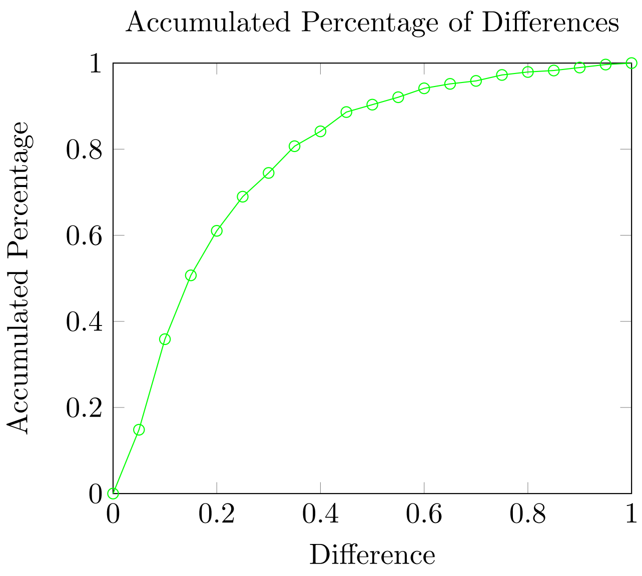


Figure 5.17: Accumulated Percentage of difference between the predicted score and the chosen score for Version 2.

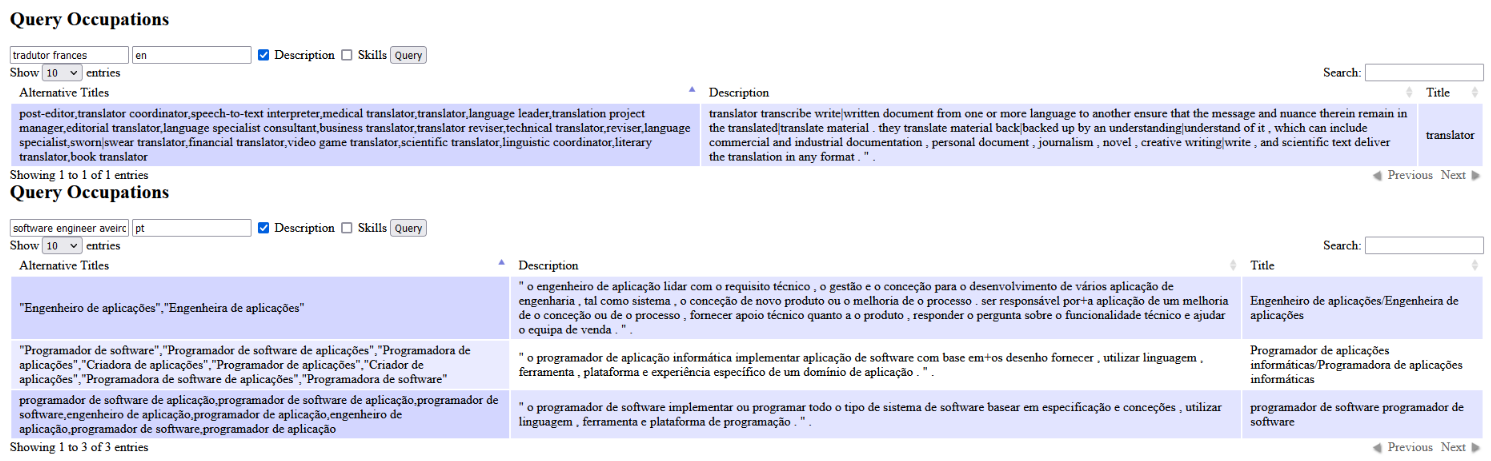
at `http://aal.ieeta.pt/oea-ws/` and one can query job titles of their choosing (in either portuguese or english), pick a language to be displayed (currently only supports portuguese and english) and ether the relevant skills should or not be displayed as well as the occupation description. It is even possible to query in english and obtain portuguese results and vice-versa.

The API makes use of what was talked before, a cutoff for the decisions it is unsure of. In case the score is not past a certain threshold this API will return three possible ESCO occupations matching the given query. These queries take about a second to display results due to the amount of queries to Elasticsearch that need to be done, with a proper service implemented these queries could take less time due to how scalable Elasticsearch is.

Figure 5.18 has an example of two queries from the website. One that returns an exact match, and one that makes use of the cutoff to display three different occupations. The queries were made to contain some "noise" to simulate real world job offer titles, these queries were "tradutor frances" and "software engineer aveiro".

5.2 Discussion

Producing this kind of results was no easy task. Working with data like these is hard, especially when it comes to trying to match it to the real world. Job offers with titles such as "designer para aveiro" or "mobilar developer" make this task that much harder due to the noise introduced by recruiters in the title or the typos produced by humans. Classifying a dataset like this into ESCO occupations by a human is challenging enough, and two different people could have drastically different views on what "designer para aveiro" could mean. Besides this, there is also implicit knowledge that is inherent to a country or to a specific region making classification of this type of data harder. To add on to these problems, the ESCO structure itself can also be confusing at times. Occupations that seem similar and feel


Query Occupations

tradutor frances | en | ☑ Description ☐ Skills | Query

Show 10 entries — Search:

Alternative Titles	Description	Title
post-editor,translator coordinator,speech-to-text interpreter,medical translator,translator,language leader,translation project manager,editorial translator,language specialist consultant,business translator,translator reviser,technical translator,reviser,language specialist,sworn|swear translator,financial translator,video game translator,scientific translator,linguistic coordinator,literary translator,book translator	translator transcribe write|written document from one or more language to another ensure that the message and nuance therein remain in the translated|translate material . they translate material back|backed up by an understanding|understand of it , which can include commercial and industrial documentation , personal document , journalism , novel , creative writing|write , and scientific text deliver the translation in any format . " .	translator

Showing 1 to 1 of 1 entries — Previous Next

Query Occupations

software engineer aveirc | pt | ☑ Description ☐ Skills | Query

Show 10 entries — Search:

Alternative Titles	Description	Title
"Engenheiro de aplicações","Engenheira de aplicações"	" o engenheiro de aplicação lidar com o requisito técnico , o gestão e o conceção para o desenvolvimento de vários aplicação de engenharia , tal como sistema , o conceção de novo produto ou o melhoria de o processo . ser responsável por+a aplicação de um melhoria de o conceção ou de o processo , fornecer apoio técnico quanto a o produto , responder o pergunta sobre o funcionalidade técnico e ajudar o equipa de venda . " .	Engenheiro de aplicações/Engenheira de aplicações
"Programador de software","Programador de software de aplicações","Programadora de aplicações","Criadora de aplicações","Programador de aplicações","Criador de aplicações","Programadora de software de aplicações","Programadora de software"	" o programador de aplicação informática implementar aplicação de software com base em+os desenho fornecer , utilizar linguagem , ferramenta , plataforma e experiência específico de um domínio de aplicação . " .	Programador de aplicações informáticas/Programadora de aplicações informáticas
programador de software de aplicação,programador de software de aplicação,programador de software,engenheiro de aplicação,programador de aplicação,engenheiro de aplicação,programador de software,programador de aplicação	" o programador de software implementar ou programar todo o tipo de sistema de software basear em especificação e conceções , utilizar linguagem , ferramenta e plataforma de programação . " .	programador de software programador de software

Showing 1 to 3 of 3 entries — Previous Next

Figure 5.18: Example of two queries with "noise" to test the API.

like they should have a similar occupation path start off in completely different.

Overall, the results were good taking everything into consideration, there are some points where better results could have been achieved with more resources, like the using more than one person to annotate the larger dataset, annotate more examples of the dataset. Training a model while having only 800 usable occupations and achieving a balanced F1-score of 83.839% with both a precision and recall of 83.839% can be considered a success. These results are good enough to be trustable, especially when it comes to the higher levels (1 to 3), which is the default used by both the ETF and the CEDEFOP (European Centre for the Development of Vocational Training), instead of the 5th. The results are also comparable to the state of the art shown in [23] that achieved an accuracy of 90% while aiming to only identify the first ISCO level, instead of trying to go for the 5th level.

Looking at the results after the algorithm has classified them, some decisions made by the algorithm can still be considered valid even if they were not classified as correct on the initial annotation. Looking at the numbers after evaluating the decisions made by the algorithm there were 66 more possible correct decisions it took, making the accuracy for level 5 jump from 63.38% to 72%.

There are also systems that can be created to further improve the model and make dataset annotation easier so that more data can be consumed by the algorithms. A system where if an occupation attains a low score is flagged and sent into a queue for human revision could help with this. After X amounts of revision the, model could be retrained and it would turn a fully supervised model into a semi-supervised model where the algorithm makes a choice, it is reviewed by a human, and then trained again.

The goal with the data visualization was to create dashboards similar to the ones made by the CEDEFOP. The example dashboards are shown in Figures 5.1 5.2 and the one made by CEDEFOP can be seen in Figure 5.19. The data presented by both dashboards is similar, albeit with different type of graphs, however, the dashboard made by CEDEFOP can only accomodate results until the 3rd ESCO level while with the algorithm here described there are results until the 5th level for both skills and occupations.

Besides this the web-service that was developed can already be a useful tool by itself. Given a free query this web-service uses our algorithm to fetch the relevant ESCO occupations for the given query (in case there are any). It can return results in both portuguese or english and the person querying the system can choose what it wants to see. This can be a useful tool for, for example, recruiters when trying to write a job offer fetching skills/knowledge or the description for the said job offer. This API is going to be used on the Observatório do Emprego de Aveiro (`http://observatoriodoemprego.web.ua.pt/en/`) website to provide classification to queries made. Moreover, the developed technology was also considered robust enough by a GOVCOPP (Research Unit on Governance, Competitiveness and Public Policies) research team which is using it to assess the competencies related to the industry 4.0 demand in the North region of Portugal.

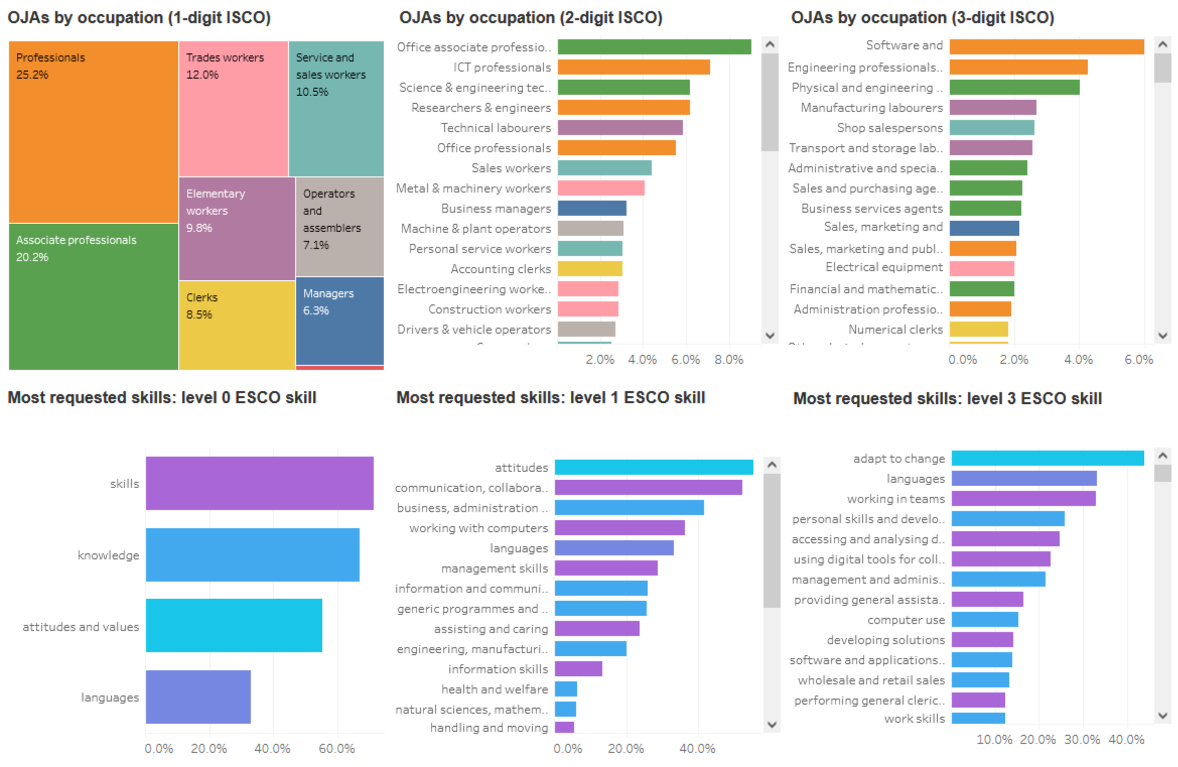


Figure 5.19: Dashboard with occupations and skills (until the 3rd level) for Europe made by the CEDEFOP [13].

Chapter 6

Conclusion

6.1 Summary of the developed work

The project started by creating a tool that would allow collection of data related to job offers for the "Aveiro" district of Portugal. This tool was created and currently pulls information from two different data sources converting them into a single, uniform data model. The data pulled is then uploaded to elasticsearch where the creation of graphs and dashboards is possible and made simple. These dashboards and graphs allows comparison between regions to know, for example, where the industry of a particular district or country is headed. As long as there is data these comparisons and dashboards can be created and updated in real-time due to the elasticsearch-kibana integration.

The next step in the project was to match these job offers with an ESCO occupation using only the job title. This task presented several challenges such as, lack of classified datasets, misspellings or useless information in job titles, ambiguous job offers that could match one or more occupation as they are defined in ESCO, the structure of ESCO having seemingly related occupations with completely different paths.

The general method used is to query the ESCO Elasticsearch instance for the job title and narrow down the search space from the 2500+ occupations ESCO has to a mix of 20 occupations and skills. From this initial query the feature vector is built based on the BM-25 score, link navigation, semantic similarity, and occupation similarity. On top of these results a model is applied to improve the BM-25 results.

There were two models tested one using a fully machine-learning model that was a multi-class problem and a mix of machine learning and state transition matrices that featured a series of binary classifications between the occupations to which the result was a percentage of chance of the occupation being the correct one.

For the first approach, the datasets were classified onto a multi-class problem and a feature vector was engineered with 60 features and 15 targets. This approach was not very successful even though it improved on the baseline (which was the Elasticsearch search) due to not having enough data, a huge class imbalance and the feature vector being sparse with many fields being equal to zero due to ESCO returning a different number of occupations for each query.

The second method used was a binary classification approach, this approach consists of instead of having a multi-classification model like before, we break these into a series of one versus ones and have a binary decision where the only thing to be decided is which occupa-

Algorithm	Level 1	Level 2	Level 3	Level 4	Level 5
Random Forest	0.8113	0.7313	0.69	0.66	0.6375

Table 6.1: Final Results attained by the algorithm

tion, between two, is most likely to match the query. Instead of getting absolute results from this model, the probabilities of an occupation winning over the other are extracted. These probabilities will then be used to build a probability matrix which will in turn be used to compute the steady state probabilities. This approach assumes that each candidate occupation is a state in a Markov model and the binary classification probabilities are approximations to state transition probabilities. The stationary probability is calculated and the final model gives a vector of probabilities for each occupation returned by the initial query being the correct one. This model achieved considerably better results than the previous ones and than the baseline so it was the best and final model.

As a proof of concept and to test out the algorithm, a very basic website was made where users can type out free queries and the algorithm will give out an answer, if it finds any. To test this website there were, as shown, studies made to find a good cut-off for when to display only one ESCO occupation, when the algorithm is absolutely positive this is the correct one, and when to display more than one, in the case of this website there are three currently being displayed when the algorithm is uncertain of which one to pick.

6.2 Main Results

While building the system from the ground up was challenging and met with many difficulties along the way, the final results were comparable to the state of the art while not having the resources to classify a big dataset. The fact that the model here presented performed so well with as few examples as it has and can easily be adapted to accommodate several languages (currently works in Portuguese and English), with little effort and as long as there are semantic similarity tools for the given language, is a major point in favor of this algorithm. Comparing this to the the results given by both the ETF and the CEDEFOP our whole system allows for a finer analysis, by city or region of a country, instead of going only by country and comparatively with the other state of the art articles here presented the focus was on ICT-related occupations while the work here shown tries to use ESCO as a whole.

The results obtained allow for a statistically significant analysis due to the metrics it reached and how similar they are when compared with the state of the art results. These final results can be seen on Table 6.1.

This work was already published at the 19th International Conference on WWW/Internet [32] and there are currently two Q1 journal publications being prepared.

6.3 Future Work

Right now there are plans for the future to build on the API and website. There are already tools in place to automatically generate the datasets (provided it is an already annoted dataset), classify the occupations and index them all in ESCO so it can be visualized as soon as the data is collected. This new API would improve on the first part of dataset generation and classification.

It would collect job offers and classify them, however, if the algorithm wasn't sure of its answer this job offer would be flagged and enter a queue system so that it could be reviewed by a human. The human would look at the options given by the algorithm and pick one or more as correct, so that it could enter the training dataset. After a given number of manual classifications, a new dataset would be generated with new and past job offers and the model would be retrained in order to, hopefully, get rid of these errors.

Adding more data sources to the data collection system is also a goal, however, this is a challenging one due to having to talk with different companies and request permissions either to get their data from an API or to webscrape which can prove to be difficult. LinkedIn would be the main one to add followed by net-empregos.

To improve on the algorithm results, a next step, would be to replace wordnet with word embeddings [24]. These are a more recent development on the natural language processing field that look at the positions of words in a text in order to classify them as being equal or not instead of using their semantic similarity. The decision to use wordnet was made at the start of the development due to the assurance that with Wordnet was possible to have fast enough runtime with the hardware that was available for the task. The use of brat [33] for annotations and trying to select the relevant bits of information from titles could remove clutter from the initial query and yield better results too.

It would also be worth testing trying to parse job descriptions using NLP and, for example, extracting useful words or skills from them to add on to the initial Elasticsearch query to see if this improves results or if it just adds noise to the queries.

Bibliography

[1] Malik Nabeel Ahmed Awan, Sharifullah Khan, Khalid Latif, and Asad Masood Khattak. A new approach to information extraction in user-centric e-recruitment systems. *Applied Sciences*, 9(14), 2019.

[2] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web: A new form of web content that is meaningful to computers will unleash a revolution of new possibilities. *ScientificAmerican.com*, 05 2001.

[3] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition, 2009.

[4] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. Dbpedia - a crystallization point for the web of data. *Web Semant.*, 7(3):154–165, September 2009.

[5] Jie Chen, Chunxia Zhang, and Zhendong Niu. A two-step resume information extraction algorithm. *Mathematical Problems in Engineering*, 2018:1–8, 2018.

[6] George Chryssolouris, Dimitris Mavrikios, and Dimitris Mourtzis. Manufacturing systems: Skills & competencies for the future. *Procedia CIRP*, 7:17–24, 12 2013.

[7] European Comission. European skills, competences, qualifications and occupations. `https://ec.europa.eu/esco/portal/home`, January 2021.

[8] Danica Damljanovic, Milan Agatonovic, and Hamish Cunningham. Freya: An interactive way of querying linked data using natural language. In Raúl García-Castro, Dieter Fensel, and Grigoris Antoniou, editors, *The Semantic Web: ESWC 2011 Workshops*, pages 125–138, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[9] Elastic.co. Information out: search and analyze. `https://www.elastic.co/guide/en/elasticsearch/reference/current/search-analyze.html`. Accessed on 14/06.

[10] Elastic.co. Similarity module. `https://www.elastic.co/guide/en/elasticsearch/reference/7.x/index-modules-similarity.html`. Accessed on 27/05.

[11] C. Fellbaum. Wordnet : an electronic lexical database. *Language*, 76:706, 2000.

[12] Luis Fernández-Sanz, Josefa Gómez-Pérez, and Ana Castillo-Martínez. e-skills match: A framework for mapping and integrating the main skills, knowledge and competence standards and models for ict occupations. *Computer Standards & Interfaces*, 51:30–42, 2017.

[13] European Centre for the Development of Vocational Training. Skills-ovate: Skills online vacancy analysis tool for europe. `https://www.cedefop.europa.eu/en/data-visualisations/skills-online-vacancies/occupations/skills`, 2021.

[14] Daniela Freddi. Digitalisation and employment in manufacturing. *AI & SOCIETY*, 33(3):393–403, August 2018.

[15] Aileen Gelpi. Focus on social media as vital element of graduate recruitment campaigns. *Recruiting & Retaining Adult Learners*, 21(10):12–12, 2019.

[16] Anna Giabelli, Lorenzo Malandri, Fabio Mercorio, and Mario Mezzanzanica. Graphlmi: A data driven system for exploring labor market information through graph databases. *Multimedia Tools and Applications*, 06 2020.

[17] Anna Giabelli, Lorenzo Malandri, Fabio Mercorio, Mario Mezzanzanica, and Andrea Seveso. Skills2job: A recommender system that encodes job offer embeddings on graph databases. *Applied Soft Computing*, 101:107049, 2021.

[18] Nicola Guarino, Daniel Oberle, and Steffen Staab. *What Is an Ontology?*, pages 1–17. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[19] Andrej Jerman, Mirjana Pejić Bach, and Ana Aleksić. Transformation towards smart factory system: Examining new job profiles and competencies. *Systems Research and Behavioral Science*, 37(2):388–402, 2020.

[20] Lorenzo Malandri, Fabio Mercorio, Mario Mezzanzanica, and Navid Nobani. Meet-lm: A method for embeddings evaluation for taxonomic data in the labour market. *Computers in Industry*, 124:103341, 2021.

[21] Joep Meindertsma. A brief introduction to linked data. `https://ontola.io/what-is-linked-data/`, July 2018. Accessed on 23/06.

[22] Fabio Mercorio. Can we use big data for skills anticipation and matching? The case of online job vacancies, 08 2019.

[23] Mario Mezzanzanica and Fabio Mercorio for the European Training Foundation. Big data for labour market intelligence: An introductory guide, 2019.

[24] Tomas Mikolov, Kai Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *ICLR*, 2013.

[25] Ryan Mitchell. *Web Scraping with Python: Collecting Data from the Modern Web*. O'Reilly Media, Inc., 2nd edition, 2018.

[26] A. B. M. Moniruzzaman and Syed Akhter Hossain. Nosql database: New era of databases for big data analytics - classification, characteristics and comparison. *CoRR*, abs/1307.0191, 2013.

[27] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. Scikit-learn: Machine learning in python. *CoRR*, abs/1201.0490, 2012.

[28] Mário Jorge Ferreira Rodrigues and António Joaquim da Silva Teixeira. *Advanced Applications of Natural Language Processing for Performing Information Extraction*. Springer, 2015.

[29] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959.

[30] Helmut Schmidt. Probabilistic part-of-speech tagging using decision trees. 1994.

[31] Toby Segaran, Colin Evans, and Jamie Taylor. *Programming the Semantic Web*. O'Reilly Media, Inc., 2009.

[32] Gabriel Silva, Mário Rodrigues, Marlene Amorim, Angélica Souza, Marta Dias, and Armando Pinho. Assessing job market dynamics using ELK stack. In *19th International Conference on WWW/Internet 2020*, pages 91–98, 11 2020.

[33] P Stenetorp, Sampo Pyysalo, Goran Topic, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. brat: a web-based tool for nlp-assisted text annotation. pages 102–107, 04 2012.

[34] Andrea Szalavetz. Industry 4.0 and capability development in manufacturing subsidiaries. *Technological Forecasting and Social Change*, 145:384–395, 2019.

[35] Udacity. Armenian online job postings. `https://www.kaggle.com/udacity/armenian-online-job-postings`, 2017. Accessed on 14/06.

[36] Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. Template-based question answering over RDF data. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, page 639–648, New York, NY, USA, 2012. Association for Computing Machinery.

[37] Marysol Villeda, Randy McCamey, Eyo Essien, Christian Amadi, and Rajunor Anani. Use of social networking sites for recruiting and selecting in the hiring process. *International Business Research*, 12:66, 02 2019.

[38] Duygu Çelik, Askýn Karakas, Gülsen Bal, Cem Gültunca, Atilla Elçi, Basak Buluz, and Murat Can Alevli. Towards an information extraction system based on ontology to match resumes and jobs. In *2013 IEEE 37th Annual Computer Software and Applications Conference Workshops*, pages 333–338, 2013.