Research Master Thesis

Semi-supervised Classification of Occupations using Pseudo-Labelling and Information Extraction

Lauren Green

a thesis submitted in partial fulfilment of the requirements for the degree of

MRes. Linguistics
(Human Language Technology)

Vrije Universiteit Amsterdam

Computational Lexicology and Terminology Lab
Department of Language and Communication
Faculty of Humanities

Supervised by: Antske Fokkens
2^{nd} reader: Marten Postma

Submitted: August 24, 2020

Abstract

Automated linguistic analysis using Natural Language Processing (NLP) and integrating ontological architectures into Human Resources (HR) process have proved to be extremely key tools in the industry's automation. Inevitably, an occupation is related to the key requirements and information within a given job description, whether this be based on the occupational title, the competencies, the education type, or all combined. However, extracting this relevant information from the raw text is a huge challenge, especially when reliant on human volunteers to manually annotate training data; this is both expensive and time-consuming. In addition to this, creating an ontology can prove to be a difficult task, especially given that many different occupations are often labelled as the same title, despite having very different requirements; without their related requirements, it can be hard to disambiguate two given occupations.

As a result, this thesis aims to leverage the information and requirements within job posting data to perform the multi-class classification of occupations, using ontology-based extracted information. Since the data is completely unlabelled, firstly we map out an *annotation guideline framework* to be used to annotate a batch of the data for gold label generation; making this a semi-supervised task. Then, a *distantly-supervised bootstrap system of pattern learning* is utilised, to extract relevant terms within the occupation descriptions using ontological seed lists as a basis. For the final component, namely the occupation classification model, the limited manually annotated data is then leveraged in a semi-supervised approach, formally known as *pseudo-labelling*. An exploration is therefore undertaken to determine to what extent the ontology-based extracted information will aid in this type of classification task.

Declaration of Authorship

I, Lauren Charlotte Green, declare that this thesis, titled *Semi-supervised Classification of Occupations using Pseudo-Labelling and Information Extraction* and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a degree degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Date: 23.08.2020

Signed: lcgreen

Acknowledgments

First and foremost, I would like to sincerely thank my supervisor, Dr. Antske Fokkens, for her continuous guidance, support and encouragement. Even whilst having undertaken this thesis during a pandemic and only being able to have our meetings and discussions online, she has dedicated her time and patience in answering my endless questions and sharing her extensive knowledge and experience. I really admire you as a researcher and cannot express my appreciation of your supervision enough.

I would also like to thank Greple GmbH for offering me the opportunity to write this thesis as part of an internship. It was a great experience to write my thesis in an industrial setting and I enjoyed being able to use my experience from the field of Human Resources in an applicable way for the thesis.

Finally, I would like to express my gratitude to my family, for their endless support and motivation throughout my entire university studies.

List of Figures

List of Tables

Contents

Preface

This research is performed as part of my academic studies for the Research Master (ReMa) Human Language Technology, undertaken at the Vrije Universiteit (VU), Amsterdam. After having studied at the VU for the past two years, we had the opportunity to conduct our final piece of work, namely this thesis, through a practical Internship. The aim of the Internship is to use the knowledge and skills ascertained through my master studies at the VU and apply them to a research project in a practical and more industrial setting. In my case, this research is performed as part of an Internship as a Human Language Technology engineer at a company called Greple GmbH.

Greple GmbH is a start-up company, founded in June 2018, who specialise in automating processing within the field of Human Resrouces (HR). Since, even in manual HR, subjective influences and unconscious bias within the workforce exist, the primary objective of the company is to use Artifical Intelligence (AI) and people analytics to ultimately try and avoid these biases; making the working world a fairer and smarter place. Greple GmbH work on a client-project basis in that they create and develop products for individual clients, depending on their demands/needs; they therefore receive project requests on a regular basis, each of which vary in terms of duration before project deadlines. It should also be noted that since their main clientele base is in Germany, the majority of the data in which they work with is in German. This project specifically, is a contribution to their Research and Development department.

After having already ascertained previous work experience in the field of HR and recruitment prior to my studies, this opportunity allowed me to thus apply both my working knowledge of the domain in combination with the skills and expertise learned at the VU.

The programming language used for performing the technical aspects of this research is Python.

The full code can be found in the GIT repository at the following link: `https://github.com/laurencgreen/researchMasterThesis`.

Chapter 1

Introduction

Occupational coding, or as referred to in this thesis, occupation classification, is the complex task of categorizing occupations into clearly defined classes aiming to describe the job labour market through the systematic aggregation of related data. Through undertaking such a task, a combination of statistical and predictive analyses can be undertaken into the job market to help with a variety of different socio-economic Human Resources (HR) and Recruitment processes. For example, by tracking salary rates for different occupations, otherwise known as *salary benchmarking*, and analysing the development of certain skills or occupation titles to determine whether they are becoming increasingly in demand or obsolete, the tasks of both *attracting* and *retaining* the right talent can be improved; these are formally referred to as *Talent Acquisition* and *Talent Retention*, respectively. However, with every company having its own unique organisational culture and language, it can be extremely difficult for them to attract and identify suitable candidates to fit into these conditions at not only a *technical* level, but also on a *cultural* level. A technical fit can be defined as a job seeker having the right set of skills, knowledge and ability to perform tasks within a role whilst a cultural fit refers to the "likelihood that [a candidate] will reflect and/or be able to adapt to the core beliefs, attitudes, and behaviors that make up [the] organization" (Bouton, 2015). Similarly then, each job seeker has their own set of aptitudes and personality traits which can only be exerted to their full-potential in the right working conditions for them. The hiring process can be both time-consuming and expensive for the prospect candidates as well as the companies, with the average cost for hiring and replacing a new employee being around 4,700 euros (Muehlemann and Pfeifer, 2016) notwithstanding the amount of time spent by candidates. If a company took the time to ensure a good balance on both fits, it could lead to a higher job satisfaction for their employees, improving the aforementioned HR practices and ultimately saving money, effort and time.

Before a *cultural* fit however, which is likely only be later determined during the interview process, the easiest and arguably most important aspect to initially match is the *technical* fit; without a match on this skill-related side, a good cultural match will deem redundant since the candidate will be unable to perform the tasks required for the given occupation. One of the most powerful and informative sources of initial information that a company can use to therefore reduce these costs for both sides by explicitly outlining these requirements is through the use of job postings. Job postings

are powerful sources of data defined as "official advertisement[s] of a [given] position for which [a] company is actively seeking a new-hire" and often comprise of a minimum of two main text fields, namely a job *title* and a job *description* (Boselli et al., 2018). Their highly influential aptitude is emphasised with the fact that they are often the *first ever interaction* that occurs between a given candidate and a company and therefore present the first impression to a job seeker. Although originally job postings were advertised on paper in newspapers, magazines and even bulletin-boards, following a rise in the popularity of internet usage and interest in automating the HR industry as a whole, there has been an increase in companies electing to utilise online social networking platforms by posting open vacancies online. By doing so, they are able to promote themselves as desirable businesses to work for and advertise potential employment opportunities at no longer just a local or national level, but on a *global* scale. Particularly then, the dynamic influx in the number of job advertisements being posted online has significantly impacted both the Talent and Job Search processes because of their increased exposure and improved accessibility. Using international job boards, including StepStone, LinkedIn and Indeed, it has been reported that the German job market has seen a total of 14,524,239 online job postings during the first 6 months of 2020 alone (Skills-OVATE, 2020).

This type of data source therefore plays a vital role in the HR industry and has the potential to both determine and predict future insights in the development of the job market through an analysis of their contents. A vacancy posting describes the key requirements and information related to a given occupation, whether this be based on the title, the competencies, the education type or all combined. With regards to occupation classification, there are multiple different types of schema that are utilised across the world, as explored in more detail in Chapter 3, with some electing to categorise occupations using the title alone, the skill *type* and/or *level*, the education or a mixture of all aspects together. Naturally, there are advantages and disadvantages to each of the systems, however for the purpose of this thesis, the schema selected categorises occupations using a combination both skills and education in a *hierarchical* structure, entitled the Standard Occupation Classification (SOC). The reason for this choice, is because of the challenge in the lack of *uniformity* that occurs when using the job title alone, as is used in some of the other candidate schema. Rather than utilising a systematic system, titles within job postings are selected manually by each individual company based on *their own understanding* of the position. This, in turn, not only means it can be difficult to disambiguate two different jobs with the same title but also introduces unconscious bias; subjective influences, conscious alongside unconscious biases are exerted into the recruitment processes by gatekeepers (HR professionals) which then exclude and discriminate against prospect candidates belonging to certain societal groups, albeit this may be unintentional. Moreover, by solely using the occupation titles for classification without a normalised system for title selection, inaccurate statistical predictive analyses may occur as a result of inaccurate title labelling and requirement alignment; despite the explicit requirements, *similar* occupations may have *syntactically different* titles, whilst different occupations may have titles that are syntactically similar. An example of this, emphasising the the importance of using the requirements contained in the descriptions, is shown below in *Figure* 1.1.

In *Figure* 1.1, the first job posting exhibits that the job requires a lot of hard, technical Information Technology (IT) related skills as well as a university degree whilst the sec-

1. Vacancy Title: **Projektmanager**
   - Vacancy Description: *"We are looking for an IT-Project manager. Qualifications: Experience using one or more of the following languages : Java/C++/Python. Degree in IT (Master/PhD), at least 2 years professional experience as Project Lead in the IT field"*
   - SOC Category: **Computer, Engineering, and Science Occupations**
2. Vacancy Title: **Projektmanager**
   - Vacancy Description: *"We are looking for a Customer Advisor/ Project manager. Qualifications: customer orientation and business thinking, experience in customer service, communicative and friendly nature"*
   - SOC Category: **Office and Administrative Support Occupations**

Figure 1.1: Two job postings of occupations labelled with the same written titles, "Projektmanager" (Project Manager), however each is categorised into a different occupational class from SOC based on their respective requirements

ond posting requests more customer-orientated soft skills with no mention of education necessary. Inevitably then, despite the fact that two jobs are labelled with the exact same job title, the occupations themselves are in fact very different and would therefore be classified into two separate occupational categories. Moreover, particularly in the German job market as is focused upon within this research, as a result of various socio-historical factors, one crucially being globalisation, there has been growing increase in Anglo-Germanic language contact that has inevitably led to an influx of anglicisms. An *anglicism* can be defined as "a word from British or American English brought into German or a change of a German word meaning or word use, according to the British or American model" (Zindler, 1959) [*Translated by Lauren Green*]. This has subsequently driven to even more variations of syntactic differences in job titles in the German job market, with many companies adopting these Anglo-transformations as exoticisms or as a sign of culture, inevitably to make them more appealing to the consumer, or in this case the job seeker.

Given the evident abundant number of syntactic variations that exist in natural language in this scenario alone, the manual coding of occupations undoubtedly can be "a demanding, time-intensive [and non-trivial] activity because in each case the [expert] coder must decide anew which category from the long list is the correct one" (Züll, 2016). In addition to this, a human expert will require extensive training in order to manually annotate each of the occupations which is both expensive and time-consuming. In a continuously growing digitalised world, these traditional and manually-expensive tasks are being enhanced by artificial intelligent (AI) applications. Attempts have been made to therefore automate this process using a combination of Machine Learning (ML) and Natural Language Processing (NLP) techniques, however because of the challenges in ascertaining high quality labelled data and automatic data normalisation, determining the correct mapping of occupations to classes still proves to be an extremely challenging task.

Furthermore, the descriptions themselves within job postings are raw, unstructured text which often not only include information about the occupation alone, but instead entail descriptions of the companies themselves as well as what the company may have

to offer. Since the task of occupation classification in our research is to categorise the occupations based on the *skill* and *education* requirements, as mirrored from SOC, it is necessary to filter out this irrelevant information. For this reason, extracting the occupational *relevant* information with regards to the *requirements* from raw text also poses as a challenge, notwithstanding the fact that this type of information extraction (IE) system requires large numbers of *labelled* data instances in order to ascertain highly-accurate, industry-ready results.

This research is therefore split into two core components; term extraction and occupation classification. The term extraction system serves as part of the feature engineering for the occupation system itself, whereby the requirements from the raw job postings are both identified and extracted. This component also serves as a comparative model, whereby the performance of the classification models will be compared both with and without the extracted terms; the aim is to emphasise how automatically classifying occupations using the extracted related requirements could lead to improved classification performance.

As mentioned in the Preface, this thesis is undertaken as part of an internship for Greple GmbH. Although there is an abundance of data accessible, this is the first research undertaken for the task using it and therefore this data is not labelled. In order to therefore obtain this partially labelled data for the occupation classification system, a set of annotation guidelines will be created for the data to be annotated by the highly skilled HR experts employed by Greple. Similarly, the same job posting data is also completely unlabelled for the term extraction system therefore the exact alignment of requirements for each occupation is not readily available. For clarity, this means that each of the job advertisements do not have any *gold* labels with regards to which occupational class they should belong nor do they have any requirement terms explicitly labelled. Although in an ideal scenario another annotation framework would also be created, specific to the Term Extraction task to at least obtain some gold-standard testing data, this is not possible due to time-constraints within this thesis. Nonetheless, it is still possible to generate a semi-supervised system using the external data resources provided. Automated linguistic analysis using NLP and integrating ontological architectures into the HR process have proved to be extremely key tools in the industry's automation, with Greple currently building their own Ontology; an Ontology can be defined as a structured knowledge base which encodes all of the key information regarding a domain in a systematic and organised format. Thus, despite the fact that we do not have any explicit labelled data in relation to the job postings, the current state of this Ontology can be *leveraged* by using the structured seed lists to extract the relevant, key terms.

The overall aim of this thesis then is to be an exploratory study by analysing and comparing the effectiveness of using specific occupational requirements and the full vacancy descriptions for the classification of occupations. Since Ontologies contain domain-specific knowledge, with key information already structured into specific classes, it is hypothesised that the ontology-based extracted features will help to create an improved classification system. In a further analysis, a comparison between different ML, NLP and Deep Learning techniques will also be made, tailored for this specific task. Although DL has also made improvements in multiple state-of-the-art in NLP related applications, its predictability can vary drastically across tasks based on not only the size, but also the accuracy of the dataset. Often an abundance of precise data, directly

impacts the levels of its performance. One of the well-known feature of semi-supervised systems is that they are able to leverage the information contained within a small labelled dataset

As a semi-supervised task, this thesis will focus on leveraging the limited labelled data to maximise efficiency and performance, and ultimately"exploit the hidden information in [the] unlabelled dataset" (Pintelas et al., 2020). A closer analysis into what extent the performance of the system using this type of learning will also be undertaken, comparing the results of the classification in a limited labelled supervised setting as well as using all of the unlabelled data in a semi-supervised environment. Here, due to the fact that a well-known attribute of semi-supervised learning involves shifting the decision boundary between classes using the unlabelled instances, it is hypothesised that this will in fact improve results. However whether DL itself will aid in improving the performance is an open question.

The overall process will firstly involve extracting relevant terms from job advertisements in a distantly supervised manner using the ontological seed lists. These extracted terms will then be used to classify and cluster occupations into the pre-defined occupational classes. The final model should therefore be able to classify a given occupation into one of the given groups, based on the requirements within the respective job advertisement. In an ideal situation, both the term extraction and the occupation classification would be evaluated however due to time-constraints, only an *extrinsic evaluation* is done of the former. This means that although an exploratory analysis of the final results is undertaken to some extent, the main focus in evaluation will be upon impact of the Term Extraction on the overall task of the Occupation Classification System. The classification system, on the other hand, is evaluated using F1, precision and recall.

1.1 Research Questions

This leads us to the main points of comparison within this research, formulated into three individual research questions:

1. To what extent can the Ontology based extracted features aid with the classification of occupations?
   - *Since Ontologies contain domain-specific knowledge with key information already structured into specific classes, it is hypothesised that these features will help the occupation classification.*
2. Despite it being used in many state of the art NLP-related tasks, how does Deep Learning compare to Machine Learning for this task?
   - *Both a SVM and CNN are used as a comparison here based on recent works for other multi-class classification tasks. Due to the fact that DL models usually require extensive amounts of training data to perform well, it is unsure how the CNN's performance will fair against more traditional ML SVM system.*
3. Given the complexity of implementing a semi-supervised approach using only a limited number of gold labelled instances, to what degree will the pseudo-labelled data not only change the decision boundary between classes, but also improve a

given system's performance.

- *Results for each of the experiments will be both undertaken in a supervised setting only utilising the manually labelled data available, as well as in a semi-supervised setting. It is hypothesised here than although this may not affect the dramatically affect the results for the SVM, it will have a greater impact on improving the performance of the CNN as increasingly more data is used as training input*

The main findings of this thesis prove that the semi-supervised pseudo-labelling approach utilised does in fact not only adjust the decision boundary, but in most cases improve the overall performance. Particularly regarding the DL experiments, the increasing amount of labelled data clearly results in a better performing system. However, in the case of the SVM, the large amount of data leads to exhausted computational resources and a negatively distorting decision boundary, resulting in a lower performance. Nonetheless, the SVM is able to employ a more calibrated and stratified approach when dealing with the imbalance of classes as used in this research, of which the CNN is unable to do with little data. The ontological extracted features, namely the skill and education related requirements, generally do improve the performance of the system, however further filtering and post-processing of them is necessary for maximal effect. The final results prove that incorporating the extracted features using an ensemble approach, whereby a calibrated SVM is implemented at the first iteration of the pseudo-labelling architecture followed by a continuation of CNNs, achieves the highest result of 0.62.

1.2 Outline

By firstly exploring the theoretical background and recent work for occupation classification and information extraction in Chapter 2, the fundamental foundations and understanding needed for this research are both described and summarised. This will lead us into the methodology in Chapter 3, which is split into four main sections; data collection, annotation framework, preprocessing and the overall system pipeline. Since the data within this research is completely unlabelled after describing the data collection, we propose an *Annotation Framework* to be used to manually annotate a subset of the data and generate *gold* labels for both testing and *training*; making the classification system a semi-supervised task. Following this, the preprocessing steps undertaken for each of the systems are outlined individually, since each require different pipelines. The subsequent Chapters 4 and 5 provide information and explicitly explain each of the systems in detail. The results however of the overall classification system with regards to each of the comparative experiments are presented in Chapter 6. These results are then presented and examined through an in-depth Error Analysis in this same chapter, before finally leading us to the Conclusion and any future recommendations in Chapter 7.

Chapter 2

Theoretical Background and Recent Work

Job postings are powerful data sources for the task of occupation classification because of their informativeness when it comes to the most integral requirements for a given occupation at hand. Although predominantly job postings include information about the role or occupation itself, they are ultimately still advertisements with the aim of *attracting* potential candidates and therefore often they also include information unrelated to the occupation itself; entailing details about the company, benefits and any other "interests [or] preferences of the specific people [the] organisation want[s] to attract" Kerzazi and Adams (2016). As a result, with the purpose of this research in classifying occupations using the related requirements, a combination of Natural Language Processing and special filtering techniques is needed to extract only information related to the given occupation itself. This chapter thus outlines both the theoretical foundations and previous related work that form the basis of this thesis, serving its purpose of introducing, and build thereupon, the core phenomena and knowledge necessary for understanding the existence of the research problem. Since there is no *directly* correlated related work, mirroring our data *type* and *language* nor the exact methodology, the chapter is divided into various sections, addressing each of the key components.

Firstly, in Section 2.1, an overview of the different Machine Learning (ML) supervision techniques is explored, with specific focus on the approach utilised in this thesis, namely *semi-supervised* learning. Here, we address and take into consideration the various submethods that can be used for classification tasks as a whole where there is a *limited* amount of *labelled* instances in comparison to *unlabelled* data. Following this, in Section 2.2, as we utilise an ontology as an additional resource, we look into the power of the *semantic web*, particularly concentrating on the power of *ontologies* and how they can be leveraged for our specific research. Since this thesis ultimately consists of two components, a detailed insight is undertaken into the recent work of both systems. In Section 2.3 we subsequently undertake a detailed exploration of similar research for the overall task of *occupation* classification before then finally investigating previous work for *term extraction* in Section 2.4. Each of the systems for both components of similar work are explored in terms of the method applied alongside preprocessing steps and input features.

2.1 Machine Learning

Here an overview of basic concepts within the field of ML is provided, with specific focus on the type of supervision utilised within this thesis, namely *semi-supervised* learning.

ML is a sub-field of Artificial Intelligence (AI) which can be defined as utilising "algorithms [which] can learn from data without relying on rules-based programming" (Faggella, 2020). Ultimately then, ML involves using state of the art techniques to process algorithms and *learn* from the data, whilst also having the capability of making predictions for future use through *generalisations*; being able to interpret and observe patterns within data, ascertaining valuable insights which can be used for future, unseen data, whether this be numerical or as is used in this thesis, textual data.

Unlike Deep Learning (DL), a subfield of ML itself in which deep neural network (NN) architectures entail *hidden* layers, traditional ML techniques typically require more structured data. For transparency, the core differences between them lies in the fact that DL makes use of language models trained on unlabelled data whereas traditional ML employs explicit linguistic features that are obtained usually through preprocessing. Nonetheless, despite these distinctions, both techniques have proved to be very effective within multiple different Natural Language Processing (NLP) challenges and should each be considered individually for implementation dependent on the task itself. There are multiple different types of learning techniques used within ML, with the most common including *supervised*, *semi-supervised* and finally *unsupervised* learning.

Supervised ML algorithms rely heavily on large collections of manually annotated data in order to make efficient predictions. Although supervised learning is proven to ascertain high accuracy scores, inevitably, the large amount of manually labelled data needed to train the algorithms can be expensive, in terms of both time and cost; a prerequisite of supervised learning is the accessibility of readily available labelled data (Sekine and Ranchhod, 2009). Some of the most common supervised *traditional* ML algorithms used include *Support Vector Machine* (SVM), *Naïve Bayes* (NB) and *Logistic Regression* (LR), whilst common supervised *DL* algorithms include *Convolutional Neural Network* (CNN) and *Long-Short Term Memory* (LSTM).

Unsupervised learning, on the other hand, is capable of learning relationships between features and mapping these structures using unlabelled data. Here, models can be trained using unannotated data in order to predict, optimise and evaluate a system through adjusting *weights* of the distributed vector representation and learn patterns in the data by itself. These networks are able to learn from all of the information that is inputted into the machine as well as the feedback that is given after it makes the predictions. Although unsupervised techniques are a lot more cost efficient in comparison to supervised methods since they do not require any labelled data during training, often this is compromised with achieving less accuracy.

For the purpose of this thesis however, a combination of both of these approaches is utilised, formally known as semi-supervised learning, as explored in more detail below.

2.1.1 Semi-Supervised Learning (SSL)

Following the rise in technological advances and the popularity of internet usage, obtaining unlabelled data has become relatively easy because of its increasing abundance. However, despite this ease in accessibility, the manual labelling of data by human experts is both time-consuming and expensive. Alternatively, without any labelled data or guidance, it can be difficult for a model to obtain precise information from the data sorting, since the output labels are unknown. An optimal solution to this therefore, is *semi-supervised learning* (SSL).

SSL is a learning paradigm in which a mixture of both labelled and unlabelled data is used during training. For this reason, it is a powerful alternative for leveraging unlabelled data, particularly when there is limited labelled data, which is both cost and time expensive to obtain (Odena et al., 2018); saving both time, money and without jeopardising accuracy. As a result of its ability to leverage a small sample of labelled data in relation to larger amounts of unlabelled data to generate a *more precise* decision boundary, semi-supervised learning is becoming increasingly more important. Multiple text classification tasks have also been shown to achieve improved results through using a SSL over a supervised approach with a model's error being reduced by up to 30% (Nigam et al., 2000).

SSL can be divided into two different types of learning, namely *inductive* and *transductive*. Inductive learning refers to *traditional supervised* learning whereby a model is trained using the labelled training data before the label predictions are made upon the *unseen* test data. Transductive learning, on the other hand, involves utilising both the *labelled* and *unlabelled* data during training, making predictions on *unlabelled* instances; the information, features and relations learned from the labelled data points are used to infer and predict the *correct* labels of the *unlabelled* instances. Examples of both of these methods include bootstrap methods of *self-training*, *co-training*, *transductive* methods and *graph-based* methods. For the purpose of this thesis, only more detail into the type of approach implemented within this research is undertaken, namely *self-training*.

2.1.2 Self-Training (ST)

The most simplest form of semi-supervised learning is known as *self-training* (ST). Initially applied to the task of *Word Sense Disambiguation* (WSD) (Yarowsky, 1995), *Yarowsky's Bootstrap self-training algorithm* followed two powerful heuristics for the task; *one sense per collocation* (Yarowsky, 1995), whereby the meaning is determined by the context and *one sense per discourse*(Yarowsky, 1995) in which the meaning of an ambiguous word will be consistent throughout the entire given corpus. Here, this ST bootstrap method involves labelling training data using a supervised classifier and then through labelling the new data in the same way, more words will be associated with a certain meaning/sense. This tactic is then repeated iteratively with more training data.

Variations of this ST approach have since then been further researched, particularly following the rise in interest in DL. Rather than just taking *all* of the predictions made by the model at each iteration, new methods of ST have been introduced using a threshold for limiting which predictions to add to the labelled data and fine-tuning DL models based on the results. Although these types of thresholds vary, one of the

most common and efficient is using the confidence level in a *pseudo-labelling* (Lee, 2013) approach; as is applied in this research.

2.1.2.1 ST: Pseudo-Labelling

Originally developed for training deep neural networks, pseudo-Labelling (Lee, 2013) is a specific variant of semi-supervised ST, based on the same theoretical foundation as that of *Entropy regularization* (ER) (Grandvalet and Bengio, 2005), known as the *cluster assumption*. The *cluster assumption* ultimately assumes in a classification setting that the instances within the data are grouped, or clustered, in a close proximity dependent on the class in which they belong; "two points are likely to have the same class label if there is a path connecting them passing through regions of high density only" (Chapelle et al., 2003). As a result, if data points labelled as the same class are grouped closely together whilst instances from different classes have increased distance apart, the *true* decision boundaries are driven through regions of low density (Grandvalet and Bengio, 2005). Through using a low-density separation between classes in this semi-supervised setting (Lee, 2013), *pseudo-labelling* is able to leverage a small amount of labelled data to iteratively train a model and generate *confident* predictions on the unlabelled data, whereby the maximum predicted probabilities, or *confidence* levels, are generated for each unlabelled instance during training. Pseudo-labelling classification models are therefore often guided to generate *confident* predictions and generate *pseudo-labels* with the aim of improving the generalisation power through the unlabelled data (Lee, 2013).

The *basic* main workflow of the method is as follows: the available labelled train data is used to train a model before the *pseudo-test* data is then used to generate predictions for each of the test instances. After these predictions are generated, the confidence score of each is calculated; if this is above the certain threshold, then the labelled instances get added to the labelled data whilst the remaining are added back to the unlabelled data. This process continues for n iterations or until all of the data has been labelled. It is important to understand here that the pseudo-labels are taken as *true labels* and therefore it is recommended to select a high threshold for the confidence.

As a result of its simplicity and generality, pseudo-labelling is becoming an increasingly used heuristic in practice (Odena et al., 2018) and the realm of semi-supervised learning. Nonetheless, it is important to note that despite its *intuitive* nature it may not always be the optimal solution; often producing false predictions of the unlabelled instances if the base classifier isn't capable of learning representative patterns between the classes (Odena et al., 2018). This measn that if the model generates *incorrect*, yet still *confident* predictions, the decision boundary will become negatively distorted. For this reason, adaptions of this pseudo-labelling paradigm have also been researched in an attempt to overcome this challenge, divided into *two* variations.

The first and more *traditional* approach involves repeatedly training a *new model* after each iteration when the pseudo-labelled data and the original labelled data are concatenated together each time. This is a simple, yet effective, method for labelling the unlabelled training data and has proven to ascertain high performance in many text classification tasks. Alternatively, the second variant considers iteratively training a given model in the same way in which it would be within a supervised setting, however rather than training the *same* model or a *new* model each time, the *weights* from the

previous model are saved each time and fine-tuned on the new *pseudo-labelled* data each time; simply here, the *last checkpoint* of the model is saved once it has converged.

Although there has not been extensive research on this variant with regards to *text classification* itself, the overall theory can be applied to other classification tasks in the field of ML and NLP.

In earlier works, Liu et. al (Liu et al., 2004) build a text classification model using the *first* variant, there is little to no labelled data. Here, they use iterative clustering for feature selection of the documents in order to determine the most representative words of each class which are then fed into two different classification models as a comparison; Naïve Bayes (NB) and the expectation-maximization (EM) algorithm. This process continues by training a new classifier at each iteration however, they do not select any confidence threshold for the classification systems; instead, manual intervention is undertaken during the word extraction, though inevitably this would be too time-consuming and inefficient for our purposes.

Babakhin et al. (2019), on the other hand, implement the *second* variant of the pseudo-labelling approach to overcome their lack of labelled data by training an ensemble of Convolutional Neural Networks (CNNs) for the *segmentation of salt bodies* in image classification (Babakhin et al., 2019). Here, they start with 4000 labelled images and 18000 unlabelled images to be *pseudo-labelled* by their system, for a maximum of *3* iterations for *200* epochs; if not all of the images are labelled by this point, the unlabelled data is discarded. Unlike previous approaches however, the original labelled data and pseudo-labelled data is *not* concatenated together each time. Instead, the *second* and *third* iterations are trained only using the pseudo-labelled data before then fine-tuning on the original labelled data; in their specific experiments the joining of the pseudo and original labelled data yielded a lower performance. Nonetheless, similar to previous work, the "unreliable predictions" are still filtered out by removing all pseudo-labelled instances that are assigned with a *low-confidence*; where the predicted confidence is below a given threshold (Babakhin et al., 2019).

Inevitably, although the second approach could help the model learn *better*, it could also lead to *over-fitting*. Diversely, the first variant would avoid any issues of potentially overfitting the model, however this also runs the risk of not learning enough information from the data given. Since the performance of both types of this pseudo-labelling approach highly depend on the the type, quality and size of data at hand, it is not clear which may prevail in our research. As a result, both are therefore carefully considered in this research.

2.2 Ontological Representations

Since the Ontology developed and provided by Greple is used to aid with the methodology in this thesis, this section offers a brief overview of *ontological representations* as a whole, exploring to what extent this additional resource can be leveraged as a powerful tool in this research. Considering the aim here is to provide the foundations needed for understanding why exactly this specific data model is selected, only a *brief* overview of the Semantic Web itself is described. The main focus instead lies on describing what exactly an ontology is, how they can be applied to different ML and NLP applications and why it is chosen for this specific task at hand.

In the world today, the accessibility and abundance of data that can be found online continues to exponentially increase as a result of the increase in popularity of the internet and the *World Wide Web* (Web). It should be noted here that terms internet and Web are not synonymous; the internet is the global system of interconnected computer network, otherwise referred to as the network of networks (Panda, 2020), whilst the Web refers to a model used for *sharing* information which is created on the internet itself and is one of the mediums that can be used to access the internet. With more people being therefore being able to share information on the internet through the Web, the internet has become one of the main sources for accessing data. However, despite this increasing supply of information, one of the biggest problems this leads to is the multiple different formats in which the data can be captured within, such as in resumes, professional social media profiles and job postings as in the HR industry. This multitude of information captured in different structures and forms therefore makes it difficult to understand the relations between each of them. In order to overcome this issue and ultimately allow these relationships become visible, the *Semantic Web* is introduced.

The *Semantic Web* can be defined as "an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation" (Berners-Lee et al., 2001). Rather than comprising of a collection of individual webpages like that of the Web, the idea of the Semantic Web is to create a single repository allowing for both data and information to be shared in a systematic, organised and free way, capturing the relationships between them. Using the definition described by Berners-Lee et al. (2001) as a basis, the Semantic Web should therefore not be regarded as a *separate* Web, but rather an *extension* of it. With its inter-connected properties between concepts in the form of triples, the Semantic Web is often referred to as the web of *linked data*. These triples thus illustrate the relations between concepts in the following manner: subject (concept 1), predicate (relationship) object (concept 2).

Over the last decade, there has been increasing interest in one of the most renowned building blocks within the Semantic Web, namely *Ontologies*. An Ontology is a data model that "represents knowledge as a set of concepts within a domain [through describing both the concepts as well as] the relationships between these concepts" (Sathe et al., 2017). A single Ontology therefore does not only contain the data itself of the domain at hand, but also entails a set of *metadata* which specifically *describe* the data by providing semantic information about its context and use. Thus, providing new data and meta-data to the existing documents within the Web. Through creating one main repository of *linked* data between all of the concepts within a domain, using various data sources, Ontologies allow for not only the sharing of re-usable knowledge but also allow for new knowledge about the given domain to also be incorporated and further shared (ont). Their organised format enables them to be easily and continuously populated with new information as a domain grows and adapts over time.

Particularly when there is a lack of labelled data, Ontologies have demonstrated to be powerful resources for multiple Knowledge Representation and NLP related tasks, such as *Named Entity Recognition* (NER), *Ontology population* and, as is utilised in our case, *IE*. Techniques involving the latter can be used to aid in the population of the Semantic Web whilst, on the other hand, the IE process can be guided by Semantic Web techniques(Martinez-Rodriguez et al., 2020). The structural and informative properties

within an Ontology, which often also include hierarchical relations linked between the various concepts, therefore provide intelligent direction for an IE system.

As a result of the organised domain-specific information captured within Ontologies as a whole, the data within the Ontology provided by Greple serves as a vital and powerful component within this research, offering a source of *labelled* seeded data to be used as leverage when developing the *term extraction* system; this is specifically explored in more theoretical detail in *Section* 2.4 and practically in *Chapter* 3. In addition to this, the current state of the Ontology used only takes in the *candidate* perspective, through capturing all of the information within professional social media platforms and resumes. Inevitably when developing an Ontology, it should capture all of the knowledge within the given domain; for this reason, the *client* perspective should also be incorporated through using the job postings, as written by the clients themselves. Consequently, the results ascertained in this research could also be used in future works to both populate the Ontology as well as provide further inferences to its current state. The implementation in using this Ontology as an additional *labelled* data source, will therefore not only allow association with meaning through the domain as a whole, but also enable a more insightful and expert-guided experiment when extracting the relevant terms of the data classes as well as during the error analysis.

2.3 Occupation Classification

Here an exploration is undertaken into similar work in relation to the different approaches which have been utilised for occupation classification (OC) as a whole. For each work, we discuss the following, where available: the core underlying approach, the data, the features, preprocessing steps, results and any future work comments. Through understanding the full-pipeline and background of similar research, this Section aims to build the foundations for the methodology and purpose of this specific research. It should be noted, however, that there are no directly correlated previous work that utilise the same *semi-supervised* approach and language data as is implemented within this research; although some of the previous related work utilise different *supervision* approaches, we still investigate them with the purpose of understanding the overall pipeline in terms of objectives, data cleaning and feature engineering. Additionally, despite the fact that the data utilised within this thesis is in the *German* language, the core body of similar work in the field is focused towards *English* data, likely as a result of its popularity and larger number of resources available for research.

2.3.1 Supervised

In earlier works, Goindani et al. (2017) implement a supervised binary classification system for determining the employer *industry* using job posting data. Here, two datasets are used for the two core industries focussed upon, namely the *Transportation* and *Health Care* industry; each of the datasets entail *positive* and *negative* examples in relation to the given industry. Minimal preprocessing is used here since the job titles and employer names are extracted from an existing external resource, *Career Builder* whereby each name is linked to a knowledge base (KB). Nonetheless, each of the normalised titles and keywords are converted into two separately calculated feature vectors, before then being concatenated together for one final vector representation. Imbalance in the classes so use *cost-sensitive learning* for training; "different costs are assigned for

misclassifying examples belonging to different classes" (Goindani et al., 2017), where the positive classes are more penalized than the negative classes. Cross-validation is also utilised. For the classification itself, two supervised algorithms are utilised; (1) Support Vector Machine (SVM) and (2) Gradient Boosted Decision Trees (GBDT). Although for the *transportation* industry the SVM outperforms GBDT with an F1 score of 0.81 in comparison to 0.79, the GBDT exceeds that of the SVM for the *health care* dataset, scoring an F1 of 0.86 compared to 0.82. For future works which could also be relevant in this research, they suggest to include additional features, such as whether a job description contains specific key terms as well as verbal string patterns such as *our client is looking for* (Goindani et al., 2017). We can assume that their suggestion of using *job-content keywords* could refer to the requirements of a specific occupation as well as any task related terms. Moreover, they note that some employers belong to multiple industries despite just having one label, referred to as "industrial conglomerates" and therefore for future works would consider the task as a *fuzzy classification* problem to output multiple class labels. As a result, this should be taken into consideration for this specific research, as some occupations may have *fuzzy* boundaries between two classes, where only *one*, single label is assigned.

Van Huynh et al. (2019) utilise deep neural networks (NN) in another *supervised* approach however rather than predicting an occupational class from the description, instead they focus on classifying the related job title using the requirements within the postings; knowledge, skills, interests. Here, the job *descriptions* from online postings are used to predict 25 different job titles, all of which are related to the IT sector. A combination of different NN are implemented, both alone as well as ensemble methods to use as a comparison through the use of *majority voiting*, including Bidirection Gated Recurrent Unit (Bi-GRU), a Convolutional Neural Network (CNN) and Long-Short Term Memory (LSTM). For the preprocessing, each of the job descriptions are converted to lowercase and tokenized, before special characters and stopwords are removed. As a final step, the text is thereafter transformed into feature representations using one-hot encoding alongside two pre-trained language word embedding models as a comparison, namely GloVe and FastText. Regarding the results, each of their proposed ensemble methods outperform the single models, including the TextCNN + FastText which scored an accuracy score of 0.69; the highest performing model is the Bi-GRU-CNN + GloVe scoring an accuracy of 0.72. For future works, it is suggested to expand the current dataset of 10,000 manually annotated instances as well as to explore with more *traditional* classifier in order to do more feature engineering. Although this approach again uses a supervised method, it is also the first to utilise DL techniques alongside a comparison in *word embeddings* at different levels whereby both types illustrate improvements in performance against using simple one-hot encoding transformations.

2.3.2 Semi-Supervised

Djumalieva et al. (2018), on the other hand, developed a *semi-supervised*, *distantly-supervised* hierarchical multiclass classification of occupations based on the *skill-related requirements* within online job postings for the United Kingdom (UK) job market. Using a total of *four* layers within the hierarchy, the first *three* layers build up granularity for clustering occupations with similar skill *types* together whilst the final layer utilises the salary to indicate the skill *level*. The highest level entailed a total of 16 classes are

utilised, otherwise referred to as *broad groups* and is based based on the UK Standard Occupation Classification (SOC) taxonomy, whilst the final level, the *skill level*, comprised of 150 *salary clusters*. Although unlike the previous aforementioned works the the full job descriptions were not available, *most* of the job postings within Djumalieva et. al research still comprised of a *job title*, a *salary* and a set of related *keywords*, otherwise referred to as *skills*. Particularly relevant for this research, is their finding during Exploratory Data Analysis (EDA) that multiple occupational categories within their data were under-represented, such as *agriculture* skill-related occupations, whilst other categories were largely over-represented, such as *business* skill-related occupations. For this reason, a *semi-supervised* over an unsupervised approach is implemented since the former is more likely to capture even the classes of which only a small number of job postings are contained within (Djumalieva et al., 2018). As a result, *seed lists of skills* and competencies from both ONS 2010 Index (Office for National Statistics)[1] and the European Dictionary of Skills and Competencies (DISCO)[2] are utilised to guide the semi-supervised classification. Regarding the preprocessing, two different approaches are implemented for both the *job titles* alongside the *keyword-skills*. For the *job titles*, the following steps were undertaken: plural forms to singular, abbreviation transformation using a dictionary, digit and punctuation removal, remove words not in *known dictionary*, removal of "and" from beginning or end of string and finally additional whitespace removal. For the *skills*, on the other hand, "inappropriate skills and language skills" are manually removed before being converted to lowercase, stripping most punctuation and removing extra whitespace characters. These final terms are then transformed into US English for normalisation and ease of string matching. Both the skills and job titles are then converted into vector representations using pre-trained word embeddings, namely *GloVe*. The overall preprocessing pipeline is summarised in *Table* 2.3.3. Both unsupervised and semi-supervised methodologies were initially compared, namely *Latent Dirichlet Algorithm* (LDA) and *k-means clustering* respectively, however due to a potential lack of *keywords*, LDA produced less stable results. As a result, k-means accompanied with the cosine score to calculate the distance between each of the clusters is primarily utilised. The cluster stability is then evaluated using the *Jaccard Coefficient*. For future works, Djumalieva et. al suggest to train "an occupation-specific word embedding model to improve the accuracy of job assignment to reference categories". give the example of "scrum" which currently links to more *leisure and sport* related categories because of its association to rugby, however in the occupational context it should be associated with *agile software development techniques*.

Another semi-supervised system for the classification of occupations for the UK job market, again using SOC, is developed by Turrell et al. (2019), however with focus on using one specific datasource of vacancies, from *Reed* [3]. Here, both the job title and the full job description are utilised in order to classify occupations at a more granular level within SOC, referred to as the 3 digit level. The basic pipeline of their methodology includes cleaning of the vacancy text, matching job titles to the SOC titles at an *exact* matching level before thereafter finally identifying similar SOC codes using *fuzzy* matching; grouping an occupation to the SOC code determined by the distance between the vectors calculated using the *cosine* similarity. For the preprocessing steps

[1] `https://www.ons.gov.uk/`
[2] `http://disco-tools.eu/disco2_portal/`
[3] `https://www.reed.co.uk/`

of the vacancy data, plurals are normalised through conversion to their singular forms, abbreviation transformation, stopword removal, digit removal, punctuation removal, extra whitespace removal and removal of non-salient words; *salient* words are determined through the use of Term Frequency- Inverse Document Frequency (TF-IDF). In addition to this, words related to the job *level* are removed, including terms such as *senior* or *junior*; both a *senior financial analyst* and a *financial analyst* should be categorised in the same way (Turrell et al., 2019). Finally, following an in-depth EDA, frequent words discovered across the job postings as a whole are further filtered and stripped since do not carry and real semantic role regarding the type of occupation; e.g. *candidate* and *role*. Although it is noted that supervised method may ascertain higher accuracy, the maintenance costs would be higher to get the manually annotated data. As a result, semi-supervised learning is implemented through the use of a *known dictionary* of job titles and their associated SOC codes for training. For linguistic feature engineering, each text field is thus transformed into a TF-IDF feature representation for the exact and *fuzzy* matching and n-grams of the most *salient* terms are used as an additional feature. For future works, rather than just outputting the soc code itself, they suggest a probability or confidence level for each class could also be given for human intervention for *marginal cases*.

2.3.3 Summary of Recent Work for OC

A summary of each of the preprocessing steps for the similar occupation-related classification tasks is illustrated in *Table* 2.3.3. Although, as mentioned, the main sources of data for each of them is in fact in the English language, the table shows that these *linguistic* data cleaning steps can still also be applied to the *German* data at hand; the steps are more *task*-related rather than *language*-dependent. It should be noted that some additional preprocessing steps may need to be taken into consideration, however this is explored in the subsequent Chapter 3.

	Goindani et al. (2017)		**Van Huynh et al. (2019)**	**Djumalieva et al. (2018)**		**Turrell et al. (2019)**	
Data	Title	Keywords	Description	Title	Keywords	Title	Description
Language	English		English	English		English	
Lowercase	*pre-normalised*	*nm*	*True*	*True*	*True*	*True*	*True*
Plural-Singular			*nm*	*True*	*False*	*True*	*True*
Abbreviation Transformation			*nm*	*True*	*False*	*True*	*True*
Digit Removal			*nm*	*True*	*False*	*True*	*True*
Punct Removal			*- only special chars are removed: (#, &, *, $ etc)*	*True*	*False*	*True*	*True*
Stopword/ Non-relevant word Removal			*True*	*- "and" before/end of sentence*	*- "inappropriate" skills - language skills*	*True*	*- stopwords - non-salient words - job level related i.e. "junior"*
Additional Whitespace Removal			*nm*	*True*	*False*	*True*	*True*

Table 2.1: Summary Table of the Preprocessing Steps undertaken for Occupation Classification in Recent Work ("nm": no mention)

Overall, it is evident that each of the aforementioned similar work primarily utilise *two* text fields; the *job title* alongside *either* relevant keywords *or* the job description itself, dependent on availability. However, none of the previous work mention utilising *all three fields* as sources for occupation-related classification. In addition to this, although a

seed list of keywords of requirements for various *occupations as a whole* is available for this thesis, unlike the previous semi-supervised methods, our seed list is not *specific to each job posting* itself and therefore these methods cannot be *directly* applied to our research task.

Nonetheless, since Greple GmbH specialise in the field of Human Resources (HR) and therefore have experienced HR experts at hand, this domain-specific knowledge can be leveraged for annotating *gold* instances for both *training* and *testing* for training a semi-supervised classifier; using a *pseudo-labelling* approach as mentioned in *Section* 2.1.2.1. Moreover, since our data already entails both a *job title* alongside the *description* for each posting, it is possible to therefore extract relevant *keyword requirements* from the descriptions themselves, enhancing on the previous methods through using *all three text fields* as features. As a result, we now explore recent work for the second component of this thesis, Term Extraction and investigate how these methods can be used for extracting keyword requirements from the job postings.

2.4 Term Extraction

Term Extraction, or terminology extraction, is a subtask of *Information Extraction* (IE), with the goal of transforming unstructured, or often semi-structured, data into structured representations by automatically extracting of *clear* and *factual* specific information.

There are multiple different approaches which have been applied for the task of Term Extraction, with some of the most common approaches being *feature based* classification and *pattern-learning* systems. The former approaches have proved particularly successful for Term Extraction in different domains, with some using Gazetteers and lists of dictionaries as *features* in themselves. An example of one of the most effective models which have been used here is Conditional Random Field (CRF) (Gupta, 2015). Although these types of classification systems can yield high results, the challenge in implementing in our research is that they require at least some partially labelled data to train the algorithm, which is not available in our case.

Pattern-based learning approaches, on the other hand, can be both supervised and *distantly* supervised, whereby the latter require only a small sample of *seeds*. These seeds can be *patterns* in themselves or, alternatively, a set of *terms*. One of the most advantageous aspects in the development of pattern-learning techniques is that they have proven to be particularly successful when there is already a certain *format* and *structure* within the unlabelled; this can be leveraged to generate clear syntactic contextual patterns surrounding new terms in a powerful way. Furthermore, increased research in *distantly-supervised* pattern-learning has demonstrated to still ascertain a high performance even when there is limited labelled data, which can be expensive to obtain in both terms of time and cost.

Since the job posting data utilised within this research does have a structure *to some extent*, whereby each posting is *searching* for a candidate with certain *skills* and other *requirements*, it is decided that this *distantly-supervised* pattern-learning approach is most suitable for this research. Moreover, although the job posting data is completely unlabelled for this Term Extraction sub-task, as mentioned in *Section* 2.2, we can leverage the available data at hand, in the form of Greple GmbH's *ontology*, as an

additional resource for generating a sample of seeds.

This Section therefore discusses variable *pattern-learning* approaches in a *distantly-supervised* setting for the task of IE, specifically Term Extraction. Within each work, we explore the different patterns, features, preprocessing techniques and scoring methods utilised for each as a comparison. Both systems which use *terms* and/or *patterns* as seeds are investigated

2.4.1 Distantly-Supervised Pattern Learning

Pattern-learning based Term extraction, otherwise known as Entity Set Expansion (ESE) is a distantly supervised task whereby a set of seed entities is given as the only supervision for the system to try and expand this list by trying to extract new *similar* terms. Unlike other unexplainable *black box* methods, distantly-supervised bootstrapped ML patterns allow for effective performance, time and money savings alongside interpretability by humans (Chiticariu et al., 2013), as is important for this research. As a result, pattern-learning methods are known as one of the most robust systems for extracting relevant key terms, often referred to as a *feature selection* approach, whereby the patterns are used as the feature *templates* themselves (Gupta, 2015). There are two variants in terms of input in this bootstrapped ML pattern-based technique; seed list of *patterns* or seed lists of *terms*, though the latter has ascertained more research in recent years. Inevitably without any types of threshold or *scoring* system of the patterns or the terms, the approach would generate *incorrect* lexico-syntactic patterns which would thus lead to the extraction of *irrelevant* new terms; leading to a low performance. Here, we investigate similar work in terms of both input seeds and scoring functions for pattern-based ML, all in a distantly supervised setting.

Riloff introduced the classical bootstrapping algorithm, known as *AutoSlog-TS* (Riloff, 1996) whereby the system extracted terms from unlabelled text only using a set of *seeds* as labelled input. Here, the model is fed a seed list of *terms* from a specific class and some heuristic rules to iteratively learn the surrounding syntactic patterns to find new terms, ultimately expanding the original dictionary set. The heuristic rules thus activate the system to therefore extract relevant noun phrases. Although this system does not use any semantic features as a result of "technical reasons", it is noted that this would have improved the precision (Riloff, 1996). *AutoSlog-TS* system ranks patterns based on the number of positive terms extracted. Each lexico-syntactic pattern is scored using a function titled *RlogF*; if the pattern is scored above a certain threshold, it is added to the pattern dictionary, else it is discarded. Ultimately then, this calculates the weighted conditional probability through summing the number of positive terms over the total number of terms extracted by a single pattern.

Thelen and Riloff (2002) build further upon the aforementioned work utilising the same *RlogF* metric for scoring however, their system, *Basilisk*, is adapted for multi-class extraction; here the begin with a value of N=20 for the first iteration. Basilisk also implements a *term scoring* system here based on the frequency of the patterns and in which only the *head noun* of each Noun Phrase (NP) is added to the *candidate* pool.

Pasca (2004), on the other hand, uses seed *patterns* rather than *terms* as the sole input, using syntactic structures which are not domain specific. Examples of patterns utilised include *Hearst Patterns*, originally used for Hyponymy detection, to learn

both *named entities* and their *categories* from unlabelled web-pages. Hearst patterns (Hearst, 1992) are explained in more detail in *Section* 2.4.2 with examples illustrated in *Table* 2.2.

English	**German**
Y such as X	Y zum Beispiel X
X and other Y	X und ander[e/n] Y
X or other Y	X oder ander[e/n] Y
such Y as X	beispielsweise Y wie X
Y including X	Y einschließlich X
Y, especially X	Y, insbesondere X

Table 2.2: List of Hearst Patterns with their German translations; X is the Hypernym and Y is the Hyponym

Gupta and Manning (2014) developed a system whereby new terms are detected and extracted through bootstrapping from just a small sample of seed dictionaries. In an iterative process, "patterns are learned using labelled entities, and [terms] are learned based on the extractions of learned patterns" (Gupta, 2015) until neither anymore patterns nor entities can be learned. As an improvement of their slightly earlier work, here they score patterns additionally through the prediction of labels of the *unlabelled* entities using a combination of Edit Distances for fuzzy matching alongside TF-IDF scores (Gupta and Manning, 2014).

So far, the aforementioned work does not evaluate any of the extracted terms on a *semantic* level, since they only rely on the syntactic rules generated by the system. Nonetheless, in more recent work, Gupta (2015) built further upon her previous work in developing one of the most powerful pattern-based learning frameworks capable of evaluating the extracted terms *semantically*, using Word2Vec [4] word embeddings. Based on the distributional hypothesis (Harris, 1954), the degree of *similarity* between two given terms can be calculated using these word vector representations to determine how semantically similar they are. In order to do so, both the seed terms and extracted candidate terms are transformed into word vectors before calculating the cosine similarity score. If the similarity scores above a certain threshold it is added to the seed list. Fuzzy matching of the tokens is also undertaken for calculating the *fuzzy* similarity score between tokens using the *edit distance* metric, however this is only as a comparative component to the word embeddings; inevitably the semantic measure ascertains higher performance. Unlike the previous systems, the preprocessing steps are also indicated in this work; tokenization on both a word and sentence level, tagging Part of Speech (POS), to be used as a feature, before then lemmatizing both the seeds and the data, for generalisation, and finally converting to lowercase for normalisation (Gupta, 2015). Regarding the contextual window size, a range between two to four word before or after a labelled tokens is utilised. This allows the system to be able to therefore learn the POS features of the surrounding words, generate patterns and ultimately extract new terms in a more robust and automatic way, taking the *context* and semantics into consideration.

Since then, similar work has been done also using word embeddings for calculating different *similarity* scores, such as in the work by Batista et al. (2015) in extracting the

[4] `https://code.google.com/archive/p/word2vec/`

relationships between two given terms. Here, a seed set of *relations* is instead used as input to the model with the task of further expanding it. For the *relationship* scoring system, again here a baseline of TF-IDF is compared with using the distributional semantics within Word2Vec word embeddings. They note here however, that despite the high performance of word embeddings, they can still introduce the *semantic drift*. The semantic drift can be defined as "the progressive deviation of the semantics for the extracted [items] from the semantics of the seed [items]" (Batista et al., 2015). As a result, this work avoids the semantic drift by both scoring the patterns and ranking the extracted relationships however only by comparing with the *original* seeds rather than the set of seeds extracted at each iteration. Again here, if a the score of a given relationship is equal to or exceeds a certain threshold then it is added to the dictionary set, else it is discarded. The extracted relationships are then used during the following iteration of the bootstrapping algorithm. Although this work is for *relationship* rather than *term* extraction, similar preprocessing is undertaken again here, with the tokenization again on both the word and sentence level, tagging POS, lemmatization however with the addition of using Named Entity Recognition (NER) for capturing local relationships.

Roller et al. (2018) however, compare pattern-based methods with distributional approaches to evaluate their performance for the task of detecting Hyponymy relations. Similar to Pasa, Hearst Patterns again are used as one of the baseline methods. In addition to this, POS tagging and dependency parsing are used as features for the distributional models. Here, the data alongside the patterns themselves are tokenized as well as lemmatized as part of the preprocessing, in order to *maximise* the number of exact matches where possible. Despite the high performance of distributional models in other works, this work illustrates Hearst patterns to outperform them.

Finally, in most recent work, Kim et al. (2019) developed a bootstrapping approach for biomedical *term* extraction, again only using a small sample of seeds, however rather implement the addition of a NER classification model for the learning of patterns using only some features. The two models used as a comparison are CRF and LSTM. Rather than manually labelling data for their classification systems, instead the classifier used to automatically label the training data in an iterative way, with the aim of improving The classifier each time. The system thus takes both the original seed corpus in addition to the iterative machine labelled data as input, combined, in a CoNLL 2003 format; each word is represented on a single row with its respective features in a tab-separated style. Biomedical domain word embeddings are used again for this system, namely BioSQ, to handle the domain specific out of vocabulary (OOV) terms which may not be present in *normal* word embeddings. Regarding the preprocessing steps, although not explicitly stated, it can be assumed that the text and seeds are lemmatized since ngrams of the term lemmas are used as a feature for the classifier. Further features include ngrams for POS tags, capitalisation (first letter and all letters). The final results illustrate the *third* iteration of the bootstrapped classifier to achieve the highest performance, of 71.87% in comparison to the initial one which only scores 43.88%. For future works, they suggest to "apply external resources [and other] approaches, such as lexico-syntactic patterns [to] improve the quality of the machine-labelled data" (Kim et al., 2019).

2.4.2 Pattern Creation

As illustrated below, a summary of the features used in the aforementioned previous work shows the variety of different *lexico-syntactic* and *linguistic* features that can be used for creating patterns.

- **Lexico-Syntactic Patterns:** Lexico-syntactic patterns do not take into consideration *semantic* or *grammatical* rules rather they just focus on the syntactic rules.
  - **Regex Patterns:** One of the most basic patterns for IE, or more specifically ESE, is the use of regex patterns which use explicit lexical items for pattern creation. These can be particularly useful for data with significant syntactic properties. In our use case of *job advertisements*, an example of a regex pattern could include "Skills should include X".
  - **Hearst Patterns:** These lexico-syntactic Hearst patterns (Hearst, 1992) are one of the most influential approaches to IE despite their original development for detecting classic Hyponymy relations (Roller et al., 2018). Examples of these Patterns with their German translations can be seen in *Table* 2.2. One thing to note about Hearst Patters is despite the fact they can ascertain high *precision*, often this is accompanied by low *recall*; they are very accurate however because of their scarcity they only encompass a minority of patterns. Jacques and Aussenac-Gilles suggest using synonyms of the patterns to improve their scarcity. For example, "such as" could be replaced with "like" in English (Aussenac-Gilles and Jacques, 2006).
- **Linguistic Features:** Manually crafted linguistic features using NLP techniques can be used to generate patterns by extracting the information encoded within each of the contextual surrounding words to that of each seed. These features offer a more *generalised* procedure.
  - **POS:** One of the most important integral features used for ESE is the use of morpho-syntactic POS tagging. This feature is particularly useful when the class of terms to be extracted is of a certain type. For example, a *skill* can be both a *noun* or *adjective*. POS tagging is also imperative for lemmatizing text, whereby words are reduce to their *root* form.
  - **Dependency Parsing:** Dependency parsing encodes information about various linguistic features, such as POS, morphology as well as grammar. As a result, this feature can be considered a good basis for not only syntactic, but semantic patterns to illustrate the relations between terms in a given string; Hearst Pattern-like features may also be captured by distributional representations such as dependency based features (Roller et al., 2018).
  - **Capitalisation:** As seen in some of the recent work, capitalisation of either an entire word or just the first letter is used for aiding in the pattern creation. Some acronyms or skills may be fully capitalised therefore this feature would help to capture this information. Although not all of the previous work use this as a feature, it should be noted that this is probably as a result of transforming the English text data to *lowercase* for normalisation. Particularly in German however, nouns are always capitalised therefore this

could be a useful feature for this research, dependent on the class of terms for extraction.

2.4.3 Summary of Recent Work for TE

Each of the aforementioned approaches within similar work demonstrate how a small number of labelled seed instances can be powerfully used to develop a system capable of extracting terms from unlabelled corpora. Inevitably, the introduction of using distributional word embeddings has been a revolutionary turning point in capturing the *semantics* within text, however as seen in work undertaken by Roller et al. (2018), the syntactic properties captured in using these *basic* patterns should not be completely discarded. In addition to this, despite the latter work by Kim et al. (2019) using a combination of bootstrapping and a classifier could be particularly useful for our research, as a result of time-constraints and because of the lack of labelled data even for our test set, it is not implemented. It could however be used in future work, building further upon the research undertaken for this thesis.

This research will therefore use a combination of the aforementioned methods, through utilising a list of seeds for each of the relevant classes of terms to be extracted, a set of patterns as well as certain selected features to be used for generating additional patterns following the first iteration. For the scoring, a mixture of the proposed algorithms will be used; (1) to capture any misspellings or typos in the data, edit distances can be a powerful tool for obtaining a fuzzy matching *term* similarity score (Gupta and Manning, 2014) (2) to capture the *semantic* similarity, distributional semantics have illustrated to yield a high performance. The technique in avoiding the *semantic drift* used by Batista et al. (2015) in using only the *original* seed patterns will also be implemented, but adapted in using the original seed *terms*. Although TF-IDF is used as a baseline comparison in most of the work, it will not be utilised in this thesis and instead we aim to further build upon the previous research using *word embeddings*; evaluating vector representations at different levels will instead be used as a comparative feature. A deeper analysis and exploration into the methodology used in this thesis is undertaken in *Chapter* 3.

Chapter 3

Methodology

In this Chapter, I present the full methodology pipeline of this research, mapped out in 4 aggregated steps, namely the *Data Collection*, *Annotation Framework*, *Preprocessing* and the *Overall System pipeline* for both the *Term Extraction* and *Occupation Classification* systems combined.

Firstly, the Data Collection in Section 3.1 provides explicit information regarding the three different job posting data sources alongside the seed lists extracted from the Ontology. A combination of descriptions alongside statistical distributions and visualisations are displayed. Since each of the sources are taken from different platforms, this section further describes the process of re-formatting the data through aligning the headers before then finally outlining the steps undertaken to extract a sample of the entire dataset, as a result of computational limitations.

Subsequently, Section 3.2 presents the Annotation Framework, in which the occupation classification schema selection, the annotation guidelines themselves alongside the intermediary analysis and inter-annotator agreement scores are discussed.

The Preprocessing in Section 3.3 involves two main phases, namely *basic* preprocessing which is applied initially to the entire dataset together as part of the *annotation data preparation*, as well as the *final* preprocessing, whereby two individual data preparation pipelines are applied for the classification and term extraction systems, respectively.

Finally, a brief overview of the entire system overview is mapped out and described in Section 3.4 for both the term extraction alongside the occupation classification. Each of the systems themselves however are explored and outlined in more detail in Chapters 5 and 4, respectively. For the purpose of this research, it should be noted that all of the computational techniques are applied using *Python*

3.1 Data Collection

The first step involves collecting the *job posting data* itself provided by Greple GmbH. This data is firstly queried and downloaded from the company database, Kibana. For the purpose of this research, data from *three* different sources are used, namely *Stepstone*, *Historical Vacancies* and *Vakanzen*. Whilst both the Historical Vacancies and

Vakanzen data resources entail a combination of job postings from various different job boards during the last 5 years, the Stepstone[1] data is specific to the single job board in the same time frame. Inevitably then, each of the datasets are formatted slightly differently in terms of both their contents alongside the syntactic forms in their headers. To filter out and determine the most relevant information necessary for this task, the data is explored and headers thus aligned, as described in more details in Section 3.1.1.

In addition to the job posting data, as mentioned in the previous chapter, Greple GmbH are currently building an Ontology containing a variety of information about occupations from the *candidate* perspective. Since this data model will be leveraged within the methodology for the *Term Extraction* system, acting as the main *labelled* resource, this data is also collected now. In order to do so, seed lists from each of the relevant classes are collected from the Ontology to be later preprocessed in the same means as the job posting data. Since the goal of the Term Extraction model is to extract only the core requirements related to a given occupation, namely the *skills* and the *education*, the related terms from each of the classes are withdrawn from the Ontology. The meta-data extracted for the skills contains both the *main* skill-related terms themselves, as well as further data describing them. In order to thus maximise the number of instances available and ascertain additional coverage, both the *main* skills alongside the related *synonyms* are extracted and concatenated together. At the time of writing this thesis, the Education class is not yet incorporated within the Ontology. However since it is hypothesised that this is a key requirement in defining how exactly an occupation can be defined, instead a list of Education-specific seeds is manually curated by both myself and a native German HR expert. To maintain quality, this list is kept at a minimal number; the quantity of seeds will thus be used as a comparative module within the Term Extraction system, outlined in more detail in Chapter 4. A summary illustrating the total number of seeds within each relevant class is illustrated below in *Table* 4.1.

Class	**#Instances**
Skills	1,004
Skill Synonyms	909
Education	12

Table 3.1: Number of instances within each of the class seed lists for the Term Extraction

3.1.1 Header Alignment

Since the job posting data is originally crawled from various websites and ultimately contains semi-structured information, the data in its raw form contains a multitude of information organised into various columns with headers. For example, alongside the *vacancy title* and *vacancy description*, other information such as the *language*, *locality*, *company name* as well as the *skill* terms used to crawl the job postings initially. In total, each posting entails around 19 different columns. After an exploration into the information contained within each and the extent of their relevance for this specific task, a total of 4 headers were selected. As shown in Table 3.2, after having aligned

[1] `https://www.stepstone.de/en/`

the headers based on their contents, it is evident that not all of the same information is contained within each of the data sources. Nonetheless, through their utilisation it still enables for some filtering of the job postings, for example through using the *language* header to filter out the postings which are not in German; as mentioned for the purpose of this thesis, using only postings in the German language are focussed upon.

Stepstone	Vakanzen	Historical Vacancies
_id	**_id**	**_id**
content.vacancytitle	**content.title**	**content.title**
content.vacancydescription	**content.description**	**content.description**
	source.language	source.language

Table 3.2: Aligned column headers from each of the data sources. Headers in bold indicate all three data sources contain same information so headers can be fully aligned.

3.1.2 Data Sampling

After having dropped all rows whereby the vacancy title or vacancy description was dropped, a total of 4,525,33 instances were left. However, due to computational power, each of the datasets are then also sampled and used as the main datasets in this research, as shown in the final column of *Table* 3.3.

Data Source	#rawInstances	#preprocessedInstances	#finalInstances
StepStone	549,948	77,512	**77,512**
Vakanzen	4,092,698	4,069,188	**150,225**
Historical Vacancies	441,000	378,638	**150,225**
Total	5,080,646	4,525,338	**377,957**

Table 3.3: Number of instances from each dataset in the raw data, the data after dropping missing values and after sampling the final instances to be used within this research.

3.2 Annotation Framework

In the world today, the classification of occupations is a key component in automating HR processes because of it's ability to group *similar* occupations together in a consistent and standardised form. Inevitably, as a result of the variances within job markets across the world, there are an abundance of different occupation taxonomies and schema used for occupation classification. Although the approaches and methodologies do vary to some extent for each of these schema, they all serve a similar purpose; grouping occupations "based on work performed and, in some cases, on the skills, education and/or training needed to perform the work" (Emmel and Cosca, 2010). By organising jobs into explicitly defined groups based on their given requirements in both a standardised and structured way, not only can it allow for a deeper analysis in occupations which share similar requirements, but it can also aid in building more refined systems, such as a offering insights into an individual's career progression within the same domain and thus developing a more robust recommendation system. However choosing such a system to use for our specific purposes can be difficult, without knowing what the underlining reasons for each of their variances.

As a result, this section presents an annotation framework for manually generating *gold* standards for the classification of the occupations within the job vacancy data at hand. Firstly, a short exploration and comparison into the three different aforementioned occupation classification schema will be made, before illustrating the reasons for selecting the framework of choice for this thesis, namely SOC. Following this, a detailed description of the level of aggregation within the framework is provided, along with its specific classes. As the final and main component within this section, we describe the methodology for the annotation process itself, including the annotation data preparation, intermediary analysis milestones as well as the inter-annotator agreement score; the latter part also includes the final result. For the full annotation guidelines used by the annotators themselves, please see Appendix A.

3.2.1 Schema Selection

As mentioned, there are a multitude of different frameworks used across the world, each with the shared aim of classifying occupations into some type of defined groups. However, since each of these systems vary in what exactly entails this type of classification, it is important to understand the reasons for each of these variances before selecting which one to use for our specific use case. After some initial research, the three main frameworks which are taken into consideration are: (1) the *German Classification of Occupations* (KldB) [2] (2) the *International Standard Classification of Occupations* (ISCO) [3], and finally, (3) the *Standard Occupation Classification* (SOC) [4].

Since the majority of the data at present is based on the German job market, the first type of classification system considered was a German based occupation classification, namely the KldB 2010. The KldB contains around 27,000 job titles and has a total of 5 different aggregation levels within its hierarchical structure, with the main level (level 2) containing 37 classes, otherwise referred to as *Occupational Main-Groups*; the level just above this (which is also the highest level) contains just 10 classes, or *occupational areas*, however as according to the official guidelines, these cannot be utilised as they are "not suitable for analytical purposes" (Paulus et al., 2013).

In contrast to the other occupation frameworks, KldB focuses primarily on classifying the job titles (Züll, 2016); "the education and training required for the competent performance of the job [...] are not classification criteria. [...] (Hoffmeyer-Zlotnik and Warner, 2013). Inevitably then, despite the fact that this is the only German occupation classification schema, using this type of classification approach would pose a problem for our research since we aim to use education as a factor for distinguishing different occupations.

In addition to this, although similar to the other frameworks the KldB is updated periodically (in this case every 5-10 years), at the time of writing this thesis the most recent version of the KldB was published a decade ago. As even stated on their own website, "the professional landscape is subject to constant changes. New jobs are created, old job profiles lose their importance or disappear completely." (original "*Die Berufslandschaft unterliegt ständigen Veränderungen. Es entstehen neue Berufe, alte Berufsbilder verlieren an Bedeutung oder verschwinden komplett*" (Kld). Thus, using

[2] `https://fdz.iab.de/187/section.aspx/Publikation/k140117303`
[3] `https://www.ilo.org/public/english/bureau/stat/isco/`
[4] `https://www.bls.gov/soc/`

an outdated system as the basis could lead to a loss of information for grouping the more recent occupations.

Aggregation Level	#Classes
Areas	10
Main Groups	37
Groups	144
Sub-groups	700
Types	1286

Table 3.4: Level of Aggregation levels within KldB-10 with the respective number of classes

The next option to be considered for the research schema is ISCO; ISCO is the "standard categorization most commonly used worldwide for reporting and comparing occupational information data" (CHOI et al., 2019) and has a total of four different aggregation levels within its hierarchical structure; see *Table* 3.5. Similar to KldB, the highest level entails 10 classes which in ISCO, is the major, *main*, level. ISCO groups occupations however not only by using the *type* of skill required within a certain occupation, but also by the skill *level*. This skill *level* is measured in various ways, with a core aspect being the *level of formal education* and *informal training*, where the latter may refer to the amount of previous experience (isc).

Nonetheless, although this system does therefore utilise *education* as a factor within its classification process, the last version of ISCO was updated in 2008 which is even older than the latest version of KldB itself. In addition to this, for time-constraint purposes of the thesis we are limited to only choosing the *major* level of aggregation; it would not be possible to annotate enough instances for each class within any of the other levels during the time given.

Aggregation Level	#Classes
Major	10
Sub-major	43
Minor	130
Unit	436

Table 3.5: Level of Aggregation levels within ISCO-08 with the respective number of classes

The final schema for consideration is the Standard Occupation Classification (SOC). SOC is used "for the purpose of collecting, calculating, analyzing, or disseminating data" developed by United States (US) government agencies although, there are also national versions across other countries such as United Kingdom, Canada, Spain the Philippines and Singapore. Similar to both KldB and ISCO, SOC also has a hierarchical structure with a total of 6 levels of aggregation, as illustrated below in *Table* 3.6.

As a result, despite the fact that is not predominantly a structure built specifically for the German job market, given the purpose of this research being that occupations should be classified by both *competencies* and *education type* and that our data is up-to-date from the past 5 years, it is decided to use SOC as the main basis for the classification schema.

Aggregation Level	#Classes
Highest	6
Intermediate	13
Major	23
Minor	98
Broad	459
Detailed	867

Table 3.6: Level of Aggregation levels within SOC with the respective number of classes

In order to aid with the annotations themselves, SOC also offers an up to date Crosswalk tool with *O*Net* [5], a freely accessible online database and includes detailed knowledge on various aspects within a given occupation, such as *knowledge*, *skills* and the *education* level. This *O*NET-SOC* combined crosswalk provides not only a detailed and structured system for classifying occupation, but also allows for a an easier annotation process; by being able to search for occupation titles and keywords to find similar matches and ultimately aid in determining a given class. Moreover, the most recent update of the O*NET-SOC crosswalk is just in 2019, inevitably meaning it contains the most current information of the job market with up to date occupational data. Although inevitably this information is not in German but rather English, since the purpose of this structure is to classify occupations as *concepts* with similar attributes rather than as single *job titles*, this should thus not affect the annotations.

3.2.2 Guidelines Development

Given the fact that there are multiple different levels of aggregation within the SOC hierarchy, after careful consideration, the *intermediate* level is selected, which includes a total of 13 different occupational classes as shown in *Table* 3.6. As pointed out by Turrell et al. (2019), there is often a "trade-off between more granularity and more accuracy in classifying jobs according to the correct SOC code". As a result, by choosing this particular *intermediate* level, it reduces the granularity in comparison to the other levels, including the major group itself, however still allows for an automatic two-tier hierarchical structure with the highest class just above. This higher tier could also be used if either the inter-annotator agreement isn't sufficient or if there are not enough instances within a given class, the annotation labels can also be easily mapped to the high aggregation level to generate larger samples for each.

The 13 annotation labels used in this project are exemplified in *Table* 3.7 in the "soc label" column, alongside both an "included groups" and "name" column. A short description of each of the three columns is given below:

- 1. *soc label*: used as a reference to the branch name and should be used as the annotation labels.
- 2. *Included Groups*: refer to the SOC occupation codes included within each branch; each occupation in SOC is assigned a 6 digit code, with the first two digits referring to it's respective major group branch number (see *Figure* A.1).
- 3. *Name*: Formal title of branch, as assigned by SOC for this level of aggregation.

[5] `https://www.onetonline.org/crosswalk/SOC/`

soc_label	Included Groups	Name (+ "Occupations")
1	11-13	Management, Business, and Financial
2	15-19	Computer, Engineering, and Science
3	21-27	Education, Legal, Community Service, Arts, and Media
4	29	Healthcare Practitioners and Technical
5	31-39	Service
6	41	Sales and Related
7	43	Office and Administrative Support
8	45	Farming, Fishing, and Forestry
9	47	Construction and Extraction
10	49	Installation, Maintenance, and Repair
11	51	Production
12	53	Transportation and Material Moving
13	55	Military Specific

Table 3.7: Overview of SOC category names, the included SOC group numbers for each branch alongside label reference to be used for annotations.

As mentioned in the previous section, the O*Net SOC Crosswalk Quick Search should be used to aid the annotation process[6]. Each occupation in this system is assigned a 6 digit code, with an exemplar breakdown shown in *Figure* A.1. The annotator should focus only on the first two digits, representing the major group, to determine which soc label the occupation should fall into, based on it's included groups.

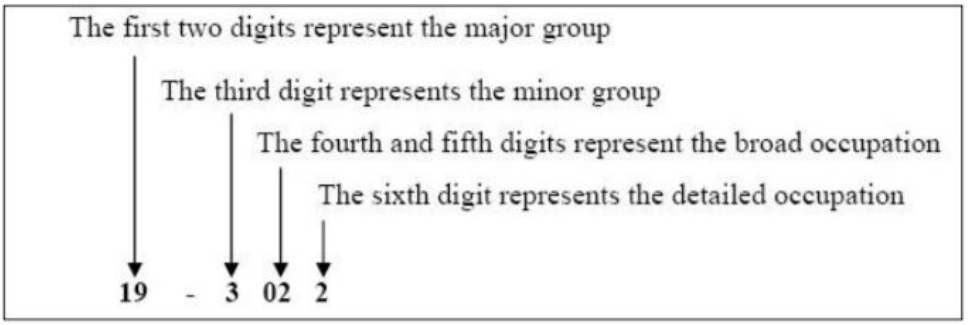


Figure 3.1: SOC Occupation Code: Structure Breakdown (of Labor Statistics)

Each of the instances for annotation should therefore be labelled with an integer between 1-13 representing the SOC *label*.

In addition to the label itself, in order to aid with the error analysis in Chapter 6 and thus measure the scale of certainty for choosing a given label, a confidence scale is also established. For each annotated item, a *confidence score* should therefore also be assigned to each instance, in the respective column. This score should correlate to an integer on the scale, ranging from 1-3, as shown in Figure A.2 below. A brief description for each label is given below, however it should be noted that full guidelines here have been purposefully omitted, to allow for a certain level of subjectivity.

[6] `https://www.onetonline.org/crosswalk/SOC/`

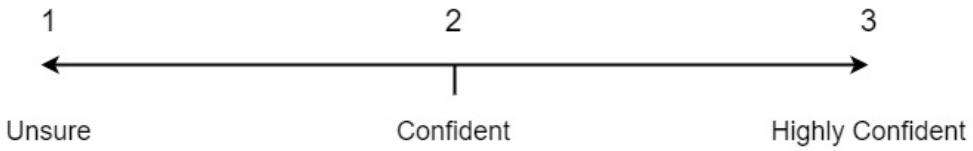


Figure 3.2: Confidence Scale

(**1**) *Unsure:* The annotator is somewhat sure however also somewhat doubtful. Both levels of confidence and doubt are equal.

(**2**) *Confident:* The annotator is somewhat doubtful, however confidence exceeds doubt.

(**3**) *Highly Confident:* The annotator is sure that the instance should be classified to the
assigned label and has no form of doubt.

Regarding the annotation tool itself, *Microsoft Excel* (xlsx) is selected for both simplicity and ease of annotations. Since the annotators in question are already familiar with the tool, it therefore requires the least amount of training in becoming familiar with it and allows the main focus to remain on the task itself.

Once these core components had been mapped out, the full annotation framework was then created to be used by both myself and an additional annotator, an experienced HR consultant, to create an unbiased and normalised process. The full annotation guideline document is illustrated in *Appendix* XX.

3.2.3 Data Preparation

In order to prepare the data for annotation, basic preprocessing is firstly carried out to ensure readability for the annotator; this is explored in more detail in *Section* 3.3.2. Following this preparation of the data, and after having tested the annotation framework with a dummy dataset, it was calculated that a single annotator would be able to annotate total of 4050 instances (*4052* after rounding to nearest divisor of 3) during the time limits of this thesis. In order to maximise the labelled data however also obtain an inter-annotator agreement score, it was then decided to randomly select an additional sub-sample from the shuffled main data to extract a further 2500 instances to be used as an overlap. This empirical choice meant that the final number of unique instances to be annotated for the classification totalled to 6,552; with exactly 2,184 instances from each of the datasets as shown in *Table* 3.8.

	#Instances
Annotator 1	4052
Annotator 2	4052
Overlap	2500
Total (Unique)	**6552**

Table 3.8: Number of Instances for Annotation

3.2.4 Intermediary Analysis

One of the main challenges during the annotation process itself when there are multiple annotators, is the level of disagreement which may come when comparing variances in annotations (Wang et al., 2017). One strategy which can be used to reduce this level is through providing rounds of *intermediary analysis*, whereby common patterns of variance, between the datasets are extracted for further discussion. For clarification, a *pattern of variance* could be where multiple instances are consistently labelled as X by one annotator and as Y by another. With the purpose of reaching a common agreement, through having these intermittent stages and discussions, it allows both annotators to understand the decision process behind certain choices and delineate any fuzzy boundaries which may lie between two given classes.

For the purpose of this research and given the size of the data, *three* key milestones were mapped out after a certain number of instances had been annotated, as shown in *Figure* 3.9.

Within each of these milestones, excluding the last, the inter-annotator agreement score is calculated using Cohen Kappa, described in more detail below, before extracting the top *four patterns of variance*. After each discussion, additional guidelines were formulated and extended to the original list to aid with the subsequent round of annotations. Some of these further guidelines are listed below:

- The *level of education*, for example a *university degree*, may affect the classification of an occupation. Some *fuzzy boundaries* which are delineated using this rule are
  - 1 vs 7
    - E.g. given an Accountant (Buchhaltungsleiter): If university degree required, the occupation is classified into *soc label 1*, else it is categorised into *soc label 7*.
  - 2 vs 9
    - E.g. given an Engineer: If university degree required, the occupation is classified into *soc label 2*, else it is categorised into *soc label 9*.
- Whether any physical contact within an occupation may affect the classification of an occupation
  - 5 vs 7
    - E.g. given a Receptionist: If the role requires more of a *front desk* position with physical contact, the occupation is classified into *soc label 5*, else it is categorised into *soc label 7*
- If an occupation seems to be split in terms of requirements across two different roles, select the occupation which fits the *majority* of the them. If the occupation is *equally* split, select the most *leading* position
  - 1 vs 5
    - E.g. given a HR assistant: If the role requires more *recruitment* and *outsourcing* skills, rather than administrative abilities, the occupation is classified into *soc label 1*, else it is categorised into *soc label 5*

- 1 vs 2
    - E.g. given an IT Assistant Manager: If the position is split and has some leading aspect however not the majority, despite "manager" being in the title, the occupation would be classified into *soc label 2*, else it would be categorised into *soc label 1*.

3.2.4.1 Inter-Annotator Agreement

In order to generate the inter-annotator agreement results, we utilise one of the most common algorithms used to calculate inter-rater reliability, formally known as *Cohen's Kappa* (McHugh, 2012). Based on the *observed* agreement and *expected* agreement, *Cohen's Kappa-Coefficient* measures the inter-annotator agreement through calculating the probability of agreeing on each individual class by chance (Brennan and Prediger, 1981). As exemplified in *Table* 3.9, through having the various milestones, discussions and confidence calculation intervals as part of the annotation process, the inter-annotator agreement score increases drastically from the first round of *0.68* to the final round at *0.84*. This final score can therefore be considered *moderate to strong* agreement (McHugh, 2012), especially given the high number of classes. It should be noted here that the number of total final annotated instances is reduced from 2500 to 2468, due to the fact that some job postings did not contain enough information about the advertised occupation and were therefore categorised as "nv", not available; these instances are dropped from the data.

Date	**Milestone**	**Cohen-Kappa Agreement**
07/05/2020	(1) 650 instances *before* Discussion	0.68
07/05/2020	(1) 650 instances *after* Discussion	0.83
14/05/2020	(2) 1300 instances *before* Discussion	0.8
14/05/2020	(2) 1300 instances *after* Discussion	0.87
28/05/2020	(3) 2500* instances *final*	**0.84**

Table 3.9: Annotation Milestone Information: Cohen-Kappa Agreement Scores. * after removing "nv" labels, a total of 2468 were left.

3.2.4.2 Final Labelled Dataset

Once the inter-annotator agreement is calculated and both of the individual datasets have been manually labelled by each given annotator, the ultimate step involves ascertaining the final labelled dataset to be used for the classification system. Although the intermediary analysis discussions and transformations aided with the process, inevitably there is still a differing overlap in some of the labels chosen by both annotators. Consequently, since I also partially annotated the data, the final labels are selected at a *random* to avoid bias; some prejudice may be unconsciously integrated through manual selection of myself. The final results are then concatenated together to form the fully labelled dataset from both annotators.

As discovered during the annotation process, there were several instances in which the vacancy descriptions contained little to no information about the given occupation. As a result, these instances which contained less than 120 characters are also removed from both the labelled and unlabelled datasets; 138 and 19,269 are dropped, respectively.

The final labelled dataset therefore consists of 6,342 instances in total. It is suggested to undertake this step before extracting a sample of the data during future works.

3.3 Preprocessing of the Data

The preparation and cleaning of textual data is a vital component in all NLP-related tasks and an integral step before applying data to any ML algorithm since it can have a direct impact on the performance of a given system. There are various approaches which can be used to preprocess raw text to remove noise and ensure machine readability however, each of these techniques are not directly transferable across tasks. For example, inevitably by cleaning the data and transforming it into a more generalized state, some of the fidelity of the data will be lost which may be integral information dependent on the task at hand (Barushka and Hajek, 2019). As a result, a careful selection of the most appropriate preprocessing steps should be made for each of the given components within this research

In order to prepare the data for the various experimental components within this research, this section describes the breakdown of the preprocessing pipeline. The first parameter taken into consideration is the fact that this research utilises *German* language data, whereby a short exploration into the linguistic differences between German and English are made, given that the abundance of NLP related data, research and computational tools are tailored to English itself. Once these foundations have been made, a description of the basic preprocessing for the annotation data preparation is undertaken. Following this, a brief description of how the data is split into *train*, *validation* and *test* subsets is made, in relation to using both the *labelled* and *unlabelled* data before finally going into detail regarding the *final preprocessing* steps. This latter part is broken-down and outlined for each of the two remaining core components within the methodology, namely the *term extraction* and the *occupation classification* systems; given the fact that these are in fact different tasks, each require different preprocessing steps.

3.3.1 Characteristics of the German Language

Since the majority of the data is in German, it is firstly important to analyse the key characteristics of the German language in itself in order to establish the primary methodology and computational tools to utilise within this research; despite the abundance of research and tools available for the English language, different computational resources specifically for German may have to be utilised as well as/instead given their linguistic differences. For example, although both English and German are Indo-European languages and share the same roots being derived from the West Germanic branch, there still remain differences regarding syntactic structures and grammatical rules. Here, an analysis into the linguistic features of the German language are made, with an indirect comparison taken into consideration with those in English. The differences which will need to be considered before designing the preprocessing approach, are listed below:

- German nouns have three genders; male vs neutral vs female. These gender markings also change when the noun becomes plural and also change dependent on the case (”der”, “die”, “das”, “den”, “dem”, “des”); 6 different forms in total

with 16 different uses across them all - in comparison to English's "the" (Articles change based on case)

- German is a highly inflectional language and therefore has more morphological inflections, primarily suffixes; as a result, German tends not to use personal pronouns
- The lexicalization of compound nouns is imminent in German; there is an abundance of *compound* words in German
- German punctuation differs from English in terms of quotation marks - these need to be included when removing punctuation.
- Commas used in place of decimals
- Word order differs greatly; German is freer than English as the dependency relates to the article of the noun rather than the word order; also differs with verb ALWAYS in second position
- Unlike English, German capitalise ALL nouns
- Reflexive verbs in German maintain reflexive pronoun where they may be dropped in English
- Separable verbs in German ("Ich kaufe im Supermarkt ein")

3.3.2 Basic Preprocessing: Annotation Data Preparation

Before we transform the text to a machine readable format, it must firstly be converted into a human readable format ready to be manually annotated; removing noise and encoding instances which may have been incorporated into the text as a result of web crawling within the data collection stage. Since the data is in German which inevitably contains non-ASCII symbols such as ä, ö, ü and ß, it is firstly UTF-8 encoded. X The following basic preprocessing steps are applied on all of the data, including both the *vacancy descriptions* alongside the *vacancy titles*.

As mentioned in the *Data Collection* section, the job posting data itself is extracted and collected through crawling multiple professional job websites. As a result of this crawling, some noisy tags are therefore added into the raw data and will need to be removed during this basic cleaning phase. Here, both HTML and JavaScript tags are stripped, before the strings are tokenized on the word level and all additional white-space characters are further removed, where two or more appear together consecutively. The reason for these steps specifically during this basic cleaning phase is inevitably to aid with the readability of the texts by the human annotators; these machine-readable tags may both distract or influence the process particularly for someone of a non-technical background.

The subsequent preprocessing stage here includes removing digits, which may refer to job numbers or salary ranges which are not relevant for this task, alongside punctuation, excluding carefully selected special characters. Both question and exclamation marks are replaced with full-stops, to still indicate a sentence ending, before nearly all remaining punctuation is stripped. A total of *4* special characters chosen to maintain within the text, including [".", ",", "#", "+"]; the reasons behind these choices are illustrated in *Table* 3.10.

Special Character	Reason
.	indicates end of a sentence; both "!" and "?" are replaced with "." for normalisation. This may be important when creating the patterns for the Term Extraction system as well as for different levels of embeddings during the Classification System.
,	may be particularly useful when creating the patterns for the Term Extraction system.
#	may be important information regarding an occupation; for example, the skill "C#"
+	may be important information regarding an occupation; for example, the skill "C++"

Table 3.10: List of special characters maintained when removing punctuation within the preprocessing stage, with their respective reason

The final stage of the basic preprocessing, includes removing and transforming abbreviations from the texts. However since this part requires external knowledge for the given domain, here we utilise two types of *gazetteers* to aid with this conversion and ultimately improve the normalisation process; these are specifically described in more detail below, in *Section* 3.3.2.1.

3.3.2.1 Gazetteers

Gazetteers are ultimately sets of lists or dictionaries consisting of terms related to a certain concept, such as locations, people or organisations, which can be used to help with both the cleaning and normalisation process within this *preprocessing* stage. Given the abundance of syntactic differences and ambiguities that exist in natural language, this direct implementation of external knowledge has demonstrated to have positively correlated effects and "a major impact on the performance of NLP applications" which apply them (Zhang and Iria, 2009). Within this research, two types of gazetteers are utilised.

The first is a *list* based resource which includes various forms of gender representative abbreviations, such as m/f/d (male/female/diverse), to be stripped from both the vacancy *titles* alongside their respective *descriptions*. Following the introduction of *anti-discrimination* law, many companies explicitly use these types of abbreviations within their corresponding job titles to emphasise the fact they do not discriminate against *any* gender, whether this be *male*, *female* or *gender neutral*. Nonetheless, for the purpose of this research, all occupations should be considered available to *all* gender types regardless of their specific markers and therefore their removal will aid in the normalisation of the occupations. However, given the abundance of their variable formulations, it can prove to be challenging when detecting all of them without external knowledge; some of these variable abbreviations here include d, for *diverse*, i for *intersexual*, a for *different* and x for *no defined gender* (kar). As a result, manual data exploration and analysis can help to ascertain improved coverage and develop a full gazetteer list for them to be stripped and thus normalise the text for our specific research.

The second type of gazetteer is a *dictionary* based resource, whereby various abbrevia-

tions or acronyms are mapped with their respective expansions. Since each additional word/character often can have a direct correlation with the price for online job postings, abbreviations are used in abundance as an alternative (kar). However, similar to the problem of variable gender abbreviations, without domain-specific knowledge and *context*, it can be difficult to computationally determine their *true* meanings. As a result, this research utilises *four* different types of abbreviation-dictionaries specifically tailored for different aspects within job posting data; *education*-related, *skill*-related *title*-related as well as a generalised *other* abbreviations.

The full lists of each of the preprocessing gazetteer utils can be found in *Appendix* B.

3.3.3 Train, Validation and Test Split

This is done before the final preprocessing, since both the Term Extraction and the Occupation Classification will require slightly different forms of input.

Whilst the entire *unlabelled* data can be referred to here as *unlabelled_train* data, since this will be used in the training phase of the occupation classification model, the *labelled* data is split into a 60, 20, 20 distribution here. The reason for this choice in split is so that we have a substantial number of *labelled* instances within the test data, yet still having enough for the system models to train and ultimately learn patterns from. Ultimately then, this provides the research with *four* data subsets, namely *labelled train*, *unlabelled train*, *validation* and finally *test*. I specifically chosen to stratify the split since some of the classes contain very few instances; this way, each data subset contains at least one instance from each of the occupational classes. The final distribution of the data is seen below in *Table* 3.11.

	#trainInstances	**#validationInstances**	**#testInstances**
Labelled	**3,697**	1,232	1,232
Unlabelled	371,796		
Total	**375,493**	**1,232**	**1,232**

Table 3.11: Number of instances within each of the datasets, split into *labelled* and *unlabelled* subsets.

3.3.4 Final Preprocessing

For the final preprocessing stage, the remaining cleaning and text transformations is undertaken to prepare the data for the models. Since ultimately this research utilises both a *term extraction* and *occupation classification* system, two different approaches are used during the latter cleaning stages to prepare the data for each. Inevitably since the data used in this research includes multiple text fields, these have also each been specifically taken into consideration through the implementation of unique preprocessing pipelines. A summary of these differences can be seen in *Table* 3.12, with a short summary of each below.

- **Umlaut Conversion:** Multiple vacancy postings contain a mixture of umlaut characters alongside their Latin encoded representations. The umlauts are thus maintained and their respective translations imputed with umlauts for the term extraction however, they are removed/converted for the classification system. The

*Preprocessing	Term Extraction		Occupation Classification	
	Seeds	**Data**	**Titles**	**Descriptions**
Umlaut Conversion	*False*	*False*	*True*	*True*
Digit Removal	*True*	*True*	*True*	*True*
URL Removal	*True*	*True*	*True*	*True*
Stopword Removal	*False*	*False*	*True*	*True*
Lemmatisation	*True*	False	*True*	*True*
Stemming	*True*	False	True	*False*
Gender Normalisation	*True*	*True*	*True*	*True*
Lowercase	*False*	*False*	*False*	*False*
Final Punctuation Removal	*False*	*False*	*True*	*True*

Table 3.12: Table illustrating differences in final preprocessing techniques for each of the systems; *False* indicates the technique isn't utilised whilst *True* indicates it is

reason for using them in the former is to avoid any mis-tagging of linguistic features when using the NLP python package, SpaCy, which could lead to misleading pattern creation. Umlauts are transformed in the latter system to their respective Latin characters (e.g. "ae") in order to normalise the corpora. In addition to this, umlaut transformation within the latter system will help to avoid any encoding issues that may arise when transforming the text into computer readable representations.

- **URL Removal:** URLs are removed for both of the systems since they do not carry the semantic meaning necessary for the task; usually these are links to the company website or webpage where the applicant can apply for the position.

- **Stopword Removal:** Stopwords are maintained for the *term extraction* however removed for the *occupation classification*; although they are considered as noise for the latter since they do not carry any real *semantic* meaning, they may prove to be important when creating *syntactic* patterns for the former. The final set of stopwords are composed from those of NLTK[7] as well as SpaCy [8] using both the *German* and *English* sets; latter used for anglicism coverage.

- **Lemmatisation:** Lemmatisation is applied to both the seed data in the term extraction alongside all of the data in the classification system to help to remove the abundance of different inflections in German and specifically here to strip gender specific endings for normalisation. Here, a German-based lemmatiser is utilised namely the *Inverse Wiktionary for Natural Language Processing* (Liebeck and Conrad, 2015)(IWNLP)[9] and is applied on *every* token. Since it requires Part-Of-Speech (POS) tags as additional input, the capitalisation of tokens is still maintained to avoid any mis-tagging.
  E.g. Input: *Entwicklerin*, Output: *Entwickler*. Lemmatisation is not applied to the job posting data in the term extraction hwoever to avoid incorrect NLP tagging of features.

[7] `https://www.nltk.org/`
[8] `https://spacy.io/`
[9] `https://github.com/Liebeck/IWNLP.Lemmatiser`

- **Stemming** Recursive Stemming using *German Snowball Stemmer* [10] is applied to the seed data for the *Term Extraction* system. By using both lemmatisation and stemming here, the most *normalised* forms of the words are ascertained which will aid in detecting the maximum number of patterns as well as the maximum number of matches possible; ultimately improving coverage. This is again applied on *every* token to ascertain each true stem. It is again not applied to the job posting data within the same system to avoid mis-tagging of linguistic features. Stemming is also applied to the occupational titles for gender normalisation in the classification system.
  E.g. Input: *Entwickler*, Output:*Entwickl*

- **Gender Normalisation:** Normalise gender word suffixes i.e. "frau"/"mann" to masculine forms. This also includes removing endings such as "innen" and "frau".
  E.g. Input: *Kauffrau*, Output: *Kaufmann*

- **Lowercase:** Capitalisation is maintained fully for the *term extraction* system to avoid NLP mis-tagging and optimal pattern learning. Although it was considered to be applied for the *occupation classification* model, capitalisation is an important aspect of the German language therefore it is retained in this model also.

- **Final Punctuation Removal:**] Remove commas for the classification system however maintain all remaining special characters; even full-stops at the end of sentences (not only within skills) will be important when generating the embeddings at different levels for the classification model.

As exemplified above, ultimately the *Term Extraction* system requires minimal preprocessing in order to capture enough *semantic* and *syntactic* information for the pattern generation and learning whilst the *Occupation Classification* requires further cleaning to remove additional noise. It should also be noted here however, that the *ontology seed lists* are also preprocessed in line with the *Term Extraction*, whereby each seed is stemmed and lemmatised in the same way; ascertaining the most generalised word forms to detect as many patterns and exact term matches as possible.

3.4 The Overall System Pipeline

This Section provides a *brief* overview of the architecture implemented in this research with regards to both the Term Extraction as well as the Occupation Classification systems. Since the results of the Term Extraction are used as a comparative aspect serving as features for the main classification model, first this is outlined. Following this, a short introduction and summary with regards to the method and structure of the Occupation Classification is given. Both systems are described in more detail in their own individual sections, Chapter 4 and Chapter 5, respectively.

In order to extract the relevant skill and education related requirements in the Term Extraction module, I implement a similar Distantly-supervised Pattern Learning methodology, as described in Chapter 2, Section 2.4. Here, the preprocessed job postings alongside the preprocessed seed lists and manually crafted patterns are used as input,

[10] `https://www.nltk.org/_modules/nltk/stem/snowball.html`

initially finding *exact* matches. As mentioned, I use two seed lists, one skill-related and another education-related. Once these matches have been found, the bootstrapped system learns the contextual patterns and features of the neighbouring words. These patterns are then applied to the text to identify new terms before the terms are thereafter scored against the terms from the original seed list. Two metrics are used to evaluate a given extracted term, namely Edit Distance for fuzzy syntactic matching alongside FastText word embeddings for semantic similarity scoring. If the terms score above a certain threshold, as discussed further in Chapter 4, the terms are labelled as either a Skill or Education, dependent on what the system is trying to extract. This system is thus implemented *twice* consecutively; once for the extraction of skill-related terms and once for the education-related terms.

Once these terms have been extracted, they used as features for their respective job postings to be inputted into the Occupation Classification model. For this classification system, the pseudo-labelling architecture is implemented, as described in Chapter 2, Section 2.1. This approach allows the partially labelled data annotated by Greple's experts to guide the process, whilst leveraging the abundance of unlabelled data which is available. This implementation is developed using two models, namely a Support Vector Machine (SVM) alongside a Convolutional Neural Network (CNN), to compare the effects of traditional Machine Learning (ML) with regards to neural networks with hidden layers, otherwise referred to as Deep Learning (DL). In addition to this, two different text representations applied to the job postings as well as the extracted requirements, namely Term Frequency Inverse Document Frequency (TF-IDF) alongside FastText. I utilise these two text representation types due to the fact that the main source language of our data is in German and thus it contains a multitude of compound words and therefore this is to determine how well the word embeddings will perform in comparison to the more traditional TF-IDF. FastText is able to capture the semantics using the sub-word level; allowing for better generalisation across these word types. Since this section only offers an overview of the systems, these embedding models are only described in full-detail in Chapter 5. Each of these models are further compared with and without the extracted terms as features.

The overall system pipeline should therefore take a preprocessed job posting as input, extract the relevant skill and education requirements before then using the extracted information to predict which occupational category the occupation advertised should fall into.

Chapter 4

Term Extraction

This chapter provides a detailed description of the Information Extraction (IE) approach used in this thesis, namely the Term Extraction model. The structure of this chapter is mapped out in five parts as outlined below.

Firstly, in Section 4.1, the two classes of terms to be extracted from the job postings are defined, namely the Skill and Education related terms. Here, a statistical overview of the distributions in terms of number of seeds are illustrated as well as the reasons for their choice in implementation.

In Section 4.2, I describe the technical details regarding the architecture of the Term Extraction model implemented, through explaining the core principles behind it; the input, method and output. For clarity, a visualisation of the pipeline is also illustrated.

Following this, Section 4.3 discusses the features which are used for both extracting and creating the patterns, whereby visualisations of the initial ontological seeds are illustrated, by looking into and comparing their respective linguistic features.

Section 4.4 explores the scoring of the candidate terms on both a syntactic alongside a semantic level.

Lastly, the results of the final extracted terms are examined in Section 4.5, whereby only an extrinsic evaluation is undertaken as a result of time-constraints within this research. In other circumstances, it would be recommended to create a test set with gold labels for a sample of the job posting data, assessing the performance of the model through using Precision and Recall as the evaluation metrics.

4.1 The Seed Data

A single job posting can contain an abundance of information about an occupation, with specific requirements relating to various aspects such as the types of *skills* or even the type and level of *education*. In fact, often it is all of these aspects combined which are integral for determining how a given occupation can be defined exactly and thus analyse market changes for given roles. In this thesis, it is hypothesised that with the inclusion of these requirements methodologically mapped out for each respective job posting, the overall performance of the occupation classification will improve. However,

given the fact that the job posting data within this research is completely raw natural language, these requirements are not defined in a systematic and structured way. As a result, using the literature described in Chapter 2 as a basis, a *Distantly Supervised Pattern Learning* architecture is thus implemented, of which only a small number of terms is required as the *input*, referred to here as *seeds* or seed data; these ultimately act as *gold* labels for the model.

For the purpose of ascertaining the requirements related to each occupation advertised in the job posting data, I look at two core classes, namely the *skills* alongside the *education* level. Skills can be defined as the expertise and specific knowledge needed to be applied to undertake a given task in a practical setting and are often the first important factors considered by both clients posting job vacancies and candidates applying to them; a good match in the required and desired skills to be used in the given advertised occupation is necessary for a good *fit* on both sides. Skills can be further divided into two categories, namely *soft* skills which are more interpersonal and *hard* skills of which relate to more hands-on, technical skills. For time purposes, both of these groups are combined into one single class though they could be later further analysed to perform a more granular analysis. Education, on the other hand, is the *process* of ascertaining given knowledge which can be used as the basis for developing certain skills. There are many different types of education which also exists, ranging from formal education with regards to school and university but also non-formal relating to more hands-on practical apprenticeships. In addition to the skills, some occupations require a a candidate to have undertaken a specific form of education; for example, in order to become a doctor it is mandatory to obtain some type of degree level and attend medical school. For this reason this Education is also selected as a requirement to extract from the text, with the aim of improving the classification system. Inevitably, education systems vary across the world with different levels and types varying from country to country. Since this research is primarily based on the German data and the German job market, only German related education types are considered for the task.

Regarding the creation of a Skill class seed list, since we already have this specific type of information contained within the company Ontology, this data can thus be extracted and used as seed information to guide the process of extracting new, relevant terms. However, since the Ontology is still under-going maintenance, it does not yet include information regarding the Education class. As a result, instead, a list of education-level related seeds are manually curated by both myself and a native German HR expert. Here, an extensively smaller list of gold seeds are created based on the Education system in German, comprising of a total of just 12 seeds. Through developing this much more limited list of seeds, it allows for the Pattern-learning system to be extensively tested and compared to using the larger list of seeds as is utilised with the Skill class. This type of seed list size is also similar to was is implemented in Gupta's Pattern Learning system Gupta and Manning (2014) as described in Chapter 2, in which a total seed list size of 4 is used as the basis. The total number of *seeds* used for both the Skill and Education class are 2,003 and 12, respectively, as illustrated in Table 4.1.

4.2 The Model: Distantly-Supervised Pattern Learning

In order to extract the competency related terms from the unlabelled data, with regards to the skills and education, a Distantly-Supervised Pattern Learning Bootstrapped ap-

Class	#Instances
Skills	1,004
Skill Synonyms	909
Total Skills	**2,003**
Total Education	**12**

Table 4.1: Number of instances within each of the class seed lists for the Term Extraction

proach is implemented, similar to which is developed by Gupta and Manning (2014) and described in Chapter 2. Here, each of the Education and Skill class seed lists alongside manually crafted patterns, as described in Section 4.3, are used as the only labelled input for identifying and extracting *similar* candidate terms within the data.

In order to do so, the system initially takes the list of seed terms as input, which are used to find *exact* matches within the entirety of the job posting corpora. Once these are detected, the model then is able to learn the contextual patterns of a given *window size* either side of the match, with regards to each of the tokens linguistic features. The window size here refers to the number of tokens either side of a matched term; if window size is defined as 1, then 1 token either side of the match is taken as the basis for learning the patterns. At this point, the manually crafted patterns can also be inputted into the system as is performed in my research, though it should be noted that this is optional. Once the system has learned the surrounding matched term patterns, these patterns are then used to search across the job posting data again, to discover *new* terms, based on whether their respective contextual patterns are matched by those learned by the system.

This process occurs iteratively for n iterations. Within each iteration, the new terms extracted by the previously learned patterns are scored semantically, with only the *higher* scoring results being added to the base dictionary. Here, if an extracted term is scored above a threshold of x, it is added to the seed term set, else it is disregarded and added to the *rejected* terms. The cosine similarity is calculated between each single extracted term and each of the terms within the seed list. It should also be noted here that the new extracted terms are *only* measured against the *original* seed list and not the newly generated one. The reason for this is to avoid the semantic drift as described in Chapter 2, which could lead to extracting terms which are far away from the original seeds; this appeared to be one of the challenges in previous similar implementations. For my research, the model will continue to learn the contextual patterns and extract new terms for a maximum total of *10* iterations before the system stops. In addition to this, as an extra parameter, I have also decided to allow the model to stop if the number of accepted terms falls below 20. The reason for both of these choices is based on the aim to further avoid the semantic drift, ultimately preventing the model from extracting noisy data and trying to ascertain more streamlined and filtered final results.

Unlike previous literature, the patterns themselves are not explicitly scored in this research. Instead, in order to try and streamline the pattern learning component of the system, the model implemented here learns the linguistic features of not only the *contextual* patterns but also the *matching* seed patterns. For example, given a window size of i of a match, the system will be able to learn the linguistic information of each token within the window size as well as the linguistic attributes of the match term

itself. As mentioned in Chapter 3 with regards to the cleaning of the the data, minimal preprocessing is undertaken. This thus allows the system to learn important information which may be relevant of terms in a given class including their capitalisation, dependencies and other linguistic details that could help the model to achieve a better understanding of the data. Regarding the contextual patterns to be learned, it should also be noted here that the system will only learn the features of the tokens within a window size given the tokens are in the same sentence; if the window size spreads across multiple sentences, it is cut short.

Ultimately, the system will thus be run individually for both the Skill and Education class; this system only focuses on extracting one unique class of terms each time. The specific features to be learned within these contextual patterns are explored in the subsequent section, Section 4.3 whilst the final window size for each of the class systems are defined in Section 4.5. The overall pipeline of the architecture is illustrated below in Figure 2.4.

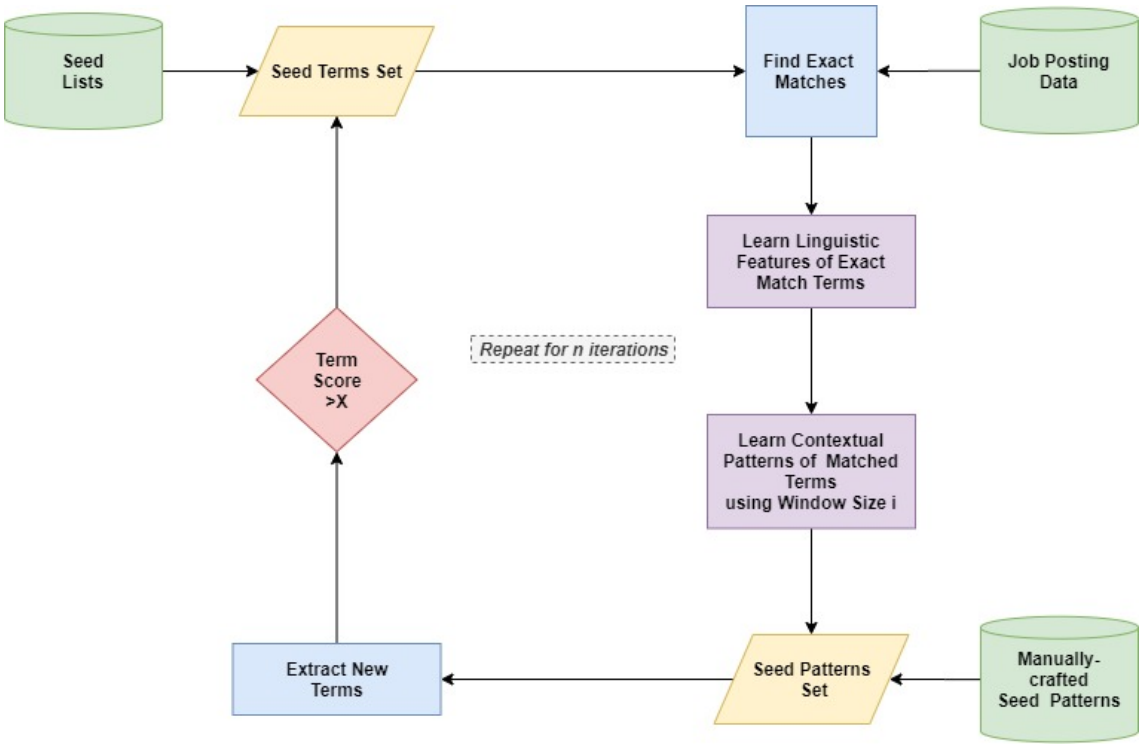


Figure 4.1: Distantly-supervised Bootstrap Term Extraction Pipeline

4.3 Features

Feature selection is one of the most integral components in nearly all Machine Learning (ML) and Natural Language Processing (NLP) tasks, having the ability to drastically improve the performance of a system through better generalisation, reducing the complexity of the data and thus improving interpretability. A feature can be defined as a specific and descriptive characteristic of data of which can be used as powerful predictive leverage within a given model by reducing the number of attributes. In order to

thus maximise the potential of this distantly supervised pattern learning architecture, this section presents the carefully selected features used for extracting the relevant terms.

Firstly, the manually crafted explicit patterns created as initial input are described, both in terms of the explicit regular expressions chosen as well as the linguistic attributes selected within them; both differing inevitably dependent on the Skill or Education class to be extracted. Following this, I explore the definite linguistic features elected to help the system create and learn the patterns surrounding the matched seeds within the main basis of the model. Unlike previous literature, not only are the features to be learned within the contextual patterns of the matched terms explored, but also the linguistic attributes of the matched terms themselves. Since the patterns themselves are not scored in my research as a result of time-constraints, the aim of this additional implementation is to enable the system to become more streamlined and improve understanding of what exact features both skill and education related terms can be comprised of.

4.3.1 Manual Pattern Creation

In addition to the seed lists, manually crafted patterns are also created for each of the Skill and Education classes, to be used as further input within the respective models. Using the high performing results from previous literature as a basis, the aim of using these patterns as additional *gold* data is to inevitably try and guide the system at the very first iteration in detecting and extracting relevant terms. If the given model is unable to extract a high quality range of terms at the first iteration, it could lead to confusion in the system and thus potentially lead to learning *false* and *misleading* patterns; ultimately worsening the performance and final results.

Both of the Education and Skill class Term Extraction systems will thus take a variety of synonymous Hearst Patterns, as implemented by Aussenac-Gilles and Jacques (2006). Since these specific patterns are in the English language, they are undoubtedly translated firstly into German, before the synonymous patterns are thereafter defined. For these patterns to extract the most number of terms, a combination of "PROPN" referring to proper noun as well as "NOUN" are used, as shown in Listing 1. Here, each of the lists within the main list is representative of a single pattern, with each dictionary relating to a single token. The token "OP" is used to refer to this being an optional parameter and therefore the system can either use this as a basis, or disregard it, dependent on if there are any matches.

In addition to these Hearst Patterns, a specific list of patterns is also specifically curated for each of the individual Education and Skill classes, illustrated in Listings 2 and 3 respectively. These each were developed as a result of an extensive manual analysis into the job posting data itself.

4.3.2 Linguistic Features

The linguistic tagging of items within the text being mandatory for both learning the contextual patterns as well as the matched term features. In order to both dertemine and learn these linguistic attributes of both the terms and the surrounding patterns, the NLP python package SpaCy is used as the core guidance. Here, it is extremely

```
hearst_patterns = [
                    [{"LOWER": "beispielweise"},
                     {"POS": "NOUN"},
                     {"LOWER": "wie"},
                     {"POS": "PROPN"}],

                     [{"POS": "NOUN"},
                     {"LOWER": "einschließlich"},
                     {"POS": "NOUN"}],

                     [{'POS':'PROPN'},
                     {'LOWER': 'wie'},
                     {'POS': 'NOUN'}],

                     [{'POS':'PROPN'},
                     {'LOWER': 'zum'},
                     {'LOWER': 'Beispiel'},
                     {'POS': 'NOUN'}],

                     [{'DEP':'amod', 'OP':"?"},
                      {'POS':'PROPN'},
                      {'LOWER': 'und', 'OP':"?"},
                      {'LOWER': 'oder', 'OP':"?"},
                      {'LOWER': 'ander'},
                      {'POS': 'NOUN'}],

                       [{'POS': 'NOUN'},
                       {'POS': 'PUNCT'},
                       {'POS': 'PROPN'},
                       {'POS': 'PUNCT',  'OP':"?"},
                       {'LOWER': 'und',  'OP':"?"},
                       {'LOWER': 'oder',  'OP':"?"},
                       {'POS': 'PROPN'}]
]
```

Listing 1: Manually Crafted Synonymous Hearst Patterns (German translated)

```
education_patterns = [
                        [{"LOWER": "Qualifikationen"},
                      {"POS": "ADP", "OP": "?"},
                      {"POS": "VERB", "OP": "?"},
                      {'POS': 'PROPN'}],

                     [{"LOWER": "Qualifikationen"},
                      {"POS": "ADP", "OP": "?"},
                      {"POS": "VERB", "OP": "?"},
                      {'POS': 'NOUN'}],

                      [{"LEMMA": "abgeschlossenen"},
                       {"POS": "NOUN"},
                       {"POS": "CCONJ", "OP": "?"},
                      {'POS': 'NOUN', "OP": "?"}],

                      [{"LEMMA": "Abschluss"},
                       {"POS": "CCONJ"},
                       {"POS": "NOUN"},
                      {'POS': 'CCONJ', "OP": "?"},
                       {'POS': 'NOUN', "OP": "?"}]
]
```

Listing 2: Manually Crafted Education Patterns

```
skill_patterns = [
                    [{"LEMMA": "Fähigkeit"},
                       {"POS": "CCONJ"},
                       {"POS": "NOUN"}],

                    [{"LEMMA": "Qualifizierung"},
                       {"POS": "ADP"},
                       {"POS": "NOUN"}],

                    [{"LOWER": "Du"},
                       {"LEMMA": "sein"},
                       {"POS": "NOUN", "OP": "?"}],

                    [{"LOWER": "Du"},
                     {"LEMMA": "bring"},
                     {"POS": "NOUN"},
                     {"LOWER": "sowie", "OP": "?"},
                     {"POS": "NOUN", "OP": "?"}],

                    [{"LOWER": "sicher"},
                     {"LOWER": "umgang"},
                     {"POS": "ADP"},
                     {"POS": "DET", "OP": "?"},
                     {"POS": "PROPN ", "OP": "?"},
                     {"POS": "NOUN", "OP": "?"}]
]
```

Listing 3: Manually Crafted Skill Patterns

important to ascertain the correct NLP tags since an incorrect tagging could lead to falsely learnt patterns and ultimately extract irrelevant terms. As described in the previous chapter, the data is only preprocessed at a basic level to avoid false NLP tagging of the linguistic items tokenized within the text. Umlauts are thus maintained, stopwords are included, capitalisation remains and even the lemmatisation and stemming is not undertaken on the job posting data; though these latter two steps are performed on the seed data.

In order to test and decipher exactly which linguistic features to implement as part of both the matching term attributes as well as the contextual pattern features, extensive testing was performed based on a small sub-sample of the training job posting data and seeds. For example, investigating to what extent features such Part-of-Speech (POS), Capitalisation, Dependency relations (DEP) information and even with regards to the Length of the terms themselves. Despite the fact that it was hypothesised that the dependencies could help the performance of obtaining high quality terms, perhaps as a result of the data being mixed in with English, the model appeared to generate more misleading results when this was implemented as a feature of the *matching token* during testing. POS and capitalisation, on the other hand, did however show to have a positive impact on the system when used for both the contextual patterns and matching terms; they are therefore implemented. Length seemed to perform also surprisingly well, perhaps as a result of filtering short stopwords which eventually appeared to seep through during the final iterations of testing. The final selected linguistic features for generating the contextual patterns for both the surrounding tokens within the window size alongside the exact matches tokens themselves, are shown below.

- Matching Token Features:
  - **POS:** The idea here is that most of the extracted terms will be a mixture of mainly proper nouns, nouns and adjectives; relating to both the Skill and Education class.
  - **Punctuation:** This is used as a boolean feature, of which aims to capture information that may be necessary for a given skill or education type. E.g. C and C++
  - **Shape:** This single feature combines both capitalisation as well as the number of characters in a token, avoiding the extraction of stopword and abbreviations where possible. Particularly this capitalisation aspect should also help since nouns are normally capitalised in German whilst adjectives are not. For example, particularly with the Education class a lot of the terms are nouns and thus capitalised.
- Contextual Token Features:
  - **POS:** The POS here may also be indicative to creating the patterns, particularly after having observed the Hearst patterns whereby some terms are in fact surrounded by conjunctions.
  - **Punctuation:** This captures information whether the matching term is used in the middle of a given sentence in a list form surrounded by commas or even at the end of a sentence followed by a full-stop.

- **Shape:** This is useful for detecting whether any shorter and lowercase stopwords are used surrounding the matching tokens.
- **DEP:** After the fourth iteration, the system then implements the dependency feature into the contextual token features. The reason for its introduction at this stage is to attempt to filter and streamline the number of patterns extracted even more so and again ultimately avoid noisy and irrelevant information being extracted.

4.4 Term Scoring

One of the most core components within this type of distantly supervised pattern learning architecture is how the extracted terms from the learned patterns are scored exactly. Since the system should not be limited to only extracting *single* token terms, but rather *multi-token* terms where necessary, it is important for the term scoring component in this research to take this into consideration. As a result, each of the extracted terms are then embedded as *sentence* embeddings, based on the number of tokens within a given term.

Although the job posting data mainly consists of German data, as mentioned in the previous Chapters, there are still some English language instances. As a result, initially the scoring system was tested using Multi-lingual sentence embeddings taken from Laser[1]. However, these proved to underperform during testing and were unable to capture the semantics of even *similar* seeds at a sufficient level, perhaps as a result of the large number of compound words within the seed lists. Although it could be considered in future works to perform some type of compound splitting, this is not undertaken within this research. Instead, since German FastText[2] word embeddings are used as one of the main semantic text representation models within this research, these were instead selected to use as the basis for the term scoring.

In order to thus generate the embeddings for the multi-token terms at a phrase or sentence level, the vectors need to be computed in a way whereby a final vector representation can be generated that captures the syntactic and semantic properties of the term as a whole. Schwenk and Douze (2017) note that "to process sequences of words [...], word embeddings need to be *combined* into a representation of the whole sequence". For this reason, each of the tokens are firstly transformed into a FastText vector representation before *pooling* is applied (Arora et al., 2016); ultimately here this means that all of the given token vectors in a term are averaged. Although I only use German word embeddings, in future works it could be suggested to combine both English and German embeddings for the scoring through reducing their dimensionality using PCA (Speer and Lowry-Duda, 2017). Again, due to time-constraints, this is not undertaken in my research.

Each of the vectorised candidate terms are then compared with each of the terms from the *original* seed list by computing the cosine similarity. To determine what type of similarity threshold to use, a sample of the seeds from each of the lists were selected and tested against each other. A score of 0.7 was firstly initialised, on the basis of the similarity between "Mann" (man) and "Frau" (woman). Since both the

[1] `https://github.com/facebookresearch/LASER`
[2] `https://fasttext.cc/`

Education systems and Skill systems proved to have varying results, different thresholds are selected for each unique system. The final threshold selected is defined in the subsequent section, Section 4.5.

4.5 Results and Discussion

This section presents the final results from the pattern learning term extraction system for both the skill and education based models. Since we do not have any hard gold labels which can be used to evaluate the system, instead an in-depth extrinsic evaluation is undertake on both sets of results. Firstly, I analyse the results obtained using the education related seeds, describing the parameters selected for the final model and the reasons behind their selection. The subsequent section repeats this analysis however based on the results obtained for extracting the Skill related terms. In order to perform this type of evaluation, the extracted terms are examined through a combination of manual exploration alongside statistical measures presented alongside illustrative visualisations for clarity. It could be suggested to perform topic modelling for further analyses and investigate how *similar* certain terms are grouped together using the FastText embeddings however as a result of time-constraints, this is not performed in this specific research. Although both of the final systems utilise a window size of 2 tokens either side of the matched term, the semantic similarity thresholds set are different; these are explored in each of the respective subsections. Different window sizes were tested for both of the systems, ranging from 1-4. Window sizes of 4 and 3 demonstrated to not extract many terms or at least when it did these were mostly noisy and single terms contained additional prepositions making them multi-tokens that did not seem semantically fitting. On the other hand, despite a window size of 1 extracting a vast number of terms, these tended to lean more towards just relevant frequent terms in general, rather than being Skill specific. A window size of 2 proved to generate the highest quality and most representative terms during testing and was thus selected. The results from both of the systems are post-processed in the same means in which the job descriptions for the classification system are undertaken, whereby the terms are lemmatised, umlauts converted for normalisation and any stopwords present are stripped. These final cleaned terms will be used as features within the classification system, as described in Chapter 5. The full extrinsic evaluation of the extracted terms and their impact on the main task of this thesis, namely the Occupation Classification, is undertaken in Chapter 6.

4.5.1 Education Level

Initially, the first threshold of 0.55 was set which extracted a total of 54 new terms. Although from my first observation this deemed to be a sufficient number of extracted terms to be expected given the specificity of the class and the fact that the seed terms are based on the German education system, some terms such as “Master” referring to a Master degree, were not recognised. This was particularly surprising given that other university degree related terms such as “Bachelor” were in fact extracted. In order to determine whether this was as a result of insufficient pattern learning or the scoring system, further analysis and testing was undertaken. This proved that when embedding this particular rejected term and semantically comparing it to the original seed list, it did not surpass the threshold of 0.55. It can be assumed that

the reason for this peculiarity is as a result of the ambiguities that exist in natural language, particularly regarding Polysemy, which can be defined as "as the form of ambiguity where two [or more] related senses are associated with the same word" (Gries, 2019). This therefore suggests that perhaps the term was transformed into a embedding representation based on the anglicism salutation for a man, "Master". Since it was expected that some occupational groups would require more university based education types whilst others more vocational, it seemed necessary to adjust the threshold so that terms such as "Master" were in fact accepted. As a result, the final scoring threshold selected is instead **0.5**, which thus allows the term "Master" to be included within the final results.

Once the terms are extracted, they are then all post-processed in the same means in which the job posting classification data is preprocessed; this involved lemmatizing the terms and ensuring not any stopwords were included. From initialising the extraction using just *12 seeds*, *6 synonymous Hearst patterns* and *4 class specific manually crafted patterns*, a total of *134 terms* are extracted across the entirety of the data, as shown in Table 4.2. Given the specificity of the Education class type, with the number of different types of education realistically available, it can be concluded that this is a good estimate of number of terms to be extracted. In order to analyse the extracted terms further however, the extracted terms are clustered together based on their respective occupational classes using the labelled data available. Inevitably, some non-related *education* noisy terms such as "Arbeit" (work) and "Bewerbung" (advertisement) were also extracted therefore these are added to a list of manually curated stopwords to be stripped. It should be noted here that to evaluate the final results without biasing the classification task, only the terms ascertained within the *training* and *validation* data subsets combined are analysed. The results for the top 30 education related terms both *exact* matching to the seeds alongside those *extracted* within these combined subsets are illustrated in Figures 4.2 and 4.3, respectively.

#Iteration	#Input Patterns	#Input Seeds	#Learned Patterns	#Extracted Terms	#Rejected Terms	#Accepted Terms
0	10	12	2,984	4,587	4,493	94
1	-	94	14,835	6,447	6,410	37
2	-	37	317	106	102	4
Total					11,005	**135**

Table 4.2: Results from Term Extraction System at each Iteration for the Education Class using 0.5 Semantic Similarity Scoring Threshold

Figure 4.3 illustrates that the pattern learning system has been able to extract a refined nuumber of fairly high quality education related terms, including terms such as "Lehrstelle" (apprenticeship), "Master" and even "Weiterbildung" (further education). Perhaps given the similar roots of German and English both being Germanic languages and therefore in some cases having similar syntax, the pattern learning system is able to extract the term "University" as well, despite the fact that the original seed list only included German terms. Nonetheless, even after manual filtering there are still some anomalies within the results including terms such as "Beschaeftigung" (employment), "Rekrutierung" (recruitment) and "Praktikum" (internship) which could be more interpreted as frequent *key* terms just not necessarily education related.

In order further analyse the terms extracted within the training and validation subsets and ultimately visualise to what extent they may prove as indicate features for the occupation classification task, they are clustered together based on the SOC label. Since the aim of the Term Extraction system is to extract features to be used for the occupation classification, here both the *exact* alongside the *extracted* terms are combined to ascertain a better coverage and understanding of each of the occupational groups. Terms which occurred most frequently in *all* of the occupational such as "Ausbildung" (vocational training), "Studium" (studies) and "Weiterbildung" (further education) are removed at this point in order to analyse the granularity of the most frequent terms *specific* to each occupational group. Terms such as "Lehre", "Berufslehre" and "Lehrstelle", each referring to a type of *apprenticeship* level of education requirement are grouped together as "Lehrstelle" (apprenticeship). Figure 4.4 illustrates the final results of the education term extraction system after post-processing, showing the top 5 education-related terms within each of the occupational categories; including both exact matches alongside extracted terms.

We can see in the doughnut chart visualisations in Figure 4.4 how different types of education types and levels are in fact to the job posting data utilised in this research. Here, the extracted term "Lehrstelle" (apprenticeship) proves to be most common amongst occupational groups 9, 10, 11 and 12 whilst more university related degree requirements, such as "Diploma", "Bachelor" and "Master" are more necessary for groups 1,2,3 and 4. Although initially it would seem perhaps some what surprising to some that the one of the most required education levels for Military specific occupations is a Diploma, these occupations could be related to the engineers within the field, such as combat engineers whereby a university degree is required. This type of cluster analysis demonstrates not only how this type of analysis can be extremely useful when aggregating occupations based on their groups, but also how the importance of an education-related requirement contributes to how an occupation can be defined exactly. The individual visualisations of the top 10 education related *exact* matches alongside the top 10 education related *extracted* terms can be found in Appendix C.

4.5.2 Skill

Again, initially the term scoring threshold was tested using 0.7 as a basis, however despite this appearing to perform relatively well, some terms such as "programmierung" meaning *Programming* where still rejected. As a result, this threshold was reduced and instead the final threshold selected for scoring the Skill class terms is **0.65**.

As illustrated in Table 4.3, the Skill Term Extraction system ran continuously using the learned patterns until the maximum number of 10 iterations was reached, before then breaking. Here, we can see that from starting of a sum of 2,003 seeds, a total of 17,902 terms are extracted as accepted items whilst an almost similar total of 19,136 terms are rejected. Despite having a higher threshold than what was determine for the Education extraction model, we can see that *more* terms are still scored as being semantically similar to the original terms in the seed list. Needless to say, with the abundance and variety of different skills necessary not only for a single domain, but a single occupation, it is not surprising that the number of accepted terms is far greater than those of the Education class. However, the difference in proportion between those terms accepted and rejected demonstrates to be very minimal with only 51.67% of the extracted terms being rejected and 48.33% accepted.

Since there is a greater number of terms accepted from the Skill extraction system in comparison to the Education system, a greater number of most frequent terms are used as part the analysis. Figures 4.5 and 4.6 thus illustrate the overall most frequent 50 exact terms alongside the most frequent 100 extracted terms across the entire training and validation corpora. From first observation, the system does appear to extract a large number of relevant skill related terms, ranging from "Kundenorientierung" (customer-orientated), "Fuehrungserfahrung" (leadership experience) as well as "Controlling" (Controlling). However, unlike the majority of terms extracted for the Education class, the results from the Skill Term Extraction model appear to combine a lot more noisy and less fine-grained terms, with some examples including "mit Angabe" (with indication), 'mit Schwerpunkt' (with main focus) as well as "von Kunde" (from customers); though the latter does seem to related to fit into the customer-orientated skill.

Additionally, as a result of the compound nouns within the German language, despite "Verantwortungsbewusst" (responsible) being a lemmatized seed from *Vertantwortungsbewusstsein* (a sense of responsibility), the system still also extracts the term "Vertantwortung" (responsibility), ultimately both referring to the soft skill of being responsible. This similar pattern is further seen with some of the adjectival nouns which inevitably are not lemmatized in the same means in which their respective adjective are. For example, "Teamfaehigkeit" and "Teamfaehig" both refer to the same skill of being *team-orientated* yet they are not yet clustered together as a single skill.

For a more fine-grained analysis, these different syntactic forms with similar *semantic* meaning and clustered together before computing the top 10 Skill related as a whole, for each of the occupational classes. These final visualisation of the doughnut charts can be seen in Figure 4.7. After the careful and manual filtering of some irrelevant and noisy terms, the final results combined show that there are in fact some indicative patterns with regards to different different occupations requiring a specific set of skills. For example, in Occupation SOC Label 4, the most frequent skills required for Healthcare practitioners include "Medizin" (medicine) and "Sozialkompetenz" (social competence). Additionally in Soc Label 9, we can see that skills specifically related to Construction occupations are computed as frequent skills, including as "CAD" referring to *Computer Aided Design* and "Umbau" translating to *reconstruction* and "Plannung" (planning). Other *related* words such as "Maschine" (machine) are also extracted within this same group however the challenge here lies in the fact that despite this term being *related* and thus *semantically* similar to the other terms, it is problematically not a skill in itself. Instead, it is more of just a *key word*, related to the field. This is mirrored in other occupational groups also, for example in Soc Label 5, the Service related occupations, where "auf Gastro" meaning *by Gastro* as well as in Soc Label 8, where terms such as "Pflanze" (plants) and "Blumen" (flowers) are extracted as frequent skills; it can be definitely considered as *key words* related to the respective class, however again they are not necessarily skills. Nonetheless, the final results do prove to illustrate a good mixture between both *soft* and *hard* skills.

4.6 Discussion

Despite not being able to evaluate the final results in a more traditional way through statistical metrics, the pattern-learning bootstrap system implemented in this research

#Iteration	#Input Patterns	#Input Seeds	#Learned Patterns	#Extracted Terms	#Rejected Terms	#Accepted Terms
0	11	1,910	12,337	12,213	8,661	3,552
1	-	3,552	60,144	8,264	5,005	3,259
2	-	3,259	12,912	2,004	795	1,209
3	-	1,209	4,165	2,802	345	2,457
4	-	2,457	11,552	1,948	307	1,641
5	-	1,641	4,429	2,985	654	2,329
6	-	2,329	5,026	1,868	643	1,225
7	-	1,225	2,532	1,672	832	840
8	-	840	1,660	1,390	671	638
9	-	638	1,604	1,026	611	415
10	-	415	923	949	612	337
Total					19,136	**17,902**

Table 4.3: Results from Term Extraction System at each Iteration for the Skill Class using 0.65 Semantic Similarity Scoring Threshold

does appear to ascertain promising results. Particularly for the Education class, we can see a variety of different related terms extracted, ranging from levels such as "Lehrstelle", meaning *apprenticeship* "Bachelor" as well as"Fortbildung" translating to *advanced training*. As described and analysed in Section 4.5.1, the clustering of these most frequent terms for each of the occupational classes illustrates how the type of education does show to vary between classes, in a similar way as what can be predicted by a human expert. For instance, with more university related education types being regarding for Healthcare Practioners and Technical Occupations whilst more practical and hands-on vocational training being required for Construction and Extraction Occupations.

Initially, it was thought that by using a larger number of seeds it would allow for greater coverage and thus extract more relevant seeds. However, as shown in the results for the Skill class, this can also lead to further noise being extracted, particularly in comparison to the results from the Education class. This difference in noise however could also be as a result of the class definitions themselves, with there being a lot more variance with regards to what can be considered as a *skill* and the number of different *education* types that exist in Germany alone. Even from visualising the most frequent *seed terms* within the Skill corpora, it is evident that the ontology is still undergoing maintenance at the time of this thesis. In fact, there are still some ambiguous and potentially noisy terms even in this *gold* standard seed list, such as "Berufserharung" which translates literally to *work experience*; it is questionable whether this can be considered as a skill as such and therefore in the final filtering, even this gold seed is removed. Furthermore, there are even some Skill seeds of which overlap with with the seeds from the Education class. Examples of such *gold* seeds from the ontology include "Abschluss einer berufsbildenden mittleren Schule" (Completion of a vocational secondary school) as well as "Google-Zertifikate" (Google Certificate). Arguably these former examples could have been classes as skills had they included exactly what *field* of education they are referring to, similar to how "Hundetrainingsausbildung" (dog training education) clearly refers to completing education of specifically training dogs. However, as in the case in the exemplar instances, this type of information is not included. Preceding on from the apparent overlap between the two seed classes, there are also other *gold* skill seeds which

appear misleading and ambiguous, without the additional meta-knowledge contained within the ontology. For instance, although it may be unknown to even the human eye how the seed "Internationales Zertifikat der Universität von Perugia" meaning *International Certificate of the University of Perugia* relates to Skill, with the external meta knowledge we learn from the description that this in fact refers to obtaining a certificate for proficiency in the Italian language.

Given the fact that the extracted terms are scored based on semantic similarity scores to the original seeds themselves, it can be concluded that if seed lists are to be used as the core gold data within this type of pattern learning system, less ambiguous and more refined seeds should be selected. The *quality* of manually cultivated, or at least clarified, seeds seems to have a greater impact in increasing the performance of this system over the *quantity* in the number of seeds for our data, as seen in the comparison between the results for the Education and Skill class. Nonetheless, this is not to say that this may differ for the extraction of a different type of requirement class or through the utilisation of a different dataset. It can be speculated however, that if the system were to be able to incorporate the meta-data from the ontology or some type of world knowledge, this could lead to the system having a greater understanding of the semantics of the seeds when scoring them and potentially meaning an improved and more robust system.

Regarding the patterns themselves, from a manual analysis into the data we can see that an abundance of more diverse patterns are learnt by the system for the skill data, although it can be suspected this is also related to the number of seeds also extracted each time, with the Skill class extracting substantially more at each iteration. For future works here, it would be suggested to implement a *pattern* scoring system, similar to what is undertaken in similar work by Gupta (2015), to further improve the robustness of the pattern generation itself.

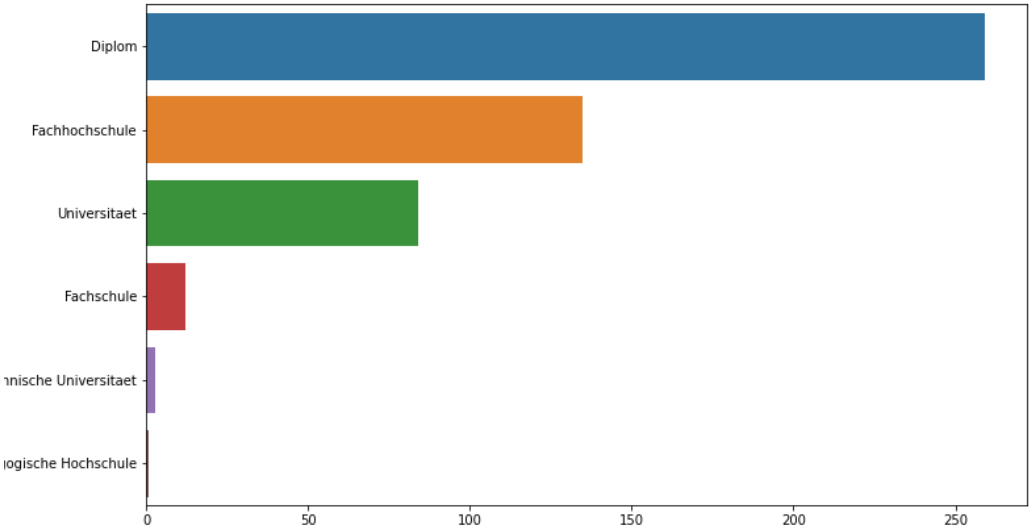


Figure 4.2: Most Frequent *Exact Seed* Education Terms across the entire Corpora (Train + Validation)

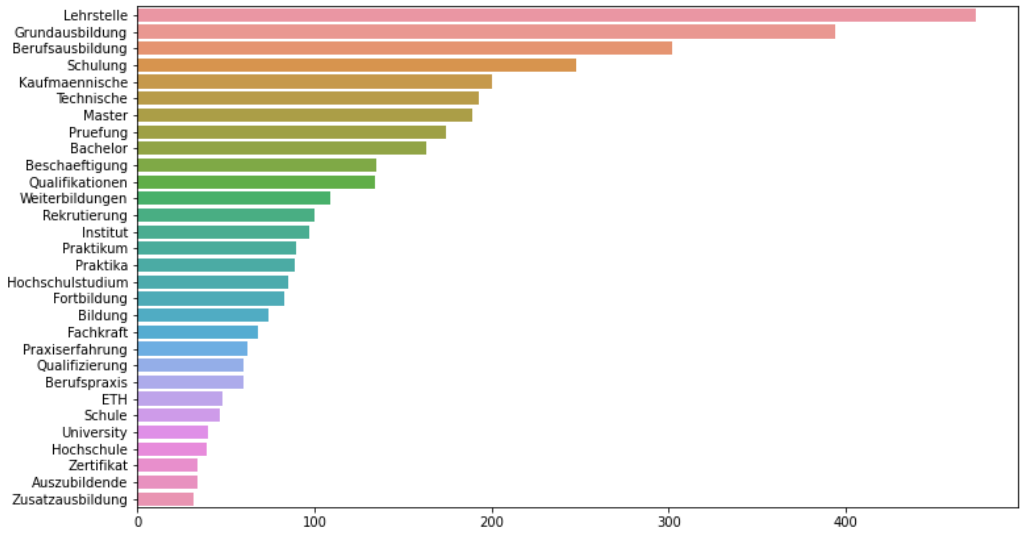
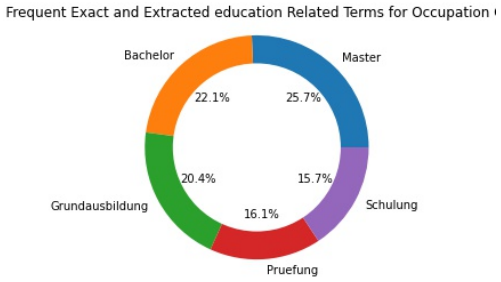
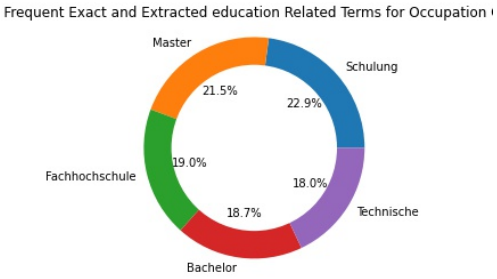


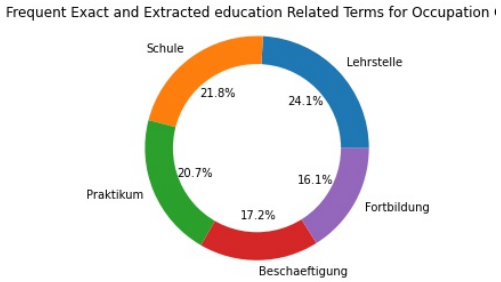
Figure 4.3: Most Frequent 30 *Extracted* Education Terms across the entire Corpora (Train + Validation)



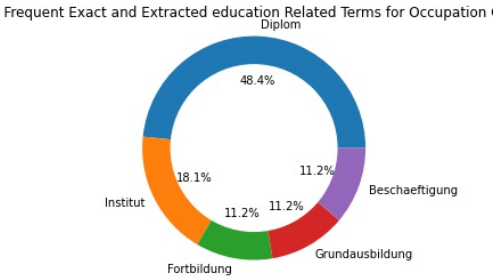
(1) Management, Business and Financial



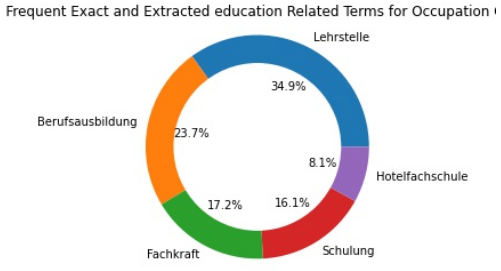
(2) Computer, Engineering, and Science



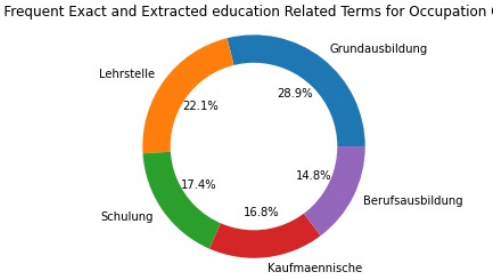
(3) Education, Legal, Community Service, Arts, and Media



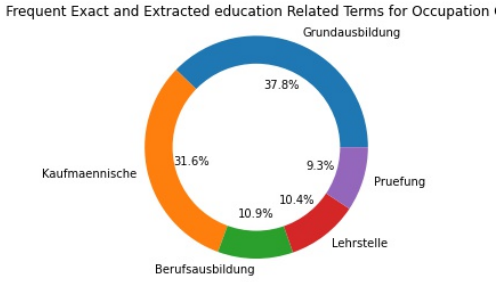
(4) Healthcare Practitioners and Technical



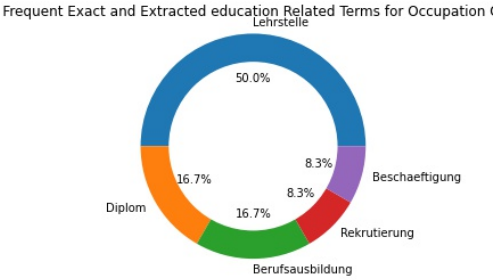
(5) Service



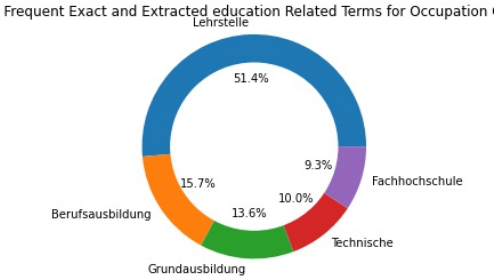
(6) Sales and Related



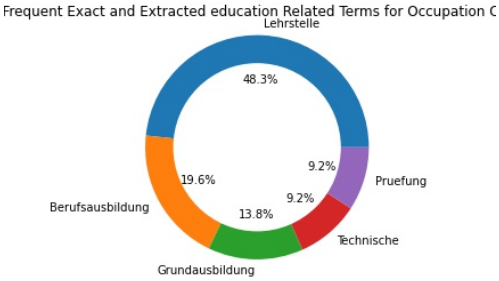
(7) Office and Administrative Support



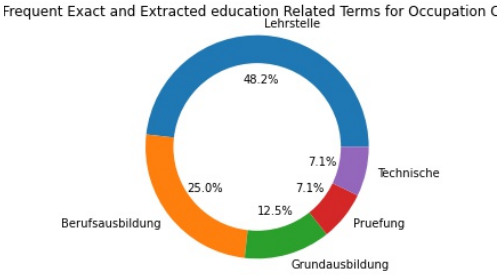
(8) Farming, Fishing, and Forestry



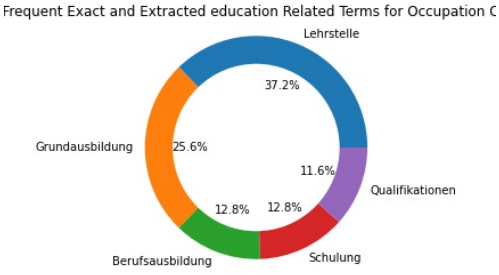
(9) Construction and Extraction



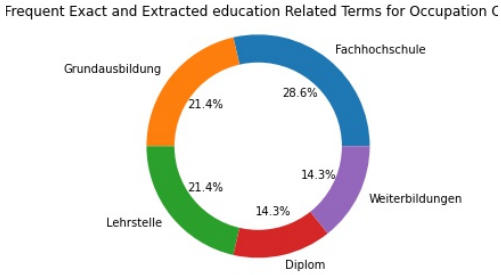
(10) Installation, Maintenance and Repair



(11) Production



(12) Transportation and Material Moving



(13) Military Specific

Figure 4.4: Doughnut Chart Visualisations illustrating Top 5 *Education* Terms within each of the Occupational Classes

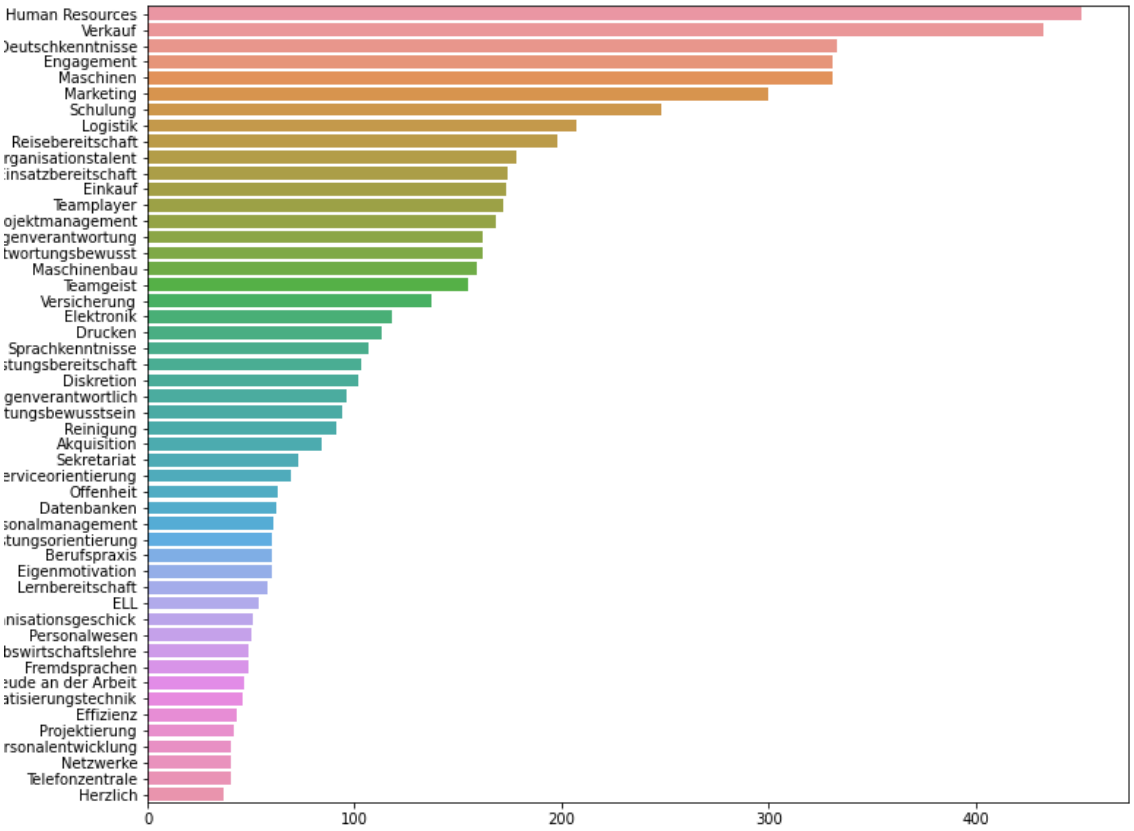


Figure 4.5: Most Frequent 50 *Exact Seed* Skill Terms across the entire Corpora (Train + Validation)

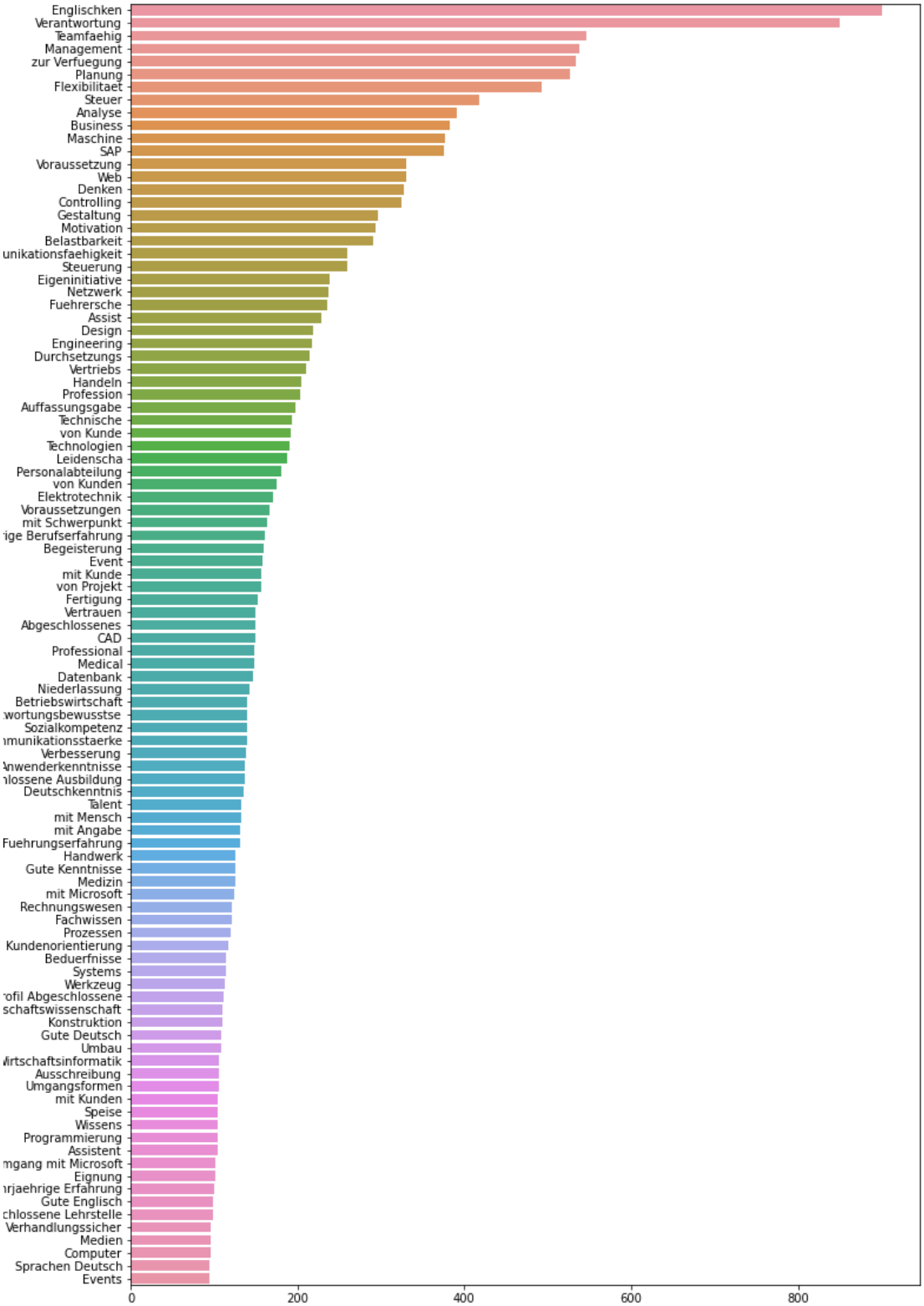
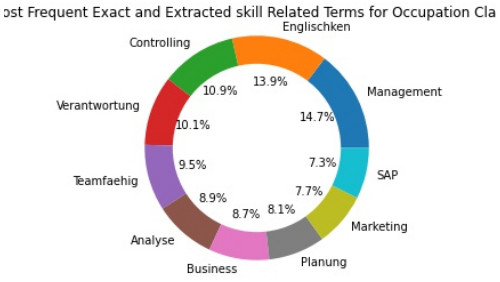
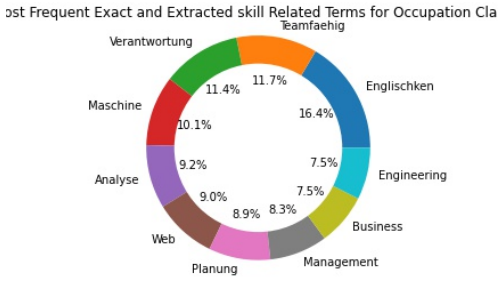


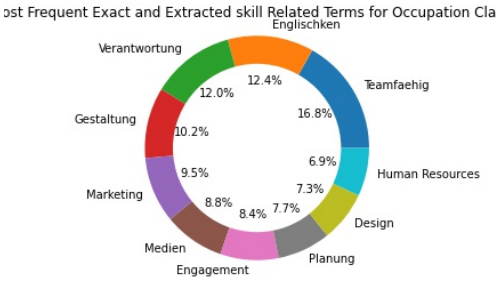
Figure 4.6: Most Frequent 100 *Extracted* Skill Terms across the entire Corpora (Train + Validation)



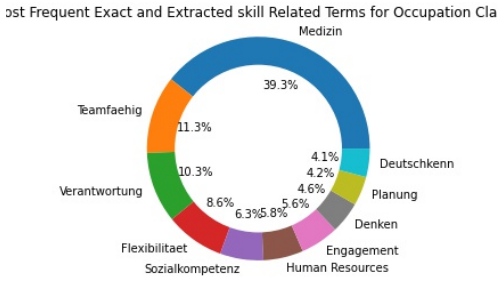
(1) Management, Business and Financial



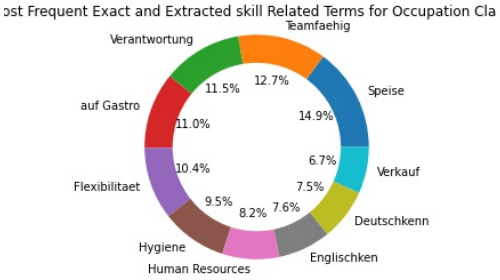
(2) Computer, Engineering, and Science



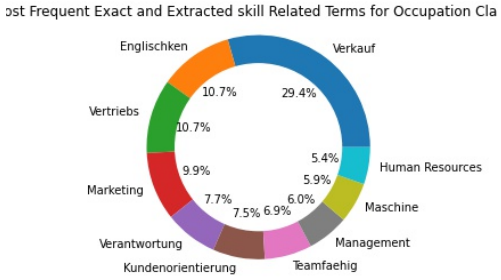
(3) Education, Legal, Community Service, Arts, and Media



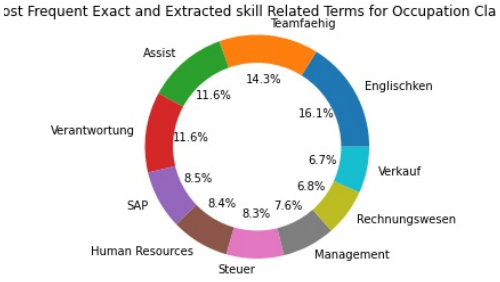
(4) Healthcare Practitioners and Technical



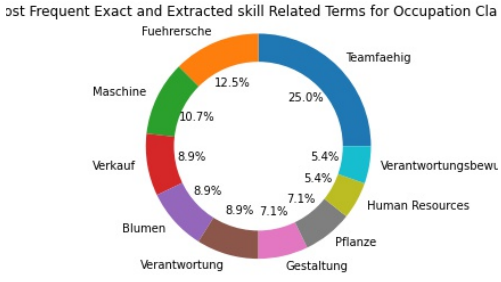
(5) Service



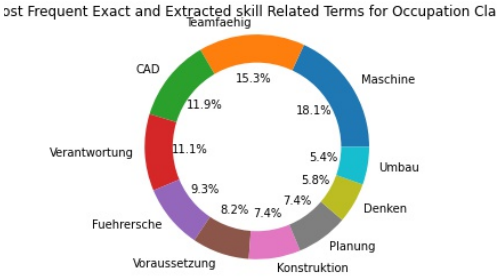
(6) Sales and Related



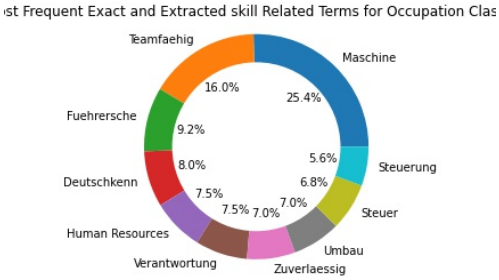
(7) Office and Administrative Support



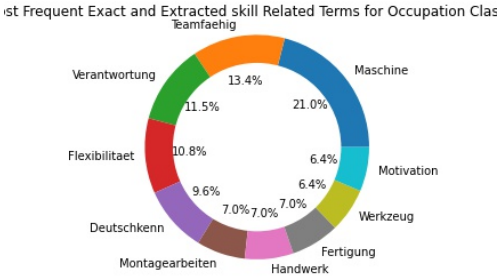
(8) Farming, Fishing, and Forestry



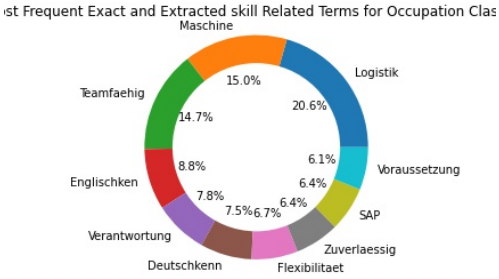
(9) Construction and Extraction



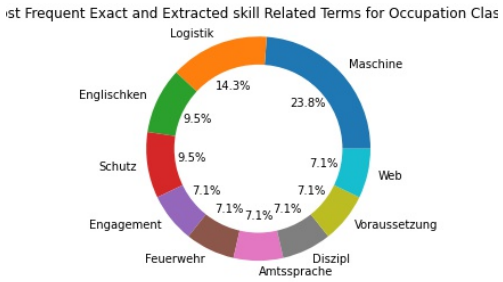
(10) Installation, Maintenance and Repair



(11) Production



(12) Transportation and Material Moving



(13) Military Specific

Figure 4.7: Doughnut Chart Visualisations illustrating Top 10 *Skill* Terms within each of the Occupational Classes

Chapter 5

Occupation Classification

This chapter provides the in-depth technical detail regarding the main component within this research, the Occupation Classification model.

First, in Section 5.1, a statistical overview in terms of the distribution in number of instances per occupational class within the *labelled* training, validation and test subsets is described, accompanied by their distributional percentages in relation to the size of the data as a whole.

Following this, in Section 5.2, I describe the architecture of the pseudo-labelling classification model itself, offering a deeper insight into the technical choices made during its implementation. For clarity of the model, this section is accompanied by both an explicit example of a data instance as input along with a visual representation of the pipeline itself.

Section 5.3 subsequently provides the key steps and illustrations necessary for undertaking a detailed exploratory analysis (EDA) into the data. Here, Rather than *describing* the job posting data as is already undertaken in Chapter 3, this Section provides an in-depth examination into the manually labelled instances through a combination of statistical counts of the data distribution alongside illustrative visualisations for each of the classes within the *labelled training* data. This type of detailed analysis thus provides the fundamental foundations needed for determining which *features* should be incorporated into the final classification model. These illustrations and analyses will also serve as a comparative basis with the final results in Chapter 6. The final features selected alongside the comparative text representation embedding models are also listed at the end of this section.

In Section 5.4, I subsequently outline the two main classification models to be implemented into the pseudo-labelling architecture described in Section 5.2, namely a Support Vector Machine (SVM) as well as a Convolutional Neural Network (CNN).

Finally, in Section 5.5, the experimental setup is described through providing more information into the parameter tuning of the final models. The related results are revealed in Chapter 6.

5.1 The Data

After having ascertained the final labelled dataset as described in Chapter 3, Section 3.2, it is important to understand the exact distribution of the data not only across data subsets, but also across classes.

In Table 5.1, both the number of instances alongside a percentage is shown for each of the individual classes within the labelled train, validation and test subsets. The percentage is calculated in relation to the number of instances in the single subset, in order to demonstrate the distribution in number of instances per class. As shown in Table 5.1, there is a large skew across each of the subsets, with a difference of 584 instances between the largest class, 2, containing 595 instances and the smallest class, 13, consisting of a mere 11 instances, in the training data alone. To emphasises this skew, Figure 5.1 illustrates a bar plot of the number of instances for each class within each subset.

As a result of stratification during the splitting of the data, this similar skew in distribution across the classes into consideration is mirrored across each of the subsets with the validation and test subsets having a similar ratio of instances for each of the categories; as seen in Table 5.1 and visualised explicitly in Figure 5.1. Despite the imbalance in classes, through stratifying the split it should prevent a given model will from drastically changing in terms of its performance as it could do if trained on different data distributions regarding the validation and test subsets.

Although it would be suggested here to perform some type of *Upsampling*, in order to try and balance out the minority class proportions, as a result of time-restrictions it is not possible to carry this out. Nonetheless, after ascertaining the model results and thereafter performing the error analysis in Chapter 6, this information may prove integral and therefore should be taken into consideration throughout the experiment.

Following this exploration into the distribution of the data as a whole, an in-depth analysis into the data is undertake order to determine which features to incorporate into the model, as discussed in the Section 5.3.

5.2 System Architecture: Pseudo-Labelling

In this section, I describe the architecture of the pseudo-labelling system implemented within this thesis, using the works described in Chapter 2, Section 2.1.2.1 as the foundations. Although there are two types of variations which can be implemented in this type of pseudo-labelling classification, with one involving re-training the same model each time whilst the other involves training a *new* model, for the purpose of this thesis only the latter is implemented. The reason for this is as a result of computational limitations and lack of time to use both types as a further comparative feature within the research; this could be incorporated in future works nonetheless.

Using the previously extracted terms related to the given occupations, we therefore come to the primary task of this thesis at hand, namely classifying each given job postings into a single occupational class relative to the occupation being advertised. The manually annotated *gold* data is used as leverage in categorising the unknown, unlabelled occupations, by iteratively training a model and generating confident predictions on the unlabelled data. In order to do so, the original gold labelled data is used as the

SOC Label	#Train	#Validation	#Test	%DS
1	553	183	184	14.94
2	595	194	198	16.03
3	144	50	49	3.95
4	359	120	120	9.73
5	398	112	127	10.35
6	253	88	85	6.92
7	481	166	162	13.14
8	20	11	8	0.63
9	274	93	92	7.45
10	364	126	122	9.94
11	69	33	25	2.06
12	171	48	55	4.45
13	11	5	4	0.32
TOTAL	**3,693**	**1,232**	**1,232**	100

Table 5.1: Number of Instances and Distributional Percentage for each class within each of the labelled data subsets

basis for generating predictions on the entire unlabelled training data, in the same way in which a supervised classification model would be run. For the purpose of this section, I refer to the labelled training data as the pseudo-train data, the unlabelled training data as the pseudo-test data. and the outputted predictions as the pseudo-labels.

After having trained a given model and generated the predictions, these pseudo-labelled instances are then added to the pseudo-train data, thus generating a larger dataset. However, rather than taking all of the pseudo-labels as the *gold* labels, the foundation of this approach relies on using a *confidence* threshold in which only predictions that surpass this number are added to the *pseudo-train*. This process continues for either n iterations, until there are no more instances in the pseudo-train data or until the number of pseudo-train instances falls below a certain *rest* length before breaking. Subsequently, the model is evaluated using the test data and the hypothesis is that the pseudo-labelled data can help to achieve a higher performing model in relation to just using the original, limited gold labelled data. For the purpose of the model in this research, the number of iterations is set to 10 whilst the confidence level varies dependent on the model utilised; the exact confidence thresholds are noted in Section 6.6. Ultimately then, this pseudo-labelling method is "based on the manifold assumption to make predictions on the entire dataset and use these predictions to generate pseudo-labels for the unlabeled data and train a deep neural network." (Iscen et al., 2019). The overall system pipeline is illustrated below in Figure 5.2.

5.2.1 Steps

1. Train a supervised model using the limited training labelled instances available.
2. Generate predictions on the unlabelled training data
3. Calculate the confidence scores for each of the predictions and thereafter the Q3 confidence threshold to be set; the confidently scored predictions act as the *pseudo-labels*

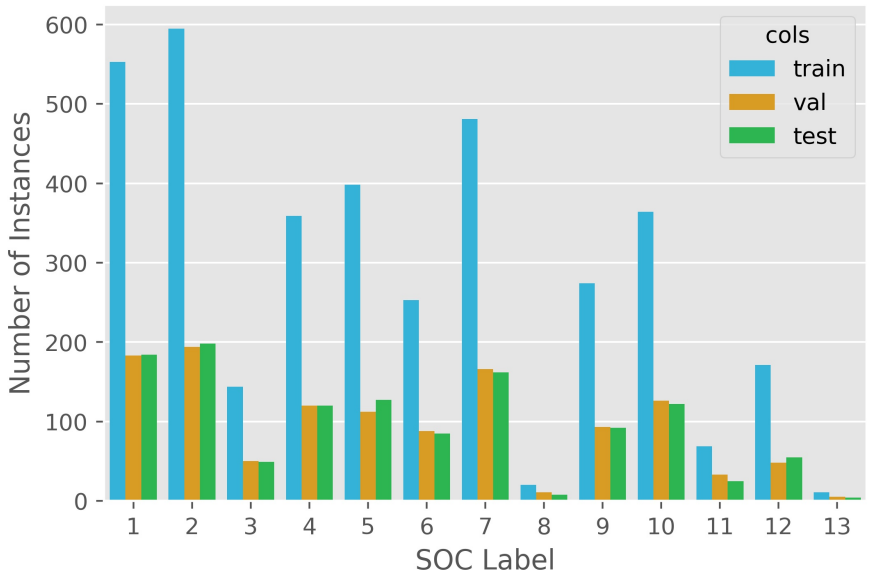


Figure 5.1: Bar Plot illustrating the Distribution of Occupational Classes across each of the labelled subsets

4. Concatenate the pseudo-labelled data with the original training data; generating a larger labelled corpus for training
5. Repeat steps 1-4 each time increasing the size of the training data until the system reaches N iterations or until the maximum rest is reached.
6. Train final model using all of the pseudo-labelled and original labelled data; evaluate using test data.

5.2.2 Confidence Intervals

One of the core components within this type of pseudo-labelling architecture that has the ability to heavily impact the performance of a model, is the type of *threshold* set. Based on the previous literature, this threshold thus is used to determine whether the pseudo-labelled instances should be added to the gold dataset or not, based on the confidence produced by the model. Confidence scores within regards to ML and classification, can be defined as a "[measurement of] the accuracy of [a] model when predicting class assignment (rather than the uncertainty inherent in the data" (Mandelbaum and Weinshall, 2017). Inevitably, if this confidence threshold is set *too* low, it will allow for more of the pseudo-labelled instances to be added to the *gold* data and the risk of incorporating *incorrectly* labelled instances to be used during training. On the other hand however, if the confidence is set *too* high, fewer instances will be added to the training data and there is a chance of pseudo-labelling hardly any instances before the architecture breaks after n iterations.

For each of the models, a float between 0 and 1 is initialised as the threshold. In the previous outlined literature, a *single* score is chosen as the confidence threshold to be used throughout each of the iterations in one given experiment. However, this raises the

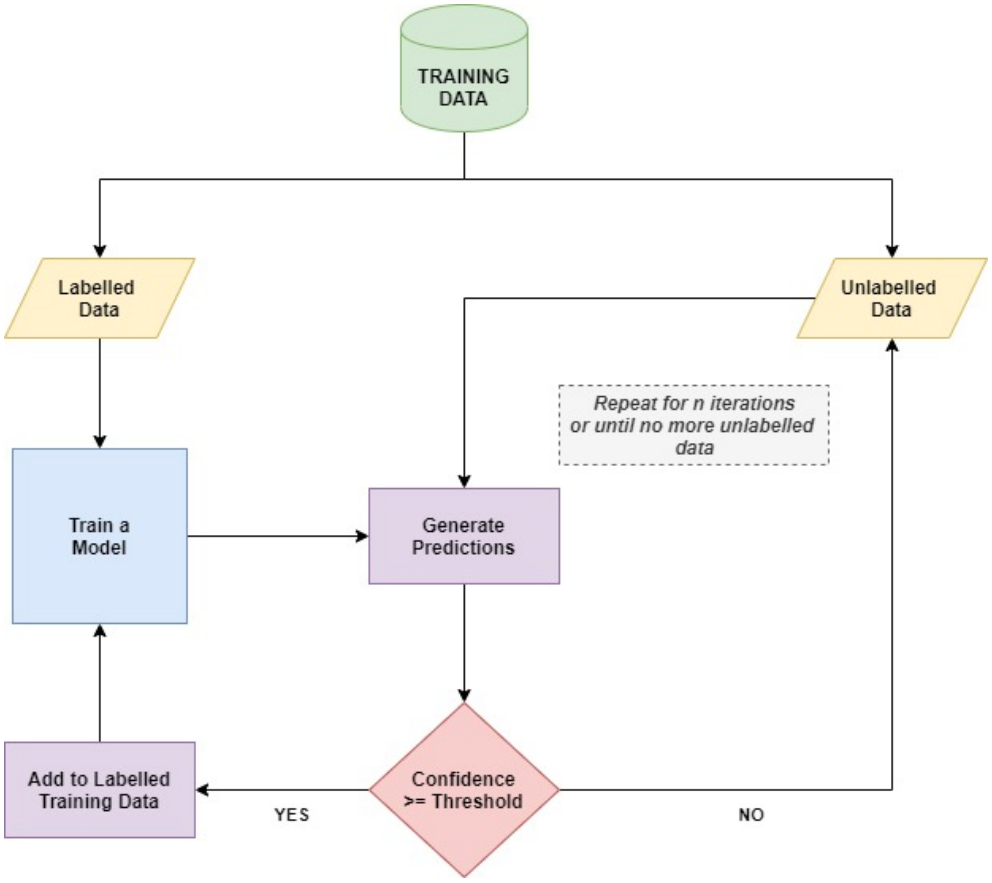


Figure 5.2: Pseudo-Labelling Classification Pipeline

question of whether the behaviour of the model changes as more labelled data is used to train it, particularly in terms of its confidence. In my research, it is hypothesised that the confidence scores will change not only across the different types of ML models implemented, but also as a result of their dependency on the number of instances that are used to train a given model. As a result, rather than using the same threshold used to initialise the model using only the limited labelled training data, the threshold is adjusted in each of the subsequent iterations as the size of the training subset increases; rather than using the same threshold throughout all of the remaining models, the system implemented in this research instead updates this confidence threshold, based on the *current* model in training.

In order to do so, I have chosen to use the Upper 75th percentile, otherwise referred to as the Upper Quartile (Q3) range. The Q3 is calculated by dividing all of the confidence scores of the *current* model into quarters before thereafter computing the median of the upper half of the data; the Q3 is thus is the value between the lower 75th and upper 25th percentile of the scores. Through selecting this interactive threshold based on the current model, it allows the model to update its parameters and weights according to the distribution of the current confidence data with the objective of improving the overall performance of the system. Furthermore, by computing the threshold automatically, less unconscious bias is also incorporated than what would have been through a manual selection.

vacancyTitle	Pflegefachperson
vacancyDescription	"Wir suchen motivierte Mitarbeiter mit entsprechender Qualifikation als Diploma Pflegefachperson DNIII. Sehr gute Deutschkenntnisse Gute MS Office Kenntnisse Zuverlaessig, flexibel und belastbar Aufgabenbereich Individuelle Pflege und Betreuung der Bewohner Interdisziplinaerer Zusammenarbeit mit anderen Pflegeteams"
extractedSkills	["DNII", "Deutschkenntnisse", MS Office","Zuverlaessig", "Pflege und Betreuung"]
extractedEducation	*"Diplom"*
socLabel	4

Table 5.2: Example of Labelled Job Posting with Extracted Information

5.3 Feature Engineering

Although recently these deep neural networks in Deep Learning (DL) have gained increasing popularity for multiple NLP text classification tasks as a result of not requiring any particularly for feature engineering, often they rely on extensive amounts of labelled data in order to perform well. Where these DL models fall short, often it is the traditional Machine Learning (ML) systems which are still able to consistently achieve high results through the explicit feature generation and engineering conducted by a human analyst with *world knowledge*. Since one of the comparative aspects within this research is to compare the performance of a deep neural network model with a traditional Machine Learning (ML) system, this section provides insights into the steps undertaken during the data analysis and ultimately the feature engineering process, with the aim of determining which manually crafted features can be discovered to help improve the performance of the classification model. The exploratory data analysis (EDA) is first outlined as a whole in the first Section, 5.3.1, illustrating all analyses made regardless of the significance of their results for the task; even if the analysed feature proves to not be indicative of a class, this is still useful and integral information to consider as we deal with a unique dataset at hand. The final list of the ultimate features to be implemented in the model are shown in Section 5.3.4.

5.3.1 Exploratory Data Analysis (EDA)

In this section, an in-depth exploratory data analysis (EDA) is performed in order to identify and examine particular patterns in the data. Although this stage was partially undertaken prior to the preprocessing step itself, here I only focus on the EDA after the data has been fully cleaned. Through looking into various different statistical and linguistic patterns in the text data across the classes, the main purpose of this section is to see whether any of these analyses may be useful for feature engineering and ultimately help to improve the performance of the classification model.

Four key analyses are explored in this EDA, namely looking into the Length (average number of sentences), relative Part of Speech (POS) tags, relative Named Entities (NEs), as well as a more illustrative investigation into the most frequent terms through N-grams. All of the analyses within the data and the visualisations thereafter produced

are solely based on the labelled *training* data.

5.3.1.1 Length Analysis

The first type of analysis undertaken is a *length* analysis. Although this type of information is more indicative as a stylistic feature of the writer itself, due to its simplicity it is undertaken to see if any insights can be offered; whether a occupational class job posting tends to have a longer, or shorter, length than others at the token and sentence level. As illustrated in the violin plot in Figure 5.4, although it appears that the average number of sentences in class 13 is slightly larger in comparison to the other classes, this is not great enough to offer any real indicative information.

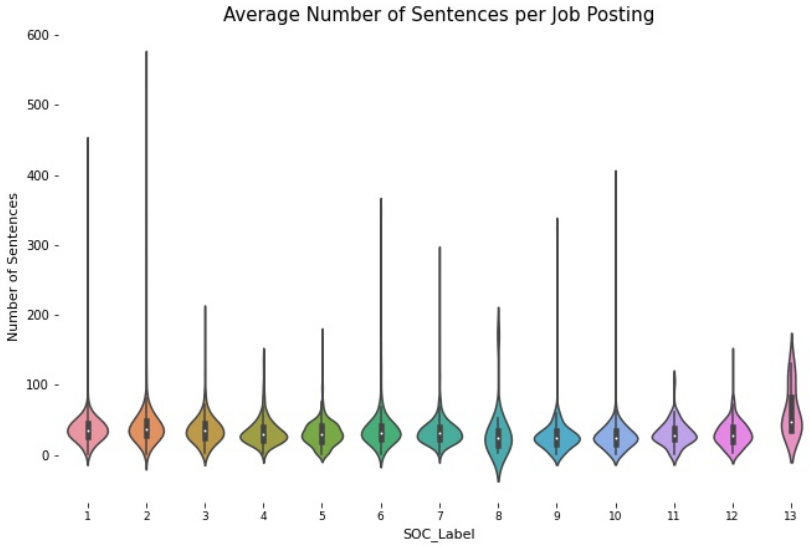


Figure 5.3: Violin Plot illustrating the Average Number of Sentences per Occupational Class

5.3.2 Linguistic Features

Here, I look into the specific linguistic features of the job postings, namely the Part of Speech (POS) tags alongside the Named Entities. The relative count for each of the occupational classes is computed and visualised for clarity.

However, as shown in Figure 5.5, here is no obvious correlation that can be seen between the classes in terms of the differing POS, with the relative number of tags across the classes all appearing to have a similar proportion to each other. As a result, there is no single POS tag which appears to be indicative of a certain class and this feature is ruled out.

In Figure 5.6 however, despite the fact that the majority of relative NE tags across the occupational classes do not show any real insights in terms of comparability, one tag which does show to have a great impact for one single class is the *Locative* (LOC) category. As shown in the figure, the LOC category appears to have a high number of location-related terms for the "Farming, Fishing, and Forestry" related Occupations

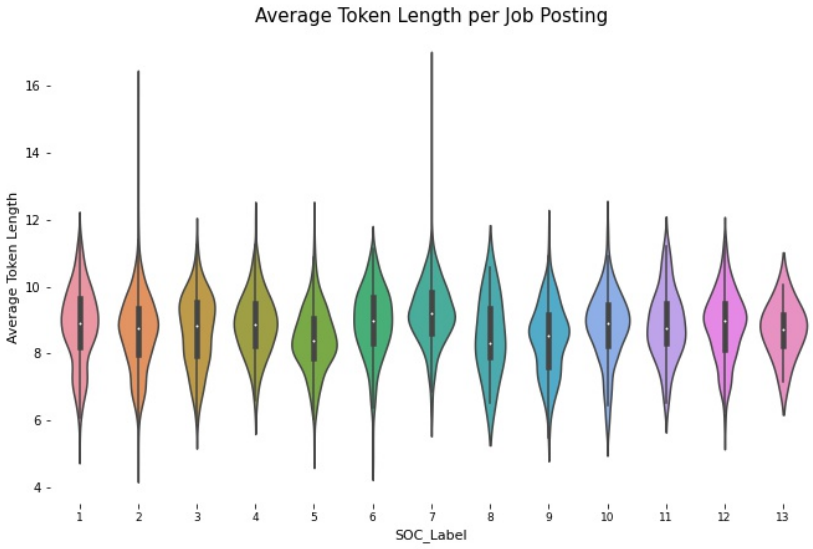


Figure 5.4: Violin Plot illustrating the Average Length of Tokens per Occupational Class

at soc label 8, which therefore suggests that it could be an indicative feature to use in helping the model classify these types of relations.

To determine the extent of how this feature could be indicative of the given class, a deeper analysis is then performed through examining the top frequent LOC terms themselves in both the the overall job posting data as a whole, alongside only in the Farming, Fishing and Forestry category. However, upon observation it is evident that the main LOC terms included are those relating to cities which are not necessarily indicative of an occupational class. In addition to this, the number of instances within this soc label only total to 20 and therefore this is probably the reason for their higher number in proportion to the other classes. For this reason, despite the falsely promising result, NE tags also are not incorporated into the final model.

5.3.3 Most Frequent N-grams

Once these statistical linguistic analyses had been undertaken, I then decided to explore to what extent the *frequency* of terms in the preprocessed textual data may be useful for this task. In order to do so, *n-grams* at the word level are computed and therefore visualised. A n-gram can be defined as a sequence of n adjacent items in a given piece of text, whereby these items can represent either characters, tokens or even sentences. For example, by looking into n-grams at a word level in which n is equal to one, each item would be taken as a single token from the sequence, otherwise referred to as *unigrams*. Using this same example, if n is equal to two, all combinations of adjacent paired tokens would be computed, referred to as *bigrams*. This latter case, alongside other variants whereby n is greater than 1, may be useful to detect collocations; multiple words which commonly appear together adjacently. N-grams have proved to be a vital feature for not just text classification tasks, but NLP challenges as a whole, primarily due to the fact that they look at series of words or characters together in order capture and explicit

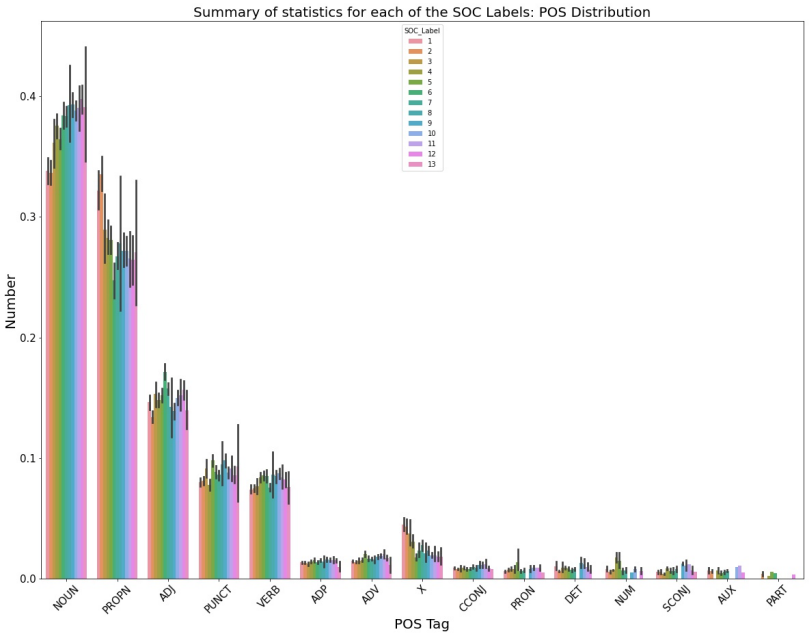


Figure 5.5: Bar Plot illustrating the Relative POS count per each Occupational Class

both syntactical and semantic information, which a single word alone is not able to do. For the purpose of this section in the EDA, I only visualise the most frequent words (MFW) for *unigrams* and *bigrams* to determine whether certain tokens are indicative of certain occupational classes.

Figure 5.7 illustrates the top 75 most frequent unigrams for each of the 13 different occupational classes. In order to ascertain these relevant terms, as undertaken in Chapter 3, English and German stopwords are stripped from the job descriptions from both NLTK and SpaCy alongside a custom set manually created through an in-depth analysis of the data. These particular visualisations of the text from the training data exemplify how each class contains its own set of semantics for each occupational group, as a result of differing language. Particularly for classes 1, 2, 4, 5, 6, 7, 8 and 13, the terms are very representative of each of the groups with key words such as "gastro" in the *Service* class and terms such as "verteidigung" meaning "defence" in the *Military Specific* occupations. It is especially important to note the fact that often the key words within each of these visualisations relate not only to the skills, but also to the education. For example, the term "diplom" referring to a phd level education appears in the *Healthcare* occupational group whilst "ausbildung", relating to more vocational training is shown in the *Office and Administrative* specific occupations.

After having generated the visualisations for the unigrams, the most frequent collocations within each of the individual classes were further extracted through the computation of bigrams. Although it is evident that there are clear patterns across the classes using the MFW unigrams, as shown above, inevitably not all information can be captured by only using the tokens individually. For example, although the words

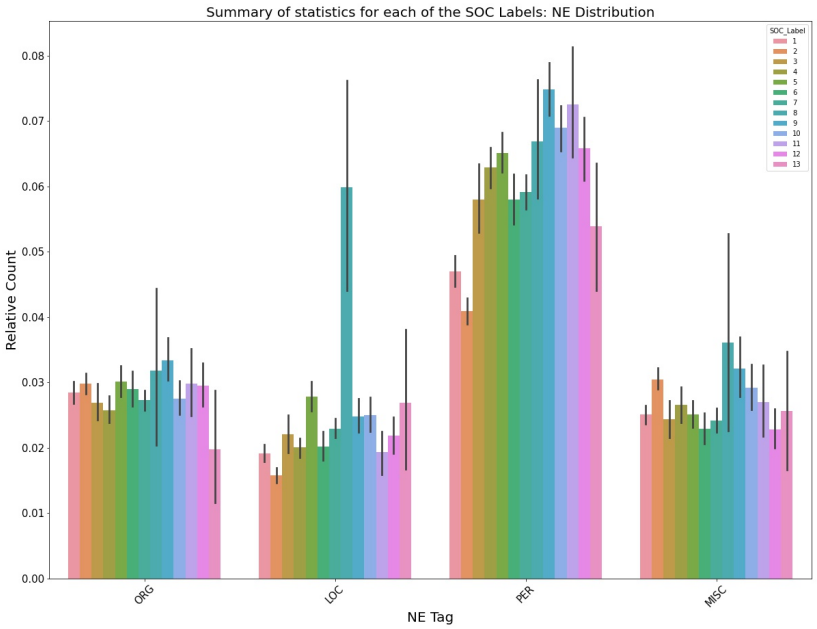


Figure 5.6: Bar Plot illustrating the Relative NE count per each Occupational Class

"human" and "resource" appear as MFW unigrams within (7) in Figure 5.7, individually they may also appear in other classes, such as "resource" in (1) of the same figure; individually the terms carry a different semantic meaning than when taken as the bigram collocation, "human resource". Bigrams can therefore be used to help to capture additional information and ultimately may be useful for leading to a higher performing model through increased pattern understanding within the data. In order to visualise the frequency of bigrams, the 20 most frequent word pairs are extracted. Examples of these most frequent collocations are illustrated below in Figures D.1, D.2 and D.3. The full list of the figures for all of the classes can be found in Appendix D.

5.3.4 Final Features

After having performed the detailed EDA, although the bigrams do prove to show some patterns which could be indicative for each of the occupational classes, as a result of time-constraints they are not implemented within this research. It could be suggested however in future works to further use this as a comparative feature, based on the visualisations shown within this Chapter. Nonetheless, the illustrations produced will also serve as a basis of comparison with the final results in Chapter 6, in order to determine and visualise to what extent the results vary from the original most frequent terms.

Despite the fact that neither any specific linguistic features nor statistical features and explicitly implemented, since the most frequent unigram cloud visualisations created for each of the occupational groups demonstrated to be highly indicative, it is also decided to implement Term frequency-inverse document frequency (TF-IDF) as an additional

text representation feature; this is explained in the subsequent section, Section 5.3.5. The main source therefore in the Feature comparison will be based on using the results of the extracted terms from the Term Extraction system as well as comparing the two text representation types, namely FastText and TF-IDF.

5.3.5 Text Representation

5.3.5.1 FastText

Word embeddings have demonstrated to be very effective for various challenges within the NLP field, primarily as they can capture additional semantic and syntactic information of lexical units, where lexical information may in fact be sparse; "you shall know a word by the company it keeps" (Firth, 1957). Since German contains a multitude of compound nouns, it is important to carefully select an embedding model capable of capturing this type of granularity. One of the most renowned word embeddings models at present is FastText[1]. FastText is a set of pre-trained word embeddings, of which use a continuous bag of words (CBOW) and represent each word as series of character n-grams, capturing information at the *sub-word* level. This subword level of granularity is thus hypothesised will help to ascertain an improved performance of the system than without them. Since the job postings are ultimately documents, composed of sequences of sentences and even sequences of tokens, a similar method is performed with regards to transforming the multi-token terms within the term scoring component of the Term Extraction system. Here, each single job posting is thus split into a series of sentences before each single token is then transformed into its respective FastText vector representation. Following this, pooling is applied to generate sentence embeddings before these are then further pooled to generate *document* embeddings.

5.3.5.2 TF-IDF

Term frequency-inverse document frequency (TF-IDF) measures the weighted frequency of which a lexical unit is used within a given document (TF) whilst also *punishing* the score if the lexical unit occurs very frequently in multiple documents. The most representative terms within one single document are thus scored with a higher TF-IDF since they appear frequently within that single document whilst seldom across the entire corpora. As a result of stopwords being removed from the data as outlined in Chapter 3, the MFW unigrams presented in the previous section show there is a strong correlation with the selection of single tokens within each of the occupational classes. Given the fact that this type of text transformation is specifically designed to take this type of information into account, whereby each class contains a unique selection of relevant words, TF-IDF is implemented as a comparative representation.

[1] *https://fasttext.cc/*

(1) Management, Business and Financial

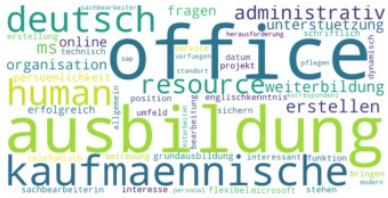
(2) Computer, Engineering, and Science

(3) Education, Legal, Community Service, Arts, and Media

(4) Healthcare Practitioners and Technical

(5) Service

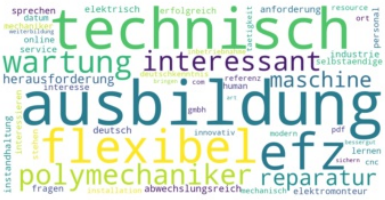
(6) Sales and Related



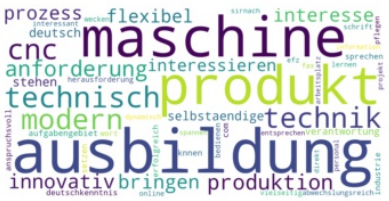
(7) Office and Administrative Support

(8) Farming, Fishing, and Forestry

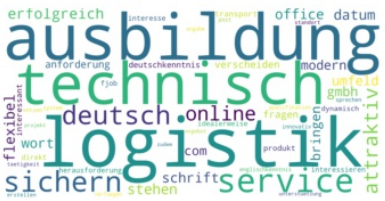
(9) Construction and Extraction



(10) Installation, Maintenance and Repair



(11) Production



(12) Transportation and Material Moving

(13) Military Specific

Figure 5.7: Word Cloud visualisations illustrating top 50 key words within each of the Occupational Classes

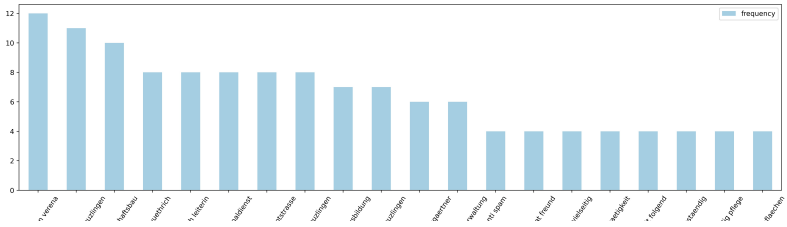


Figure 5.8: Bar Plot illustrating top 20 Bi-grams within soc label 8, “Office and Administrative”

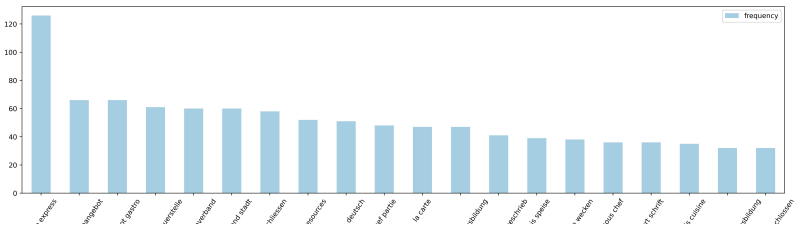


Figure 5.9: Bar Plot illustrating top 20 Bi-grams within soc label 5, “Service”

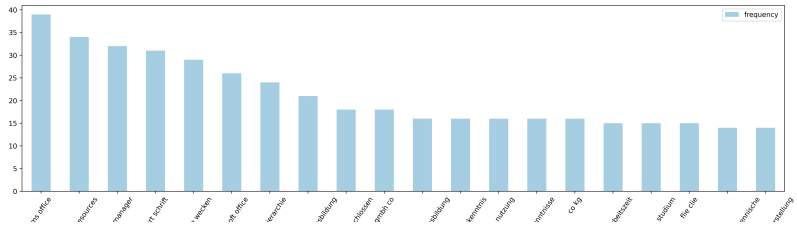


Figure 5.10: Bar Plot illustrating top 20 Bi-grams within soc label 6, “Sales and Related”

5.4 The Classification Models

As aforementioned in recent works, the use of manually extracted linguistic features in combination with traditional ML techniques in comparison to DL architectures have both proven to ascertain high performing results for multiple NLP and text classification tasks.

As a result, in order to compare the performance of both ML and DL techniques, a total of two classifiers are implemented, namely Support Vector Machine (SVM) and a Convolutional Neural Network (CNN). Regarding the practical implementation of the models, the python libraries Scikit-learn[2] and Keras are selected. The theory behind each of the models alongside the reason for their implementation within this research is explored in more detail below.

5.4.1 SVM

Support Vector Machine (SVM) classifiers ultimately uses a supervised learning algorithm to classify data through mapping and separating data points in a high dimensional space using *hyperplanes*; distinguishing distinctive boundaries between classes. Unlike other ML techniques, SVM is capable of *understanding* and distinguishing patterns when performing non-linear classification (Colas and Brazdil, 2006), (Zampieri et al., 2019). For this reason, alongside the fact that it is considered to be one of the most high performing and efficient ML techniques for this type of task, SVM was selected as the comparative ML technique for the paper.

5.4.2 CNN

Convolutional Neural Networks (CNN) are specialised *deep* artificial networks, consisting of many hidden layers, specifically in this case called *convolutional layers*. Unlike traditional neural networks, which only are unable to catch contextual information as a result of a single value used as input, CNNs use a filter which contains a neighbourhood of values; allowing them to capture this missing contextual information.

The convolutional layers within CNNs aid in detecting specific patterns within the input data, whether this input entails images or, in this case, sequential data like text. They are therefore able to automatically extract distinctive feature vectors from the training data to create sub-samples; assuring the best class separation possible. Global maxpooling layers can also be implemented in an attempt to reduced the size of the feature vector into a fixed length vector, thereafter enabling the extraction of long range dependencies present in the data. Recent studies have illustrated how these deep CNNs have the ability to attain high scoring performance, even in their *simplest* forms with little to no hyper-paramter tuning (Kim, 2014). As a result, a simple CNN model is implemented in itself; the exact parameters are illustrated in Section 5.5.

In order to input all of the extracted skill and education related alongside the vacancy title and description into the cnn classification model, the features are concatenated together as strings before thereafter being converted into the respective text representations with regards to the embedding models.

[2] `https://scikit-learn.org/stable/`

5.5 Experimental Setup

In this final section, I outline the final decisions made during the experimental setup in terms of any of the final system parameter tuning. As shown below, the specifications for each of the models differ between each other in order to try and develop the most optimal solution for each. For example, when applying a confidence of 0.9 to both the SVM and the CNN, the SVM did not return any pseudo-labelled results throughout any of the iterations where as nearly all of the unlabelled data was confidently labelled by the CNN in the same setting.

- **Both**
  - Confidence Threshold: **Upper Quartile Range (Q3)**
  - N-iterations: **10**
    The model will stop iterating over the data and labelling it once the number of iterations is greater than 10.
  - Minimum Rest: **5%** of unlabelled data instances; **27,399**.
    This means that the model will either stop after the *n-iterations* or if the number of unlabelled training instances is equal to or less then the *minimum rest*.
- **SVM**
  - Kernel: **Radial Basis Function (RBF)**
    This RBF, otherwise referred to as Gaussian, kernel has an infinite number of dimensions which means it can compute more complicated decision boundaries and find a better hyperplane that separates the data. This kernel is selected as a result of the fact that it is the optimal kernel to use when the number of features is *less* than the number of training examples, similar to this research.
  - **Calibrated Classifier Cross Validation (CCCV):**
    Although it is possible to generate the probability scores using the more traditional SVM with the input of an additional parameter setting, I also compare training the SVM as a CCCV. This CCCV model is built specifically to better calibrate the probability predictions of the model and produce the confidence scores required for the pseudo-labelling architecture. In addition to this, the CCCV applies a stratified approach and tends to not bias as much towards majority classes, which may prove to increase the performance of the model given the fact that there is an imbalanced label distribution within the dataset. However, it should be noted here that the calibration of a model can also result in requiring more extensive computational resources and time, therefore it is unknown exactly how practical the calibration will be, given the size of the unlabelled training dataset. Both the non-calibrated and calibrated SVM will be tested firstly in a supervised setting, using the labelled train and test data, to determine which appears to achieve a better performance. The model which does in fact obtain the highest F1 score will be used as the baseline for the remaining experiments using the SVM. The aim of doing this is to generate the most confident pseudo-labels across all classes, including those more under-represented, during the first itera-

tion which could help the model improve in later iterations. The following parameters are set within the CCCV.

  * Method: **Isotonic** The default function method applied to the CCCV in calibrating the probability predictions usually is "Platt Scaling" of which performs "most effective when the output distortion of the predictions is Sigmoid shaped", and transforms the SVM outputs to posterior probabilities (Niculescu-Mizil and Caruana, 2005). However, due to the imbalanced distribution in the dataset, "Isotonic" regression is instead implemented on the basis of it reportedly having an "overall [better] performance on imbalanced datasets than parametric and other complex non-parametric methods" (Huang et al., 2020). Isonotnic regression assumes that the calibrated confidence probabilities are monotonically increasing and therefore rescales this output by fitting a free-form line. Isotonic regression is also known to perform better for multi-class classification, as undertaken in this research, whereas Platt Scalaing is more commonly applied to binary classification tasks.

- **CNN**
  - Number of Epochs: **30**, **15 10**
    Here, a total of 15 epochs are used within the majority of the training experiments using the CNN however as the size of the labelled data changes, this parameter is also adapted accordingly to fit within the limitations of computational resources whilst also obtaining a good performance. For the initial models, 30 epochs are implemented in which the labelled data remains under 100,000 instances. Following this, the default epoch number of 15 set. Once the labelled data surpasses the threshold of 300,000, the epoch number is then reduced to just 10 in order; this was the optimal integer for each before the models appear to begin overfitting.
  - Convolution Layers: A total of two **Maxpooling** layers alongside a single **Global Maxpooling** layer are implemented within this CNN. The Maxpooling layers are used to reduce the number of dimensions and preserve the context whilst the Global Maxpooling is used to make the output of the previous layer compatible with the fully connected layer.
  - Activation Types: **Sigmoid, Rectified Linear Unit (ReLu), Softmax**: Sigmoid is applied initially which transforms the input to an output between 0 and 1. Here, a high value will have a high probability but not necessarily the *highest* probability and can also lead to vanishing gradients. Thus, ReLu is applied to the hidden layers of the Neural Network (NN) with the aim of helping to avoid and rectify any vanishing gradient problems. Softmax is applied to the final layer in order to generate the predictions necessary for determining and evaluating the confidence scores, since it transforms the output between 0 and 1 for each neuron; as desired for calculating the probability percentages.
  - Cost Function: **Categorical Cross Entropy** - Decay using Adam - Learning Rate of 0.0001

Chapter 6

Evaluation

This chapter presents the final results of the Occupation Classification experiments undertaken, explicitly comparing the performance of the different combinations of text representations, ML models and and notwithstanding the extracted term features described in Chapter 4. Section 6.1 describes the evaluation metrics used to compute the final scores, namely Precision, Recall and F1. Following this, section 6.2 presents the results from each of the experiments in a clear and systematic structure, clearly illustrating which models ascertained the best results alongside those where improvement can be made. In the final section, namely Section 6.3, I perform a detailed exploration into the results, primarily focussing upon the errors produced by the models. Here, a sample of the incorrectly predicted samples from a select few of the experiments are analysed in a way to try and determine any possible causes.

6.1 Evaluation Metrics

The selection of an appropriate evaluation measure is one of the most important things not only in Natural Language Processing (NLP) but Machine Learning (ML) as a whole and is particularly important given the fact that both a model and its respective the results should be *explainable*. In order to evaluate each of the comparative experiments within this research, this section thus outlines and describes the *three* main metrics used to measure the performance of each model, namely Precision, Recall and F1.

Figure 6.1 illustrates the confusion matrix for evaluating a given model, based on the number of correctly or incorrectly labelled instances.

6.1.1 Precision

$$Precision = \frac{TP}{TP + FP}$$

Precision is used to ultimately show exactly how *precise* a given model is, determined by how many of the instances are predicted *correctly*. It is the ratio of correctly labelled instances based on all of the positively predicted instances of the model. It is calculated

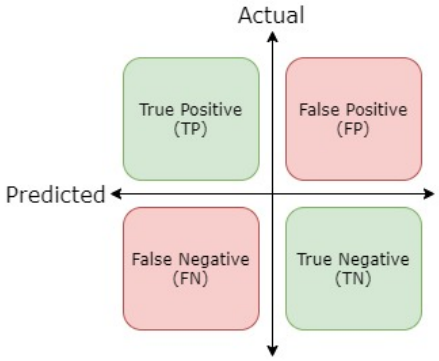


Figure 6.1: 2x2 Confusion Matrix

therefore by dividing the number of True Positives by the sum of the True and False Positives.

6.1.2 Recall

$$Recall = \frac{TP}{TP + FN}$$

Recall, on the other hand, relates to the number of correctly labelled instances based on all of the predictions in the given class. There is often a trade-off between recall and precision, whereby a low precision and a high recall would mean that the model correctly predicted a large number of instances within a given class correctly, however other instances not from this class are also predicted the same label. Alternatively, a high precision and a low recall indicates that although the model does predict the correct instances of a given class, this happens seldom and therefore not all of the instances are correctly observed.

6.1.3 F-measure

$$F1 = \frac{2TP}{(2TP + FP + FN}$$

Although both precision and recall are both informative, often there is a trade-off between them, as seen in the previous subsection. For this reason, the F-measure is also used to provide a single measurement that combines the two. The F-measure, otherwise referred to as *F1*, is the weighted mean of both precision and recall and is most applicable when there is an imbalance of classes within the dataset, as is used in this research.

6.2 Results

The F1 scores are selected as the primary scores when comparing the performance of each of the models, due to the fact that they take into consideration both the False

Positive (FP) as well as the False Negatives (FN), equally; using the weighted average of both precision and recall. In order to determine what influence this pseudo-labelling semi-supervised approach will have on the results, firstly each of the basic models alongside the text representations are implemented in a supervised setting; using only the labelled training set and evaluating using the test data. Through initially running these supervised experiments, both the performance of the models using a smaller dataset can be observed whilst also serving as a comparative aspect upon obtaining the semi-supervised results; showing to what degree leveraging the unlabelled in a semi-supervised pseudo-labelling approach can in fact adapt the decision boundary and improve the overall performance. As mentioned in the previous chapter, due to the imbalance of the dataset, it was decided to test the supervised SVM with a *Calibrated* SVM. The goal of using this Calibrated classifier is to help stratify the output and try to avoid the SVM model leading to any bias towards the classes with a higher number of instances.

Model + Representation + Features	**F1**
SVM + TF-IDF	0.65
SVM + FastText	0.59
SVM + TFIDF + Education + Skill	0.61
SVM + FastText + Education + Skill	0.61
SVM Calibrated + TFIDF	**0.68**
SVM Calibrated + FastText	0.59
SVM Calibrated + TFIDF + Education + Skill	0.63
SVM Calibrated + FastText + Education + Skill	0.53
CNN + None	0.26
CNN + None + Education + Skills	0.39
CNN + FastText	0.42
CNN + FastText + Education + Skills	0.45

Table 6.1: Results in a Supervised Setting (F1 measures)

The supervised results illustrated in Table 6.1 show that the SVM in both scenarios appears to ascertain much higher results in all cases against the CNN, with the Calibrated SVM with TF-IDF achieving the best F1 score. Upon a further analysis using the classification reports as shown in Appendix E, we can see that the Calibrated SVM is able to predict instances for nearly all of the classes, par two of the minority classes; classes 8 and 13 containing only 8 and 4 instances, respectively. On the other hand, the CNN without any embeddings only assigns labels to the top 6 majority classes, disregarding the remaining classes altogether. Since the SVM appears to thus perform substantially better using the smaller dataset in a completely supervised setting, I decided to further include additional experiments using an ensemble approach. Here, the ensemble method involves initialising the system using a Calibrated SVM for the first model, before then training the remaining models using a CNN. The reason for this is to try to avoid increasing the distribution skew after the first model which is hypothesised could thus lead to an even larger imbalance of classes in the later iterations. The additional reason for implementing this Ensemble approach using the Calibrated SVM paired with the CNN is based on my hypothesis that the performance of the CNN will improve as the number of training examples increases. Regarding the remaining semi-supervised experiments using the SVM, only the Calibrated SVM is implemented

since it ascertains the highest result in a supervised setting. This model also applies a Softmax layer, allowing for more interpretable confidence probabilities between 0 and 1.

Experiment	Model	Representation	Features	#Models	#Total Labelled	F1 Scores
1	SVM	TF-IDF	None	5	257,853	0.38*
2	SVM	FastText	None	5	257,854	0.57*
3	SVM	TF-IDF	Sk + Ed	5	257,855	0.61*
4	SVM	FastText	Sk + Ed	5	257,854	0.58*
5	CNN	None	None	10	347,578	0.30
6	CNN	FastText	None	10	354,558	0.41
7	CNN	None	Sk + Ed	10	347,579	0.32
8	CNN	FastText	Sk + Ed	10	347,579	0.52
9	Ensemble	TF-IDF + None	None	10	347,578	0.53
10	Ensemble	FastText	None	10	356,541	0.50
11	Ensemble	TF-IDF + FastText	None	10	347,580	0.60
12	Ensemble	TF-IDF + None	Sk + Ed	10	347,581	0.58
13	Ensemble	FastText	Sk + Ed	10	347,579	**0.62**
14	Ensemble	TF-IDF + FastText	Sk + Ed	10	347,581	0.61

Table 6.2: Results in a Semi-supervised Setting (F1 measures). Sk = Skill, Ed = Education

The results for each of the semi-supervised experiments are presented in Table 6.2. In addition to the Experiment number, F1 score and technical parameters within each experiment, the final number of models used to train the system alongside the total number of labelled instances used for testing are also shown in the table. Due to computational limitations, the SVM models are only able to be both trained for a total of 5 iterations of the architecture; following this, the resources become exhausted and and memory errors occur. The models which do not pseudo-label all of the unlabelled data before either reaching the minimum rest set at 5% or before reaching the threshold maximum of 10 iterations are marked with an asterisk (*) in the result tables.

This reduced number of models within the SVM experiments inevitably proves problematic for comparative purposes in that the results cannot be *directly* compared to those of the CNN.

Nonetheless, when comparing the SVMs in a supervised and semi-supervised setting, it is particularly surprising to see that the introduction of additional data has in fact worsened the overall performance, in nearly all experiments. Particularly in Experiment 1 which obtained the highest result of 0.68 in a supervised setting, we see a drop in 0.30 when using the pseudo-labelling approach. From visualising the classification report shown in Appendix E, we can see that although the model here in Experiment 1 has managed to assign labels to even some minority classes, all except SOC Labels 8 and 13, the model does not pseudo-label *any* instances for one of the majority classes, namely SOC Label 1. It is suspected that this is as a result of the *stratifying* approach used in the calibrated SVM as well as the lack of generalisation power from the TF-IDF representation. This deterioration in F1 is also seen in Experiment 2, dropping from 0.59 to 0.57, however the same number of classes are still assigned labels in both scenarios. Nevertheless, the incorporation of extracted terms within the SVM model with FastText however, do in fact also improve results, albeit somewhat minimal, with

Experiments 4 obtaining a final result of 0.58, increasing by 0.05 than that of the supervised setting. Based on the fact that this latter model does at least appear to improve using the pseudo-labelling system and the extracted terms, it can be suspected that with further iterations, the results may still improve further, though this cannot be certain.

All of the CNNs and Ensemble models, however, do in fact reach the breaking counter threshold of 10 models, meaning that they could have also continued in further iterations to label all of the data. Each of the experiments here tend to label around 350,000 instances upon reaching the training counter of 10, whereas the SVM models are able to label around 250,000 instances at just the 5th iteration before testing.

Although it is evident that there is still room for improvement, given the complexity of the task with a specific and unique set of data which started off completely unlabelled and that this is a multi-class classification task with 13 different classes, the results are nonetheless promising. We can see that *nearly* all of the experiments undertaken in a supervised setting are in fact improved through the incorporation and leveraging of the unlabelled data in this pseudo-labelling system. For example, with regards to the CNN with no representation nor features in Experiment 5 improves its performance compared with that of it in a supervised setting through the incorporation of the unlabelled data, albeit minimal at 0.04 difference. It is very evident here through increasing the size of the training data, the overall performance of the CNN model increases.

It should be noted here that as a result of the probability parameter within the SVM and the large number of training instances after the first iteration, a single iteration of a given SVM experiment took up to 4 days. For this reason, the extensive computational time needed to run these experiments also acts as a disadvantage in comparison to both the CNN and Ensemble experiments, particularly from an industry point of view.

Nonetheless, the extraction of the skills and education terms combined as features clearly does positively impact the majority of the experiments in comparison to those without,with the best model at Experiment 13 achieving a F1 score of 0.62 with their incorporation alongside the FastText embeddings. It is clear that there is vast room for improvement, however given that the CNN in a supervised setting with the extracted terms only achieves 0.45, and yet in this semi-supervised setting achieves 0.52, it is clear that the pseudo-labelling approach does improve the performance; albeit somewhat small. Both with and without the inclusion of the extracted Skill and Education terms, the SVM outperforms the CNN with each of the different text representations. The highest F1 score achieved by the CNN with FastText and the extracted terms is 0.52, whilst the SVM with FastText alone results in a score of 0.57 at just 5 iterative models. Despite the fact that the Ensemble approach implemented in Experiment 10 does not beat the calibrated SVM implementation with the same features and text representation in Experiment 2, it does however ascertain a higher final score than in Experiment 6. This shows that through using the SVM in the first iterations, an improved distribution of the imbalanced classes are assigned which allows the CNN to improve its generalisation power using FastText.

From looking at the classification report shown in Figure 6.3 for the best performing experiment, namely Experiment 13 with FastText and the Skill and Education extracted features, we can see that although the iterative training, this optimal model does not manage to predict any instances for three of the classes; SOC Labels 8, 11

	precision	**recall**	**f1-score**	**support**
1	0.71	0.38	0.50	184
2	0.64	0.85	0.73	198
3	0.75	0.31	0.43	49
4	0.70	0.72	0.71	120
5	0.72	0.67	0.69	127
6	0.49	0.65	0.56	85
7	0.64	0.71	0.67	162
8	0.00	0.00	0.00	8
9	0.50	0.61	0.55	92
10	0.53	0.70	0.61	122
11	0.00	0.00	0.00	25
12	0.60	0.49	0.54	55
13	0.00	0.00	0.00	4
accuracy			**0.62**	1231
macro avg	0.48	0.47	0.46	1232
weighted avg	0.62	0.62	0.60	1231

Table 6.3: Classification Report for Experiment 13

and 13. In addition to this, despite the highest achieving class is unsurprisingly the majority class, namely SOC Label 2 with F1 of 0.73, it is particularly surprising to see the second highest achieving class, SOC Label 4, achieves a F1 of 0.71, despite being the 6th majority class. Furthermore, the classification report shows that although in general both the precision and recall is relatively high for each of the labels, recall is also fairly low in some cases, as seen in SOC Labels 1, 3 and 12. This means that even though the model does not assign instances to these labels often, when it does so, these are labelled correctly.

Granted that the results in the classification report do offer some insights into the highest performing model, further analysis is needed to understand them better. As a result, in order to investigate these results, an in-depth Error Analysis into the possible causes for hindering the predictions is undertaken in the subsequent section, Subsection 6.3.

6.3 Error Analysis

This sections presents an analysis into the errors generated within the output by each of the models. Since there are a variety of different comparative aspects to explore and analyse within each of the experiments undertaken, this section will be split into the following subsections. Firstly, in Subsection 6.3.1, I take a look into a sample of explicit errors within the data produced by Experiment 13, the highest performing experiment, in order to try and see whether any patterns exist between them. Ultimately, this type of manual exploration is used to determine what the main causes of the errors appear to be using the human eye and domain knowledge. The subsequent subsections then are used as a more automatic analysis, through producing statistical visualisations in order to understand further where exactly the main source of improvement lies. Consequently, since the dataset itself is imbalanced, Subsection 6.3.2 examines the *distribution* of

the classes predicted by each of the models in the results. As characteristic of the pseudo-labelling architecture, each Experiment is comprised with a multitude of models, each aiming to *pseudo*-label the data. Therefore the distribution of predicted labels here is not only analysed for the final model, but rather compared with any possible further skews generated throughout each of the iterative models. One of the most imperative components as part of this semi-supervised pseudo-labelling system is the confidence threshold set since it determine what percentage of the data is used during the next iteration of training. For this reason, the final section, Subsection 6.3.3 inspects and scrutinizes the confidence intervals of each of the models in the best performing Experiment whilst inevitably comparing these with the number of correct predictions generated by the model.

6.3.1 Data Analysis

In order to perform the manual inspection of the errors produced from the models, I analyse a sample of explicit examples from the highest performing experiment. Thus, I look into the errors produced by Experiment 13 using the ensemble approach of a SVM and CNN with FastText representation and extracted term features. The results from Experiment 13 illustrate a total of 764 job postings which were classified correctly whilst a total of 467 were incorrectly labelled by the model. A random sample of 200 instances of this latter subset is withdrawn for scrutiny, as explored and described below.

One of the first aspects noticed upon commencing the Data Error Analysis is specifically regarding the extracted terms related to each respective instance. One of the main things seen here is the fact that some instances do not have any skills nor education types extracted for them, which probably has contributed to their false prediction in this model. This is not particular to one single occupational class, but rather appears across various classes, mostly within very short job descriptions. For example, one of the postings advertising a Service-related cleaning position only consisted of 50 words and did not have any extracted Skill nor Education related terms. Comparing the number of extracted Skill terms in relation to amount of Education terms, it is clear that there are far greater missing values for the Education field. Inevitably by having a large number of *missing* features across various classes, it introduces confusion into the model since these may be grouped together as a class; with the lack of information in the feature becoming a feature in itself.

In addition to this apparent lack of extracted education terms, there is also seemingly a large *overlap* of skill-related terms extracted for the incorrectly predicted instances. Particularly, soft skills such as "responsible", "team-orientated" and "communicative" are relevant for nearly all of the occupations advertised therefore this brings noise into the model. This could therefore blur the boundaries between already some fuzzy classes, such as those between the Sales (SOC Label 6) and Office and Administrative (SOC Label 7) related occupations. Here, a large number of not only soft skills overlap with some including being responsible, flexible and having good communication skills, but also some more occupation specific such as having specific language skills, having a good competence in Microsoft Office and being customer-orientated. Given that there is a large number of soft skills which do appear somewhat generic across *all* of the classes, it can be thought that through more cleaning and filtering, it could reduce the blurred boundaries and improve the model's performance.

Nonetheless, it is particularly interesting to see that although some instances are categorised as *incorrect* against the *gold* annotation labels, from further inspection it can be seen that the model has actually predicted correctly and the fault is at that of the annotator. For example, one instance entitled as a "Kunden Berater Supply Chain", *Customer Support*, ultimately refers to an occupation involving the sales component within supply chain management; the skill extracted terms here are "Customer-orientated" and "reliable" whilst the education related terms are just "Further Education" (each already translated in English). This instance alone is annotated as SOC Label 5, the *Service* industry however, predicted by the model as SOC Label 6, the *Sales* related occupations; the fault is of the human expert annotators. Similar instances can be seen with, for example, a job posting advertising for a"Site Production Manager" with Education terms "University" and "Master" extracted is predicted by the system as SOC Label 1, however annotated as SOC Label 12. Given further inspection, since this particular occupation requires a higher level of education, it would in fact be categorised into SOC Label 1, as according to the annotation guidelines developed through intermediary analysis.

However, the model does appear at fault for other similar instances whereby a given instance is predicted seemingly based on *key terms* rather than the occupation itself. For example, a job posting with the title "Supply Chain Opportunities Pharmaceutical" and extracted skills of "Medicine" and "Medical Devices" is predicted as SOC Label 4, the *Healthcare*-related occupations group, whereas labelled as SOC Label 1, *Business, Management* by the given annotator. Here, the model has clearly detected a pattern between the medical related terms and associated it with the respective group on that basis, as even a human without domain experience may do. Another similar pattern with the model picking up key terms is seen with the following instances "Logistik Controller" whereby the model has associated the term controller with SOC Label one, a job title highly associated with that related class. It is clear why there is ambiguity however there, because even as an expert in the field it is relatively difficult to determine which group the occupation advertised should be classified into.

The next potential cause of errors seems to partially based on the language of the data itself. As mentioned in the previous chapters, the data does not only contain German language instances, but rather is filled with a large number of English postings despite filtering. From the analysis here, we can see that there are also other languages included in both the titles and descriptions, with the majority language being French. Particularly occupations related to Service occupations, job postings seem to have a French job title yet is followed by a description in the English language. This has also led to multiple incorrectly predicted instances as a result, with some occupation titles including "Chef de Partie" and "Chef de Service". As a potential result of the English job descriptions, it can also be seen that very *few* Skill and Education class terms were extracted from the Term Extraction system. This links back to one of the first points made in this subsection, in that a lack of information within the features could have led to the addition of noise. Despite the number of Anglicisms and French borrowings in German, the extent and variety of different languages also could have hindered text representation with the FastText embeddings, since these are not multi-lingual but rather German based.

Finally, the inspection and analysis of the errors highlights that in fact some of the incorrectly labelled data point could be as a result of the preprocessing and cleaning

process. For the extracted terms, there is still a high number of terms which are not related to the respective class and do not contribute any semantic contribution to the classification of a given occupation. For example, terms related to days of the week (e.g. *Monday*), terms related to benefits of the company (e.g. *remote* and *free breakfast*) as well as generic advertisement terms such as *contact form* and the names of contact person(s). Through an even more intense cleaning and creation of a tailored stopword list, this noise would be significantly reduced. The cleaning of the job posting descriptions themselves, on the other hand, also show that a large number of terms are in fact lemmatised incorrectly, as a result of the NLP tagging and preprocessing; adding noise again to the data. This could be as a result of the large number of compound words within the German language and therefore compound splitting could be used in future work to avoid this.

We can thus summarise the causes of errors within the subset of 200 instances therefore into five main groups: (1) a lack of extracted terms, (2) too similar overlap of extracted terms, (3) false annotations, (4) language noise with not only German job postings and finally (5) incorrect NLP tagging leading to false lemmatisation and further noise incorporation. After having investigated the language and explicit examples of the incorrectly predicted instances, the subsequent section now analyses the pseudo-labelling system itself with regards to any technical causes for the errors, particularly concentrating on any adaptions in class distribution.

6.3.2 Distribution Analysis

As mentioned in previous chapters, the data at hand is highly imbalanced in that the majority class, SOC Label 2, entail a total of 595 training instances whilst the minority class, SOC Label 13, contains a mere 11. In order to determine whether the incorporation of Education and Skill terms has any impact on the distribution of the predicted classes, here I present visualisations illustrating any changes in the skew between classes within each of the iterations in a given model; to see whether the models bias towards the prevalent classes. For comparative purposes, the models selected all utilise FastText as the respective text representation and thus a total of 6 illustrations are produced; both with and without the extracted features for the SVM in Experiments 2 and 8, the CNN in Experiments 4 and 10 and finally the ensemble methods in Experiments 6 and 12. The first visualisation however, will illustrate the distributions of the lowest performing model using FastText as the text representation, namely Experiment 4; the CNN without extracted term features.

As shown here in Figure 6.2, the *pure* CNN in Experiment 4 only predicts a total of 4 classes out of the entire 13 within the *first* model of the architecture, with them all being majority classes. As a result, we can see in the graph that this strongly heightened imbalance within the data, which inevitably leads to the remaining models mirroring this bias towards the prevalent classes and assigning labels only to these classes; in a negative cycle. Even in the second iteration we can see that the model biases even further to the majority classes of the predicted labels, with the proportion of samples between SOC Labels 4 and 5 increasing; eventually leading to zero instances predicted as SOC Label 4 in the last iteration.

We can see, however, that the Calibrated SVM in Figure 6.3 produces an adverse result in terms of the distribution of classes. As shown in Figure 6.3, despite fewer

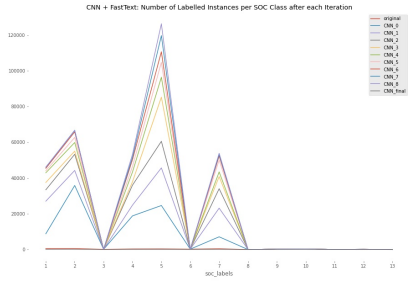


Figure 6.2: Line Graph of the Distribution Skew in the Number of Instances per Class within each iteration for Experiment 4 (*CNN*, FastText without the Extracted Skill and Education Terms)

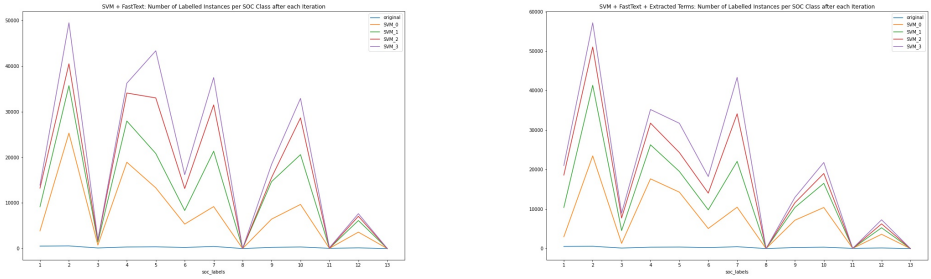


Figure 6.3: Line Graph Comparison of the Distribution Skew in the Number of Instances per Class between *SVM* Models (FastText) both with and without the Extracted Skill and Education Terms. (Experiment 2 vs Experiment 4)

iterations due to computational time in generating the predictions itself, the SVM assigns a stratified approach in labelling the pseudo-test data and does manage to label instances to the nearly all of labels, despite the small training sample. We can see in each of the sub-figures that even without the incorporation of the extracted features, the SVM assigns nearly all of the classes with at least some new instances after the first iteration. Given that the remaining two classes, SOC Labels 8 and 13, had less than twenty instances each during the training phase, the fact that both of the respective models label instances foe the remaining 11 classes is very promising. The introduction of extracted Skill and Education terms in the bottom sub-figure illustrates how the model is able to lessen the skew in number of instances per class against SOC Label 9.

In Figure 6.4, the visualisations further show that with the incorporation of FastText, the CNN without the extracted terms is able to generate pseudo-labels for at least one extra occupational class, totalling 5 and out 13 classes in comparison to the model without, which stands at 4. These visualisations particularly emphasise how integral the predictions and consequently class distribution is during the first model, since the remaining iterations replicate a similar proportion; as seen also with the heightened negative skew in Figure 6.2. The increase in number of labelled instances after the first model, then causes the CNN to bias towards the majority classes, whilst also lengthening the difference in instances between that of the minority and majority class.

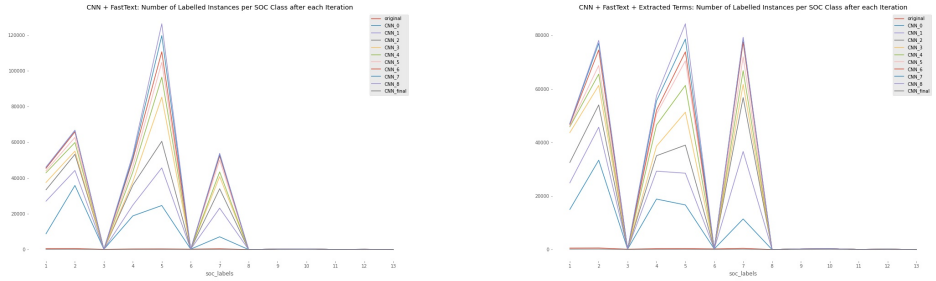


Figure 6.4: Line Graph Comparison of the Distribution Skew in the Number of Instances per Class between *CNN* Models (FastText) both with and without the Extracted Skill and Education Terms. (Experiment 6 vs Experiment 8)

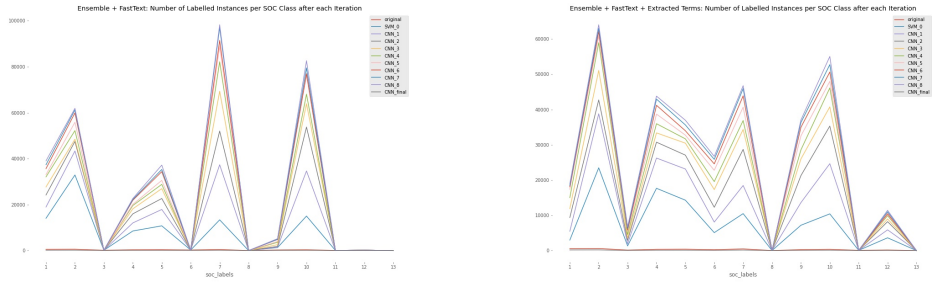


Figure 6.5: Line Graph Comparison of the Distribution Skew in the Number of Instances per Class between *Ensemble* Models (FastText) both with and without the Extracted Skill and Education Terms. (Experiment 10 vs Experiment 13)

Based on the visualisations produced for both of the SVM and CNN experiments shown, it seems apparent that the first iterative model of the pseudo-labelling architecture acts as the key drive for the later models; each remaining model thereafter produces a similar distribution based on that of the first. To further investigate this assumption and see whether even the CNN is able to mirror the more stratified distribution after the first iteration of the SVM, I now look into the distribution of the ensemble approaches.

As mentioned, the ensemble implementation utilises the calibrated SVM for the first iteration whilst the remaining are ran using a CNN. Figure 6.5 emphasises the statement in that the distribution of predictions generated by first iterative model does in fact shape the remaining iterations. Comparing this same figure with the distribution in 6.4, we can evidently see that through generating a more stratified distribution in the first iteration with the SVM, it leads to the remaining CNN models mirroring a similar shape. The analysis of the distribution of the classes at each iteration therefore explicitly demonstrates how the model used in the first iteration of the pseudo-labelling architecture has the capacity to determine the overall performance. We can see that it makes a negative impact if the distribution skew increases after the first iteration as shown in Figure 6.4 as the model will continue to bias towards the majority classes throughout the remaining iterative models, leading to a lower performance.

The distribution analysis results illustrated for each of the given experiments suggest

that the combination of the dataset and occupation classification task at hand is highly sensitive to not only the features implemented, but also the model utilised as well. The SVM does evidently perform at a much higher standard with the smaller dataset and is able to label the instances in a stratified way, without biasing towards the main majority classes. This is adverse for the CNN experiments, whereby only the majority classes are assigned new instances which thus heightens the imbalance and skew of the data; leading to similar bias in later models. Regarding the SVM implementations, it has been reported that there are some issues with uses the probability generating function, namely *predict_proba*[1], as utilised in this thesis, with some noting its instability. This seems apparent in the first iteration of each of the Ensemble methods in Figure 6.5, which should mimic the distribution of the first models in the *pure* SVM experiments, as shown in Figure 6.3. Instead, despite the fact the *same* parameters were selected, we can see that the distribution of classes is slightly different, highlighting that it is somewhat unstable.

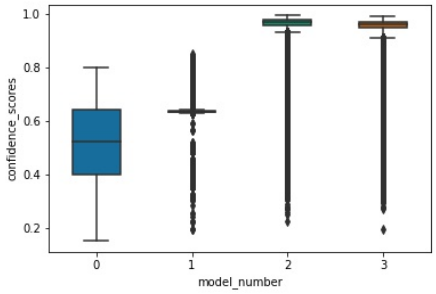
The visualisations clearly depict how in most cases the first model determines the distribution of the class predictions during the remaining iterations of the pseudo-labelling architecture. Since the SVM in a supervised setting does still ascertain a higher F1 score than that of the SVM experiments in a semi-supervised environment and the difference in the distributions of the same models, it leads us to question the confidence itself of the models; the supervised model may have not been *confident* in some predictions that were in fact correct and vice versa. As a result, this next section explores the confidence probabilities produced by each of the models to further determine whether this has a contribution to hindering the model performance.

6.3.3 Confidence Intervals

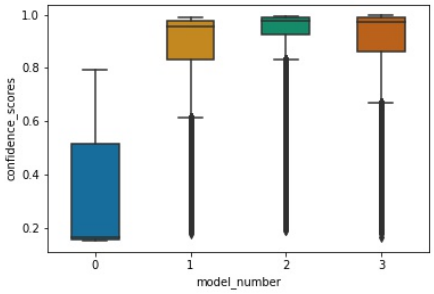
Confidence probability selection is one of the key components in the pseudo-labelling classification model, whereby the threshold will ultimately determine exactly how many predicted instances will be *pseudo-labelled* and added to the training data. As explained in Chapter 5, the confidence threshold score is initialised and updated automatically using the Upper Quartile (Q3) result based on the current model's predictions throughout each of the iterations. As mentioned, since the *supervised* Calibrated SVM still proves to be the highest performing model, it leads us to questioning the reliability and validity of not necessarily the semi-supervised architecture implemented in this research, but rather specifically the confidence scores. This section thus explores and analyses the confidence thresholds set within each of the models further to see whether there is a correlation between confidence and *correctly* predicted instances.

As shown in Figure 6.6, the confidence scores not only vary across the different types of ML models, however also across each model within a single given experiment. As hypothesised, the confidence scores of the given models adapt as more training data is incorporated within them. Particularly however what is interesting, is the fact that as the size of the training data is increased, in some cases, the models gradually become *less confident* in terms of their predictions, particularly regarding the CNN models. Although at the very second iteration within each of the experiments with the increase in training data the models all become drastically more confident, predicting, this

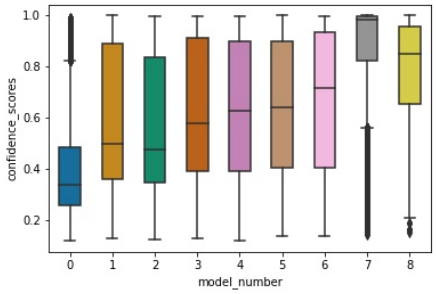
[1] `https://scikit-learn.org/stable/modules/generated/sklearn.calibration.CalibratedClassifierCV.html`



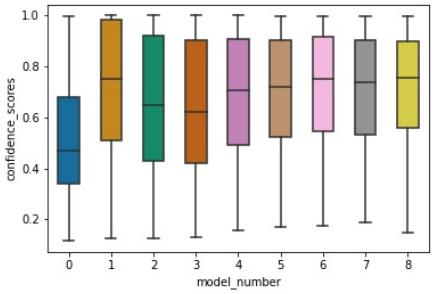
Experiment 2: SVM



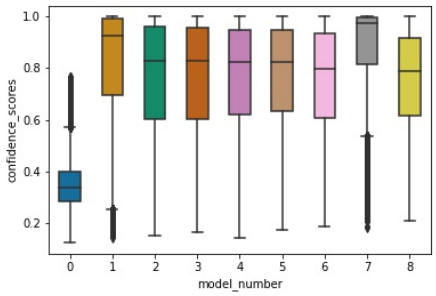
Experiment 4: SVM + Extracted Skills



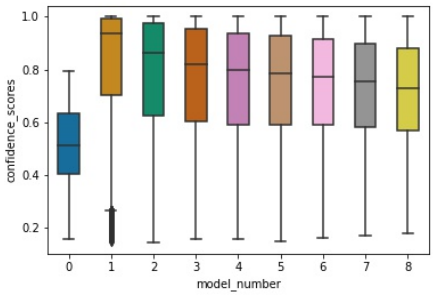
Experiment 6: CNN



Experiment 8: CNN + Extracted Skills



Experiment 10: Ensemble



Experiment 13: Ensemble + Extracted Terms

Figure 6.6: Box Plot illustrating the all levels of confidence produced by each iterative model with FastText comparing with and without Extracted Terms

pattern is not continued throughout all of the remaining iterations. For example, in Experiment 13 we see a very high confidence spread at the second iteration which gradually decreases as more training data is increased. Although this was initially surprising, this could in fact make sense; more data could also mean more noise, further ambiguities and thus additional uncertainty.

Despite the fact that the CNN appears to be generally a lot more confident in its predictions across each of the models within the experiment in comparison to the SVM, the final result shown in Table 6.2 leads us to the questioning the reliability in using the confidence scores as a whole. Figure 6.6 emphasises this with the low confidence intervals assigned during the first SVM iteration of Experiment 4 in comparison to those predicted in the same CNN iteration of Experiment 8, despite the latter achieving a sufficiently lower F1 result.

As shown in the sub-figures for SVM models, both the *pure* SVM experiments and the first iteration of the Ensemble, the confidence varies with no real correlation with regards to the extracted terms. Even the SVM models with the same parameters at the first iteration appear to have a very different confidence spread. For example, with the incorporation of extracted terms in Experiment 4, the highest confidence predicted is around 0.5 whilst the model with the same parameters in Experiment 13 predicts instances over 0.6 confidence. Again, this emphasises the instability of the probability prediction function and thus deems it to be not so reliable, despite the evident more stratified distribution illustrated in the previous section.

It is ultimately clear that an interactive confidence threshold is ideal when being set, since a give model's confidence does in fact vary given the number of training instances used as input. This same principle should be applied to each given model implemented in that not one single threshold is representative of a *good* confidence across all ML models. However, whether the SVM should be used in this setting is unclear given the instability of the predictions. Perhaps further research should be undertaken into different algorithms for calculating the confidence probabilities of the SVM model, but this is not explored further in this research.

6.3.4 Discussion

As shown in the error analysis, there are several classes that overlap, seemingly introducing confusion into the model. Even for the human eye, some of these instances prove to be difficult to assign to one single class, with the Education and Skill requirements also often appearing to fit into the specifications for the other classes. This can be seen in Chapter 3 during the annotation process, whereby additional rules and guidelines had to be implemented after discussions and intermediary analysis. This confusion within the models is then heightened given the large skew in distribution even from the starting labelled dataset. As explained by Ber and Haramaty (2020):

> *"In datasets characterized by overlapping classes, there are significant 'confusion effects' in which an increase in a certain label's probability results in the model estimating an increase in other labels' probabilities, even when the best possible models are used. These effects become even more dominant when classes are imbalanced since, given an overlap between a rare class and a prevalent class, models will learn to assign most of the probability from data points associated with the rare class to the prevalent class".*

Using this as a basis alongside the distribution analysis undertaken, it suggests that both the overlap in classes alongside the imbalanced dataset are some of the main contributions for the errors produced by the model. Even the highest performing model in Experiment 13 assigns the instances from the minority classes to those within the majority and confuses labels with overlapping requirements, such as occupations within SOC label 9, the *Installation, Maintenance and Repair* occupations and SOC label 10, the *Construction and Extraction* class. Although the same experiments could be ran by transforming the SOC labels to the higher level of aggregation with only 6 classes, this level is not practical for industry and for further analyses of occupational classes as there would be *too* much overlap between classes. It could be thus said that perhaps a more fine-grained schema with *additional* classes may help to delineate the fuzzy, overlapping boundaries, since they will have more specific and tailored requirements for each, even if some still overlap. The lower level of aggregation of SOC could thus be used. Of course, this introduction of more classes would inevitably require an extensively larger number of labelled instances in order to have a substantial amount within each class. On the other hand, to avoid these overlapping of classes and avoid further expensive and time-consuming annotations, it could be better to build a binary one verses all classifier. This could also potentially help to avoid the overlapping classes with have fuzzy boundaries and improve the final results as a whole.

Since an unpremeditated number of models were run for the SVM and all experiments are run using this unique dataset, it must be noted that the results cannot not representative of all ML and DL models, nor this semi-supervised pseudo-labelling approach in general. Nevertheless, from this investigation it can be noted that ultimately the unlabelled data can be leveraged to some extent to improve the decision boundary of a model and ultimately heighten its performance, albeit often somewhat minimal.

The compared results of using the extracted terms against those without do confirm the hypothesis in this semi-supervised setting, despite them in fact worsening the performance of the SVMs in a supervised environment. However, despite both the Skill and Education terms combined improving the results, the error analysis indicates that there is still a lot of noise incorporated into the models and therefore further filtering and cleaning of the skills needs to be undertaken, in order for them to have a greater and more positive impact. A detailed custom made stopword list could thus be composed to aid with this process. In addition to this, although the TF-IDF representations do seemingly help improve the final F1 score within each of the SVM models, this type of representation does not necessarily take into consideration the *semantics* of the text. Therefore it is unlikely that results will improve in future works using this representation in combination with other features. The FastText document embeddings however do capture the semantics necessary for classifying the occupations with a high accuracy, despite their sub-word matrix capacities, showing their benefits particularly within the CNN and Ensemble Experiments. This is not surprising since it uses the embeddings of subwords and thus works well with the German compound words. Moreover, it also has the capability to handle out-of-vocabulary (OOV) tokens which may arise due to the abundance of specific terminology used within each occupational class.

The complexity and specificity of the language used not only being inherent to the domain of Human Resources (HR), but also with each occupational class having its own terminology. For example, terms such as “DNI”, “AKP” and “HF” are all related to the medical domain however without specific domain knowledge, they are not neces-

sarily recognisable as skills by the human eye let alone with the FastText embeddings; particularly as a result of not being transformed from the abbreviated forms. Consequently, given the fact that there are in fact a large number of specialised occupations with their own specific terminology used within each one, it can be noted that potentially results would have further increased if the embeddings had been trained on this particular corpora.

Through analysis of the predictions, we can also see that further NLP preprocessing is needed for implementation, particularly with regards to lemmatisation. As a result of false NLP, or specifically POS, tagging, some words are still not lemmatised correctly. Mostly, we can see here that these are in fact the common compound words found in German; it could thus be suggested to perform some type of compound splitting in future works.

Finally, I do not analyse the extent of the impact with regards to each of the extracted classes; rather focus on combining the class requirements together and analysis the models both with and without their incorporation. Although it can be speculated that there is more noise introduced with the incorporation of skills, given the larger number of them, it is not certain. The same experiments could also thus be run using the Education and Skill terms as features *individually* in order to determine exactly what impact each of the extracted term classes has on the model in this semi-supervised setting.

Chapter 7

Conclusion

The aim of this thesis was to explore to what extent the automatic classification of occupations in a semi-supervised setting could be improved with the incorporation of specific occupational requirement classes, namely related to skills and education types. In order to address this goal, my research was split into three core components namely an Annotation Framework used to create the entire labelled dataset, a Term Extraction System for extracting the relevant requirements and finally an Occupation Classification architecture for specifically classifying the job advertisements into the occupational classes.

Using the carefully curated guidelines as a basis, both myself and an expert in the field of HR manually labelled a total of 6,052 instances from the entire dataset of 375,493 job advertisements, to be used as the guidance for the classification system. Following this component, I implement a distantly supervised pattern-learning architecture for extracting the relevant skill and education related terms, using seed lists from the ontology provided by Greple GmbH. From initialising the Term Extraction system for the Education class using just 12 seed terms and 10 manually crafted patterns, a total of 135 new terms are extracted. Regarding the Skill class, the results demonstrate that the input of 1,910 seed terms and 11 manually crafted patterns manage to extract a total of 17,902 new terms. Finally, a total of 14 experiments are ran for classifying the occupations within job advertisements. Here, three main comparisons are undertaken across each of the experiments: (1) Between two specific machine learning (ML) models, namely Support Vector Machine (SVM) and a Convolutional Neural Network (CNN), (2) Between two different text representations, Term Frequency Inverse Document Frequency (TF-IDF) and FastText word embeddings and finally (3) Between the incorporation of the extracted terms as features and without them.

In this research, only the intermediate level of aggregation within the Standard Occupation Classification (SOC) schema at 13 occupational classes is utilised, yet the most granular level of aggregation has 867 different occupational classes. Given the results even just using this level alone, it is evident that the classifying of occupations using job postings proves to be an extremely non-trivial task and certainly one which has multiple different layers in complexity. Although potentially using a higher level of aggregation with fewer classes may in fact improve the classification results, this would lead to more overlap between the categories and thus mean less fine-grain analyses could be undertaken thereafter. For example, the higher level is SOC groups *Management,*

Business, Financial, Computer, Engineering, Science Education, Legal, Community Service, Arts, Media, Healthcare Practitioners and Technical occupations all into the very first class. As shown in the results from the Term Extraction, although perhaps unsurprisingly a large proportion of the same *soft* skills exist in all classes, there is still an overlap of *technical* skills and education levels required for some classes, causing *fuzzy* boundaries. Therefore, using a more fine-grained schema could suggestively improve results.

Using this as a basis, I now address each of the three main research questions:

1. *To what extent can the ontology based extracted features aid with the classification of occupations?*

Based on the highest performing experiment, namely Experiment 13 which uses an Ensemble approach, FastText and the Skill and Education terms as features, it can be seen that the extracted terms do show improvements and contribute to classifying the occupations. With the same parameters however without the extracted features in Experiment 10, the model's performance drops from 0.62 to 0.50, thus confirming the first hypothesis. However, despite their contribution, it is clear from the Term extraction system results that a more carefully selected sample of seed data should be selected, particularly if the system does not have access to the meta-data contained within the ontology. Despite the ontology inevitably containing information in a structured and systematic way, some seeds appear misleading without respective knowledge needed for their contextual semantics. It is evident from the results that even further filtering and processing of the extracted terms should be undertaken, to avert the incorporation of noisy terms as well as avoid confusion between overlapping classes.

2. *Despite it being used in many state of the art NLP-related tasks, how does Deep Learning (DL) compare to Machine Learning (ML) for this task?*

Regarding the second research question, it is clear that ultimately DL does not necessarily outperform the more traditional ML with regards to this type of text classification problem, given the unique dataset. That being said, DL does show to have potential for growth particularly since its performance positively increases with the size of the data, however it is unknown just how well it would fair in even later iterations with more data. Traditional ML, on the other hand, evidently is capable of learning and understanding the relationships between features and patterns within classes within the smaller datasets, which the CNN in this research is evidently unable to do. With the incorporation of the FastText text representation, the traditional ML in the form of the SVM is even able to achieve the second highest result at 0.61 showing it does have the capability to continue this high level of understanding with the increase of data. However, given the extensive amount of time needed to run the SVM with the increasing amount of data, with a single iteration within the architecture taking up to 4 days, it is not functional enough in an industrial setting. Although distinct comparison cannot be determined between the two approaches given the number of iterations run, it can be said that traditional ML does achieve higher results with smaller datasets. For this reason, it would seem clear why the implementation of the Ensemble approach combining the SVM at the first iteration and the CNN for the remaining iterations obtains the highest result; with the inclusion of the extracted terms, it should be added.

3. *Given the complexity of implementing a semi-supervised approach using*

> *only a limited number of gold labelled instances, to what degree will the pseudo-labelled data not only change the decision boundary between classes, but also improve a given system's performance.*

The final research question can be answered by the results of the Occupation Classification system itself, whereby they positively confirm that in utilising a semi-supervised approach and leveraging a large number of unlabelled data, the overall performance of a given model does, in most cases, increase, albeit often somewhat minimal. With regards to the DL model, namely the CNN, the increase in training instances substantially improves the performance of the system, when compared in a supervised setting. However this is not the case for the traditional ML, whereby the score decreases for the SVM. Moreover, since the data is highly imbalanced with a skewed number of classes to instances ratio, it goes without saying that in future works some type of data augmentation should be undertaken. Thus, particular attention should thus be paid to the distribution of the classes and where highly disproportionate, techniques should be undertaken to balance it out. The impact of the imbalanced dataset led to the models often only predicting the majority classes and thus the proportion of the distribution is spread out further as the iterations increased. At least having a strong first few models using the more traditional ML, particularly with calibration, proved to ascertain the most promising results using a smaller dataset. Whether this be Upsampling, or if using an ensemble method with a ML model at the start, then even a Downsampling approach may deem more suitable; with the ML model implemented in this research seemingly performing efficiently with a smaller dataset. Although the confidence probabilities generated by the models do not necessarily become *more* confident as more labelled data is used, the model does adapt its confidence accordingly and therefore the hypothesised better interactive threshold does appear to be most suitable.

For future work, this research could be expanded through the incorporation of training domain specific embeddings using the entirety of the corpora available, in order to ascertain a better coverage of the language used in this type of data. Furthermore, the alternative pseudo-labelling architecture could be implemented given the size of the dataset, whereby the model from each previous iteration would be *re-trained* on the new pseudo-labelled data each time, potentially leading to increased pattern learning between the classes and an overall improved performance.

Ultimately, it can be concluded that the overall results are very encouraging. Given the complexity of not only the task of occupation classification itself, but the whole process in creating specific annotation guidelines, the development of a Term Extraction system as well as the implementation of the semi-supervised pseudo-labelling architecture, the highest result of 0.62 is highly promising. It can be expected that a deeper data analysis, preprocessing and feature engineering process would however lead to improved results, given that the performance of the majority of models improved with the introduction of the extracted terms. Whether these be linguistic or statistically derived, is uncertain.

**

Appendix A

Methodology: Annotation Guidelines

A.1 Overview

The 2018 Standard Occupational Classification (SOC) is a coding system established to classify all occupations into respective occupational branches. It uses a hierarchical structure, comprised with a total of 6 tiers: high level (6 branches); intermediate level (13); major groups (23); Minor groups (98); Broad occupations (459), and Detailed occupations(867). For the purpose of this thesis and this annotation framework, the task is to use the **intermediate level of aggregation** as a basis to classify and label occupations into the respective 13 branches/classes.

A.2 Annotation Process

Step 1: Familiarisation

Prior to starting the practical annotations, the annotator should ensure they are familiar with the annotation concepts (both labels and their contents based on the SOC structure). For further information and clarity about the SOC structure, the SOC User Guide may be used as a reference [1] (particularly pages 10 through to 17)

Step 2: Labelling of the Data

Once familiarity is established, the annotator can begin practical work. Further information regarding the data for annotation is described in *section* A.4.

It is recommended to use the **O*Net SOC Crosswalk** [2], *Occupation Quick Search* to aid the annotation process, which groups occupations in the Occupational Information Network (O*Net) database to the SOC structure. As a result of using this Crosswalk tool, specific vacancy titles can be searched for and suggested classification branches will be shown.

[1] `https://www.bls.gov/soc/2018/soc_2018_user_guide.pdf`
[2] `https://www.onetonline.org/crosswalk/SOC/`

Step 3: Agreement

There will be multiple intermediary discussions and analyses throughout the annotation process between the annotators. The precise deadlines and time constraints will be finalised between the annotators together. Once all of the annotations have been undertaken by both annotators, the "inter-annotator" agreement will be calculated; measuring the degree in which the annotations across experts align.

A.3 Annotation Types

For the purpose of this framework, there are **two** annotation types which should be inputted for each instance; both a **label** and **confidence** score.

A.3.1 Label

The **13** annotation labels used in this project are listed in *Table* 1 in the "soc_label" column, alongside both an "included groups" and "name" column. A short description of each of the three columns is given below:

1. **soc_label**: used as a reference to the branch name and should be used as the **annotation labels**.

2. **Included Groups**: refer to the SOC occupation codes included within each branch; each occupation in SOC is assigned a 6 digit code, with the first two digits referring to it's respective major group branch number (see *Figure* A.1). For examples see *Section* A.3.1.1.

3. **Name**: Formal title of branch, as assigned by SOC for this level of aggregation.

soc_label	**Included Groups**	**Name (+ "Occupations")**
1	11-13	Management, Business, and Financial
2	15-19	Computer, Engineering, and Science
3	21-27	Education, Legal, Community Service, Arts, and Media
4	29	Healthcare Practitioners and Technical
5	31-39	Service
6	41	Sales and Related
7	43	Office and Administrative Support
8	45	Farming, Fishing, and Forestry
9	47	Construction and Extraction
10	49	Installation, Maintenance, and Repair
11	51	Production
12	53	Transportation and Material Moving
13	55	Military Specific

Table A.1: Overview of SOC category names, the included SOC group numbers for each branch alongside label reference to be used for annotations.

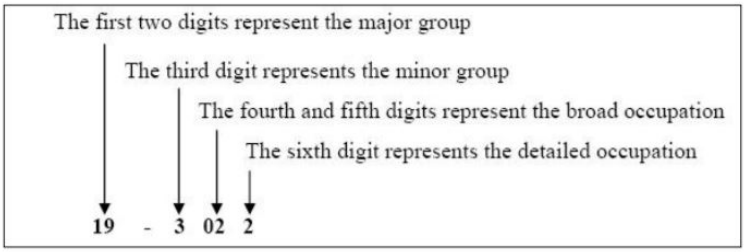


Figure A.1: SOC Occupation Code: Structure Breakdown

A.3.1.1 Determining SOC label: O*Net SOC Crosswalk

As mentioned in step 2 of section A.2, the O*Net SOC Crosswalk *Quick Search* should be used to aid the annotation process (`https://www.onetonline.org/crosswalk/SOC/`). Each occupation in this system is assigned a 6 digit code, with an exemplar breakdown shown in *Figure* A.1. The annotator should focus only on the *first two digits*, representing the major group, to determine which soc_label the occupation should fall into, based on it's included groups. It should be noted that the O*Net SOC Crosswalk can only be used in English, therefore the appropriate English equivalent should be searched.

A.3.1.2 Example

1. Occupation: **Klempner** (Plumber)
   SOC 6-digit code: **47**-2152.02
   Group: 47
   soc_label: 9
   soc_name: *Construction and Extraction Occupations*
   O*Net Soc Crosswalk reference link: `https://www.onetonline.org/link/summary/47-2152.02`

A.3.2 Confidence

In order to measure the scale of certainty, a confidence scale is established for labelling the instances. For each annotated item, a *confidence* score should also always be assigned, in the respective column. The score should correlate to an integer on the scale, ranging from 1-3, as shown in Figure A.2 below. A brief description for each label is given below, however it should be noted that full guidelines here have been purposefully omitted, to allow for a certain level of subjectivity.

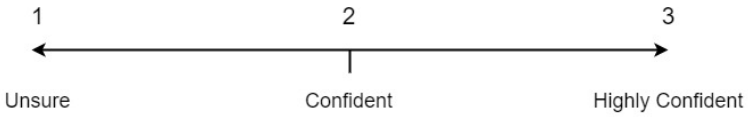


Figure A.2: Confidence Scale

(**1**) *Unsure:* The annotator is somewhat sure however also somewhat doubtful. Both levels of confidence and doubt are equal.

(**2**) *Confident:* The annotator is somewhat doubtful, however confidence exceeds doubt.

(**3**) *Highly Confident:* The annotator is sure that the instance should be classified to the
assigned label and has no form of doubt.

If there is too little information for the instance to be annotated at all, potentially because of a lack of information in the vacancy description, a score of **nv** should be assigned to *both* the *label* and *confidence* columns.

A.4 The Data

The data for annotation is comprised of 4526 instances, based on job vacancy data provided by Greple GmbH. The data is represented in a csv file format, with a total of 6 columns; (1)_id, (2)_vacancy_title, (3)soc_label, (4)confidence, (5)_soc_name and (6)_vacancy_description (see *Table* A.4).

The ID refers to the unique identifier assigned to the given occupation vacancy as according to ElasticSearch (the database) with an additional prefix of two letters, referring to its data source (e.g. "st" refers to "stepstone"). Whilst the ID can be used as a reference, the *_vacancy_title* and *_vacancy_description* should be used as the core references for annotation.

As mentioned in the previous section, the two columns for annotation are the **soc_label** and **confidence**; the *_soc_name* will *automatically populate* itself once the label has been inputted and can be used as an additional checking point to avoid mix-ups. *Note:* for further clarity, **only column titles that do not begin with "_", should be directly filled in by the annotator.**

_id	_vacancy_title	soc_label	confidence	_soc_name	_vacancy_description
st_45372	Volljurist(m/d)				Wir suchen [...]

Figure A.3: Format of Annotation Data Template with instance *empty*

_id	_vacancy_title	soc_label	confidence	_soc_name	_vacancy_description
st_45372	Volljurist(m/d)	**3**	**3**	Education, Legal[...]	Wir suchen [...]

Figure A.4: Format of Annotation Data with instance *complete*

A.4.1 Vacancy title

The vacancy title refers to the title as assigned by the vacancy job poster. The vacancy title alone can, in some cases, be enough information to be able to annotate the instance into one of the given branches.

A.4.1.1 Examples

1. vacancy_title: **Softwareentwickler**, soc_label:**2**
   - soc_name: *Computer, Engineering, and Science Occupations*

-O*Net Soc Crosswalk: `https://www.onetonline.org/link/summary/15-1133.00`

2. vacancy_title: **Geschaeftsfuehrer**, soc_label:**1**
   - soc_name: *Management, Business, and Financial*
   -O*Net Soc Crosswalk: `https://www.onetonline.org/link/summary/11-1011.00`

3. vacancy_title: **Koch**, soc_label:**5**
   - soc_name: *Service Occupations*
   -O*Net Soc Crosswalk: `https://www.onetonline.org/link/summary/35-2014.00`

A.4.2 Vacancy Description

In cases where the vacancy title alone is not enough information for the instance to be labelled, perhaps because of ambiguity or lack of detail, the vacancy description itself should be used as further reference for information.

A.4.2.1 Examples

1. vacancy_title: **Projektmanager**
   VacancyDescription: *"Wir suchen eine IT-Projektmanagerin / IT-Projektmanager Infrastruktur- und Digitalisierungsprojekte. Qualifikationen: abgeschlossenes Hochschulstudium vorzugsweise in einem IT- oder IT-nahen Studiengang (Diplom / Master), nachweisbare mehrjaehrige Berufserfahrung als Projektleiterin / Projektleiter im IT-Umfeld"*
   soc_label:**2**
   - soc_name: *Computer, Engineering, and Science Occupations*
   -O*Net Soc Crosswalk: `https://www.onetonline.org/link/summary/15-1199.09`

2. vacancy_title: **Projektmanager**
   VacancyDescription: *"Dafür brauchen wir einen Kundenberater(in) / Projektmanager(in). Aufgaben: Verwaltung der Kundenkonten. Qualifikationen: Kundenorientierung und betriebswirtschaftliches Denken, Erfahrung im Kundenservice, Kommunikatives und freundliches Wesen*
   soc_label:**7**
   - soc_name: *Office and Administrative Support Occupations*
   -O*Net Soc Crosswalk: `https://www.onetonline.org/link/summary/43-4051.00`

3. vacancy_title: **Mitarbeiter Hotspot Wildfluesse**
   VacancyDescription: *"Aufgaben: Unterstuetzung in der administrativen Projektverwaltung, Mitwirkung bei der Planung, Organisation und Bewerbung von Publikumsveranstaltungen, wie Flussfilmfest und Exkursionen. Qualifikationen: versierte MS-Office Kenntnisse"*
   soc_label:**7**
   - soc_name: *Office and Administrative Support Occupations*
   -O*Net Soc Crosswalk: `https://www.onetonline.org/link/summary/43-6014.00`

In the case where *both* the vacancy title and vacancy description *cannot* be classified into any of the branches due to a *lack of information* in either fields, a label of **nv** (nicht verfuegbar / not available) should be assigned.

A.5 Important Coding Guidelines offered by SOC

"The following SOC coding guidelines are intended to assist users in consistently assigning SOC codes and titles to survey responses and in other coding activities.

1. A worker should be assigned to an SOC occupation code based on work performed.
2. When workers in a single job could be coded in more than one occupation, they should be coded in the occupation that requires the highest level of skill. If there is no measurable difference in skill requirements, workers should be coded in the occupation in which they spend the most time. Workers whose job is to teach at different levels (e.g., elementary,middle, or secondary) should be coded in the occupation corresponding to the highest educational level they teach.
3. Data collection and reporting agencies should assign workers to the most detailed occupation possible. Different agencies may use different levels of aggregation, depending on their ability to collect data.
4. Workers who perform activities not described in any distinct detailed occupation in the SOC structure should be coded in an appropriate "All Other" occupation. These occupations appear as the last occupation in a group with a code ending in "9" and are identified by having the words "All Other" appear at the end of the title.
5. Workers in Major Groups 33–0000 through 53–0000 who spend 80 percent or more of their time performing supervisory activities are coded in the appropriate first-line supervisor category in the SOC. In these same Major Groups (33–0000 through 53– 0000), persons with supervisory duties who spend less than 80 percent of their time supervising are coded with the workers they supervise.
6. Licensed and non-licensed workers performing the same work should be coded together in the same detailed occupation, except where specified otherwise in the SOC definition. " [3]

[3] `https://www.bls.gov/soc/2018/soc_2018_user_guide.pdf`

Appendix B

Methodology: Preprocessing Gazetteers

```
(m/w/d)
(m/d/w)
(w/m/d)
(w/d/m)
(d/m/w)
(d/w/m)
(d/f/m)
(d/m/f)
(f/d/m)
(f/m/d)
(m/f/d)
(m/d/f)
(m/w)
(w/m)
(m/f)
(f/m)
(m/w/x)
(m/x/w)
(x/m/w)
(x/w/m)
(w/x/m)
(w/m/x)
(m/d)
```

Listing 4: Gazetter of Gender forms to be stripped

```
{
  "ms":"Microsoft",
  "js":"Javascript",
  "r&d": "Research and Development",
  "paed": "paedagogiae",
  "edv": "Elektronische Daten Verarbeitung",
  "eva": "Eingabe Verarbeitung Ausgabe"
}
```

Listing 5: Gazetter of Skill related Abbreviations

```
{
    "sr":"senior",
    "sr.": "senior",
    "jr":"junior",
    "jr.": "junior",
    "dipl": "diploma",
    "dipl.": "diploma",
    "dipl.-ing": "Diplom-Ingenieur",
    "dr": "doktor",
    "prof": "Professor",
    "prakt": "Praktikant",
    "intern": "Intern",
    "ceo": "Chief Executive Officer",
    "cfo": "Chief Financial Officer",
    "coo": "Chief Operating Officer",
    "cio": "Chief Information Officer",
    "cmo": "Chief Marketing Officer",
    "cto": "Chief Technical Officer",
    "cta": "Chief Technical Architect",
    "svp": "Senior Vice President",
    "vp": "Vice President",
    "admin": "Administrator",
    "hr": "Human Resources",
    "pr": "Public Relations",
    "md": "Managing Director",
    "pdms": "Plant Design Management System",
    "bta": "Business Technology Administrator",
    "r&d": "Research and Development",
    "bdo": "Business Development Officer",
    "cra": "Clinical Research Associate",
    "pm": "Projekt Management",
    "pmo": "Projekt Management Office",
    "pmp": "Project Management Professional",
    "qa": "Quality Assurance",
    "qsc": "Quality Service Certificate",
    "cso": "Chief Strategy Officer",
    "mvo": "Motor Vehicle Operator",
    "qm": "QualitÃtsmanagement",
    "kfm": "Kaufmann",
    "erp": "Enterprise Resource Planning",
    "dcs": "Deputy Chief of Staff",
    "cse": "Computer Science Engineer",
    "lmt": "Licensed Massage Therapist",
    "tcms": "Train Control Monitoring System",
    "ipm": "Integrated Pest Management",
    "seo": "Search Engine Optimization",
    "ddd": "Domain Driven Design",
    "cpa": "Certified Public Accountant",
    "eto": "Electro Technological Officer",
    "ksk": "Kommando Spezialkraefte",
    "kam": "Key Account Manager",
    "IT": "Informatik"
}
```

Listing 6: Gazetter of Job Title related Abbreviations

```
{
  "uni": "Universität",
  "tu": "Technische Universität",
  "th": "Technische Hochschule",
  "ph": "Pädagogische Hochschule",
  "fh": "Fachhochschule",
  "ba": "Bachelor of Arts",
  "ba.": "Bachelor of Arts",
  "b.a": "Bachelor of Arts",
  "b.a.": "Bachelor of Arts",
  "ba.": "Bachelor of Arts",
  "bsc": "Bachelor of Science",
  "b.s.c": "Bachelor of Science",
  "b.s.c.": "Bachelor of Science",
  "bsc.": "Bachelor of Science",
  "beng": "Bachelor of Engineering",
  "beng.": "Bachelor of Engineering",
  "b.eng.": "Bachelor of Engineering",
  "llm": "Master of Laws",
  "llm.": "Master of Laws",
  "l.l.m": "Master of Laws",
  "ma": "Master of Arts",
  "m.a": "Master of Arts",
  "ma.": "Master of Arts",
  "msc": "Master of Science",
  "msc.": "Master of Science",
  "m.s.c": "Master of Science",
  "m.s.c.": "Master of Science",
  "mba": "Master of Business Administration",
  "m.b.a.": "Master of Business Administration",
  "mba.": "Master of Business Administration",
  "m.b.a": "Master of Business Administration",
  "hd": "Hochschuldozent",
  "hd.": "Hochschuldozent",
  "h.d.": "Hochschuldozent",
  "phd": "Doctor of Philosophy",
  "phd.": "Doctor of Philosophy",
  "p.h.d.": "Doctor of Philosophy",
  "p.h.d": "Doctor of Philosophy",
  "diplom": "diploma",
  "diplom.": "diploma"
}
```

Listing 7: Gazetter of Education related Abbreviations

```
{
  "bzw": "beziehungsweise",
  "bzw.": "beziehungsweise",
  "b.z.w": "beziehungsweise",
  "zb": "zum Beispiel",
  "zb.": "zum Beispiel",
  "z.b": "zum Beispiel",
  "etw": "etwas",
  "etw.": "etwas",
  "beisp": "beispielsweise",
  "beisp.": "beispielsweise",
  "dazw": "dazwischen",
  "dazw.": "dazwischen",
  "kompl": "komplett",
  "kompl.": "komplett",
  "tel": "Telefon",
  "tel.": "Telefon",
  "usw": "und so weiter",
  "usw.": "und so weiter",
  "u.s.w": "und so weiter",
  "inkl": "inklusiv",
  "inkl.": "inklusiv",
  "zhd": "zuhanden",
  "z.hd": "zuhanden",
  "z.h.d": "zuhanden",
  "zhd.": "zuhanden",
  "nr": "Nummer",
  "nr.": "Nummer",
  "pkw": "Personenkraftwagen",
  "p.k.w": "Personenkraftwagen",
  "pkw.": "Personenkraftwagen",
  "b2b": "Business to Business",
  "eidg": "Emissions Inventory Database Group",
  "R&D": "Research and Development",
  "1st": "First",
  "2nd": "Second"
}
```

Listing 8: Gazetter of Miscellaneous Abbreviations

Appendix C

Term Extraction Results: Education

Appendix D

EDA: Most Frequent Bigram Barplots

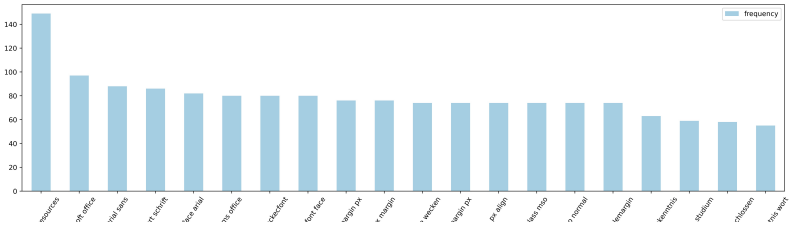


Figure D.1: Bar Plot illustrating top 20 Bi-grams within soc label 1, "Office and Administrative"

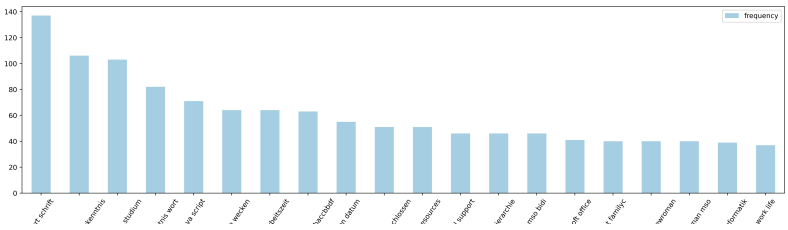


Figure D.2: Bar Plot illustrating top 20 Bi-grams within soc label 2, "Office and Administrative"

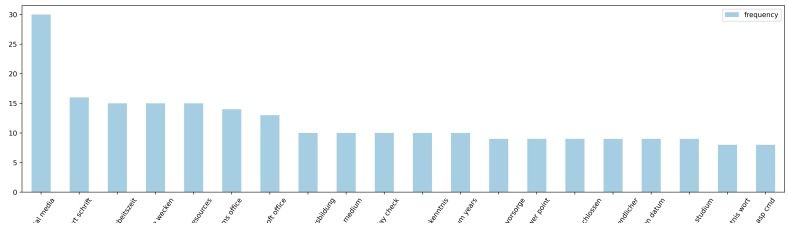


Figure D.3: Bar Plot illustrating top 20 Bi-grams within soc label 3, "Office and Administrative"

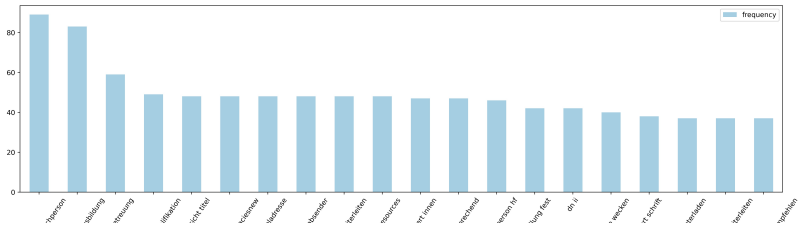


Figure D.4: Bar Plot illustrating top 20 Bi-grams within soc label 4, "Office and Administrative"

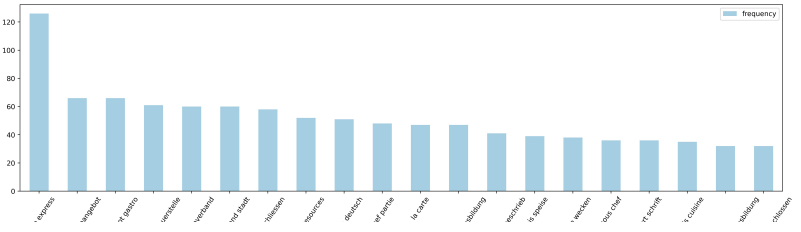


Figure D.5: Bar Plot illustrating top 20 Bi-grams within soc label 5, "Office and Administrative"

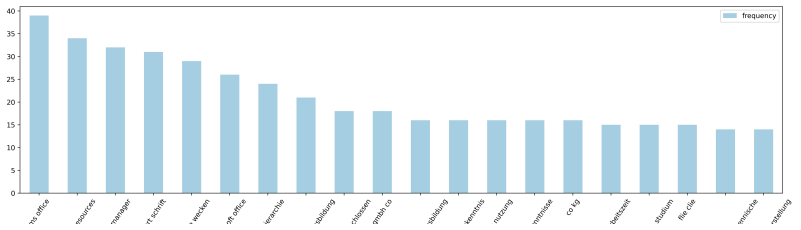


Figure D.6: Bar Plot illustrating top 20 Bi-grams within soc label 6, "Office and Administrative"

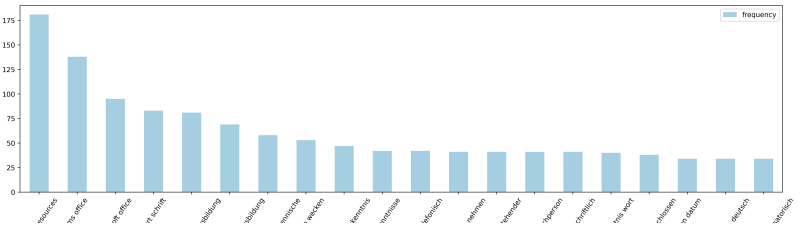


Figure D.7: Bar Plot illustrating top 20 Bi-grams within soc label 7, "Office and Administrative"

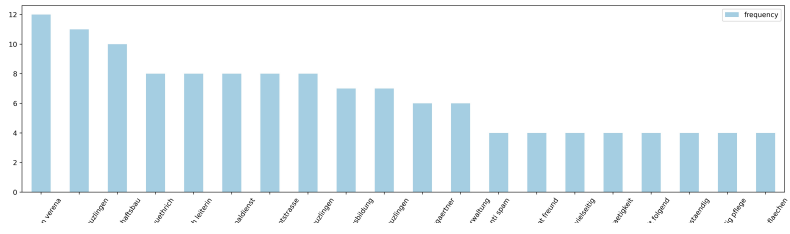


Figure D.8: Bar Plot illustrating top 20 Bi-grams within soc label 8, "Office and Administrative"

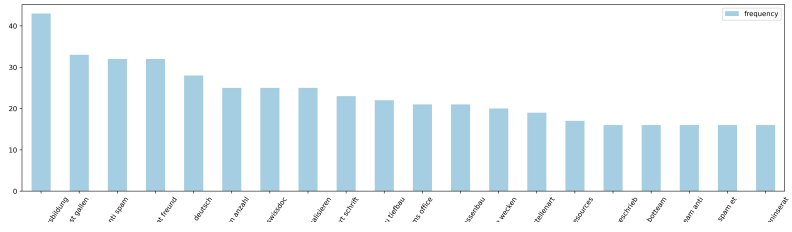


Figure D.9: Bar Plot illustrating top 20 Bi-grams within soc label 9, "Office and Administrative"

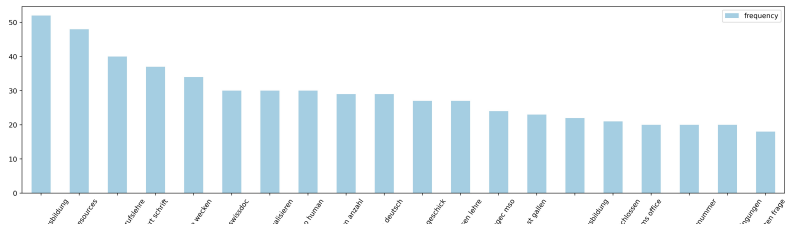


Figure D.10: Bar Plot illustrating top 20 Bi-grams within soc label 10, "Office and Administrative"

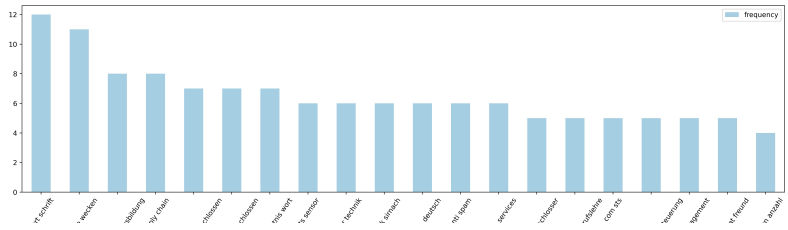


Figure D.11: Bar Plot illustrating top 20 Bi-grams within soc label 11, "Office and Administrative"

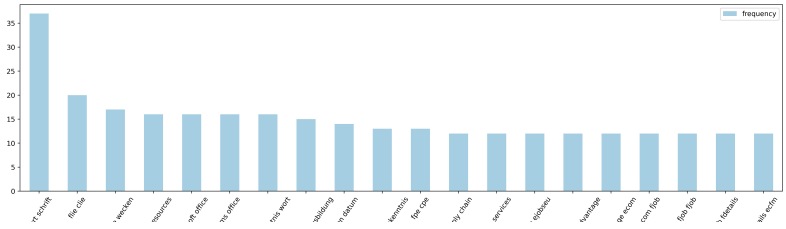


Figure D.12: Bar Plot illustrating top 20 Bi-grams within soc label 12, "Office and Administrative"

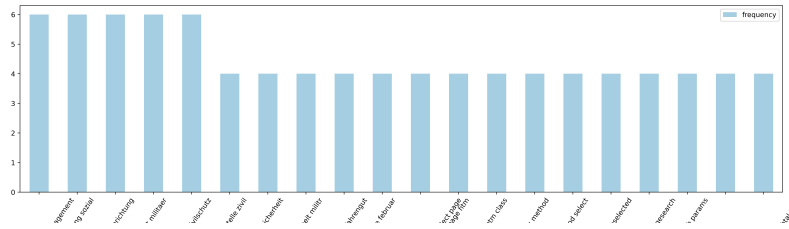


Figure D.13: Bar Plot illustrating top 20 Bi-grams within soc label 13, "Office and Administrative"

Appendix E

Results: Classification Reports

E.1 Supervised Results

Classification Report

	precision	recall	f1-score	support
1	0.54	0.77	0.64	184
2	0.69	0.87	0.77	198
3	0.83	0.39	0.53	49
4	0.84	0.61	0.71	120
5	0.85	0.67	0.75	127
6	0.80	0.61	0.69	85
7	0.67	0.73	0.70	162
8	0.86	0.75	0.80	8
9	0.72	0.61	0.66	92
10	0.65	0.74	0.69	122
11	1.00	0.12	0.21	25
12	0.64	0.45	0.53	55
13	1.00	0.25	0.40	4
accuracy			0.68	1231
macro avg	0.78	0.58	0.62	1231
weighted avg	0.71	0.68	0.68	1231

Calib_SVM + TF-IDF

Classification Report

	precision	recall	f1-score	support
0	0.33	0.24	0.28	184
1	0.87	0.46	0.61	198
2	0.01	0.02	0.01	49
3	0.17	0.30	0.22	120
4	0.24	0.60	0.35	127
5	0.00	0.00	0.00	85
6	0.25	0.27	0.26	162
7	0.00	0.00	0.00	8
8	0.00	0.00	0.00	92
9	0.12	0.20	0.15	122
10	0.00	0.00	0.00	25
11	0.00	0.00	0.00	55
12	0.00	0.00	0.00	4
accuracy			0.26	1231
macro avg	0.15	0.16	0.14	1231
weighted avg	0.28	0.26	0.25	1231

CNN Raw

Classification Report

	precision	recall	f1-score	support
1	0.45	0.83	0.58	184
2	0.66	0.87	0.75	198
3	1.00	0.12	0.22	49
4	0.92	0.58	0.71	120
5	0.88	0.64	0.74	127
6	0.87	0.46	0.60	85
7	0.66	0.72	0.69	162
8	1.00	0.25	0.40	8
9	0.75	0.60	0.67	92
10	0.65	0.72	0.68	122
11	0.00	0.00	0.00	25
12	0.82	0.33	0.47	55
13	1.00	0.25	0.40	4
accuracy			0.65	1231
macro avg	0.74	0.49	0.53	1231
weighted avg	0.71	0.65	0.64	1231

Non-calib_SVM + TF-IDF

Classification Report

	precision	recall	f1-score	support
1	0.53	0.66	0.59	184
2	0.73	0.75	0.74	198
3	0.78	0.14	0.24	49
4	0.72	0.57	0.64	120
5	0.75	0.63	0.69	127
6	0.71	0.34	0.46	85
7	0.47	0.68	0.56	162
8	0.00	0.00	0.00	8
9	0.60	0.47	0.52	92
10	0.44	0.78	0.57	122
11	0.00	0.00	0.00	25
12	0.71	0.31	0.43	55
13	0.00	0.00	0.00	4
accuracy			0.59	1231
macro avg	0.49	0.41	0.42	1231
weighted avg	0.61	0.59	0.57	1231

Non-calib_SVM + FastText

Classification Report

	precision	recall	f1-score	support
1	0.54	0.65	0.59	184
2	0.72	0.75	0.74	198
3	0.64	0.18	0.29	49
4	0.67	0.57	0.62	120
5	0.66	0.64	0.65	127
6	0.70	0.35	0.47	85
7	0.49	0.67	0.57	162
8	0.00	0.00	0.00	8
9	0.56	0.51	0.53	92
10	0.49	0.73	0.59	122
11	0.40	0.08	0.13	25
12	0.62	0.33	0.43	55
13	0.00	0.00	0.00	4
accuracy			0.59	1231
macro avg	0.50	0.42	0.43	1231
weighted avg	0.59	0.59	0.57	1231

Calib_SVM + FastText

Classification Report

	precision	recall	f1-score	support
0	0.32	0.45	0.37	184
1	0.60	0.69	0.64	198
2	0.00	0.00	0.00	49
3	0.64	0.54	0.59	120
4	0.59	0.64	0.61	127
5	0.06	0.01	0.02	85
6	0.41	0.44	0.43	162
7	0.00	0.00	0.00	8
8	0.21	0.15	0.18	92
9	0.25	0.50	0.34	122
10	0.00	0.00	0.00	25
11	0.00	0.00	0.00	55
12	0.00	0.00	0.00	4
accuracy			0.42	1231
macro avg	0.24	0.26	0.24	1231
weighted avg	0.37	0.42	0.38	1231

CNN + FastText

E.2 Semi-supervised Results

Classification Report

	precision	recall	f1-score	support
1	0.00	0.00	0.00	184
2	0.22	0.98	0.36	198
3	1.00	0.08	0.15	49
4	0.69	0.48	0.57	120
5	0.87	0.59	0.70	127
6	1.00	0.14	0.25	85
7	0.82	0.41	0.55	162
8	1.00	0.12	0.22	8
9	0.86	0.20	0.32	92
10	0.72	0.34	0.47	122
11	0.00	0.00	0.00	25
12	1.00	0.05	0.10	55
13	0.00	0.00	0.00	4
accuracy			0.38	1231
macro avg	0.63	0.26	0.28	1231
weighted avg	0.60	0.38	0.36	1231

(1) Experiment One
SVM + TF-IDF

Classification Report

	precision	recall	f1-score	support
1	0.70	0.41	0.52	184
2	0.64	0.82	0.72	198
3	0.44	0.39	0.41	49
4	0.57	0.62	0.59	120
5	0.50	0.66	0.57	127
6	0.49	0.48	0.49	85
7	0.52	0.65	0.58	162
8	0.00	0.00	0.00	8
9	0.62	0.45	0.52	92
10	0.56	0.61	0.58	122
11	0.00	0.00	0.00	25
12	0.59	0.40	0.48	55
13	1.00	0.25	0.40	4
accuracy			0.57	1231
macro avg	0.51	0.44	0.45	1231
weighted avg	0.57	0.57	0.55	1231

(2) Experiment Two
SVM + FastText

Classification Report

	precision	recall	f1-score	support
1	0.74	0.41	0.53	184
2	0.57	0.86	0.69	198
3	0.36	0.29	0.32	49
4	0.57	0.65	0.61	120
5	0.64	0.64	0.64	127
6	0.58	0.60	0.59	85
7	0.58	0.71	0.64	162
8	1.00	0.50	0.67	8
9	0.76	0.52	0.62	92
10	0.58	0.71	0.64	122
11	0.00	0.00	0.00	25
12	0.84	0.38	0.53	55
13	1.00	0.25	0.40	4
accuracy			0.61	1231
macro avg	0.63	0.50	0.53	1231
weighted avg	0.62	0.61	0.59	1231

(3) Experiment Three
SVM + TF-IDF + Terms

Classification Report

	precision	recall	f1-score	support
1	0.74	0.47	0.57	184
2	0.67	0.79	0.72	198
3	0.80	0.16	0.27	49
4	0.58	0.63	0.61	120
5	0.55	0.68	0.61	127
6	0.47	0.52	0.49	85
7	0.50	0.68	0.58	162
8	0.00	0.00	0.00	8
9	0.61	0.50	0.55	92
10	0.51	0.66	0.57	122
11	0.40	0.08	0.13	25
12	0.56	0.33	0.41	55
13	0.00	0.00	0.00	4
accuracy			0.58	1231
macro avg	0.49	0.42	0.42	1231
weighted avg	0.59	0.58	0.56	1231

(4) Experiment Four
SVM + FastText + Terms

Classification Report

	precision	recall	f1-score	support
0	0.28	0.30	0.29	184
1	0.72	0.52	0.60	198
2	0.00	0.00	0.00	49
3	0.00	0.00	0.00	120
4	0.19	0.38	0.25	127
5	0.00	0.00	0.00	85
6	0.25	0.44	0.32	162
7	0.00	0.00	0.00	8
8	0.00	0.00	0.00	92
9	0.26	0.77	0.39	122
10	0.00	0.00	0.00	25
11	0.00	0.00	0.00	55
12	0.00	0.00	0.00	4
accuracy			0.30	1231
macro avg	0.13	0.19	0.14	1231
weighted avg	0.24	0.30	0.25	1231

(5) Experiment Five
CNN Raw

Classification Report

	precision	recall	f1-score	support
0	0.48	0.56	0.52	184
1	0.62	0.73	0.67	198
2	0.00	0.00	0.00	49
3	0.35	0.60	0.44	120
4	0.26	0.75	0.39	127
5	0.00	0.00	0.00	85
6	0.43	0.55	0.49	162
7	0.00	0.00	0.00	8
8	0.00	0.00	0.00	92
9	0.18	0.02	0.03	122
10	0.00	0.00	0.00	25
11	0.00	0.00	0.00	55
12	0.00	0.00	0.00	4
accuracy			0.41	1231
macro avg	0.18	0.25	0.19	1231
weighted avg	0.31	0.41	0.33	1231

(6) Experiment Six
CNN + FastText

Classification Report

	precision	recall	f1-score	support
0	0.29	0.45	0.35	184
1	0.81	0.46	0.59	198
2	0.00	0.00	0.00	49
3	0.22	0.67	0.33	120
4	0.30	0.53	0.38	127
5	0.00	0.00	0.00	85
6	0.31	0.30	0.31	162
7	0.00	0.00	0.00	8
8	0.31	0.17	0.22	92
9	0.19	0.04	0.07	122
10	0.00	0.00	0.00	25
11	0.00	0.00	0.00	55
12	0.00	0.00	0.00	4
accuracy			0.32	1231
macro avg	0.19	0.20	0.17	1231
weighted avg	0.31	0.32	0.28	1231

(7) Experiment Seven
CNN Raw + Terms

Classification Report

	precision	recall	f1-score	support
0	0.49	0.49	0.49	184
1	0.53	0.77	0.63	198
2	0.00	0.00	0.00	49
3	0.52	0.64	0.57	120
4	0.31	0.72	0.43	127
5	0.25	0.02	0.04	85
6	0.37	0.59	0.46	162
7	0.00	0.00	0.00	8
8	0.19	0.05	0.08	92
9	0.40	0.05	0.09	122
10	0.00	0.00	0.00	25
11	0.00	0.00	0.00	55
12	0.00	0.00	0.00	4
accuracy			0.42	1231
macro avg	0.23	0.26	0.21	1231
weighted avg	0.36	0.42	0.35	1231

(8) Experiment Eight
CNN + FastText + Terms

Classification Report

	precision	recall	f1-score	support
0	0.64	0.08	0.14	184
1	0.73	0.75	0.74	198
2	0.33	0.06	0.10	49
3	0.75	0.58	0.66	120
4	0.78	0.65	0.71	127
5	0.19	0.64	0.30	85
6	0.58	0.78	0.66	162
7	1.00	0.50	0.67	8
8	0.70	0.47	0.56	92
9	0.48	0.80	0.60	122
10	0.00	0.00	0.00	25
11	0.46	0.29	0.36	55
12	0.00	0.00	0.00	4
accuracy			0.53	1231
macro avg	0.51	0.43	0.42	1231
weighted avg	0.60	0.53	0.51	1231

(9) Experiment Nine
Ensemble + TF-IDF

Classification Report

	precision	recall	f1-score	support
0	0.45	0.60	0.51	184
1	0.60	0.80	0.68	198
2	0.00	0.00	0.00	49
3	0.65	0.46	0.54	120
4	0.66	0.59	0.62	127
5	0.00	0.00	0.00	85
6	0.37	0.69	0.48	162
7	0.00	0.00	0.00	8
8	0.49	0.25	0.33	92
9	0.45	0.64	0.53	122
10	0.00	0.00	0.00	25
11	0.00	0.00	0.00	55
12	0.00	0.00	0.00	4
accuracy			0.50	1231
macro avg	0.28	0.31	0.28	1231
weighted avg	0.42	0.50	0.44	1231

(10) Experiment Ten
Ensemble + FastText

Classification Report

	precision	recall	f1-score	support
0	0.67	0.32	0.43	184
1	0.61	0.86	0.71	198
2	0.00	0.00	0.00	49
3	0.66	0.72	0.69	120
4	0.69	0.66	0.67	127
5	0.44	0.67	0.53	85
6	0.60	0.73	0.66	162
7	0.00	0.00	0.00	8
8	0.49	0.58	0.53	92
9	0.61	0.65	0.63	122
10	0.00	0.00	0.00	25
11	0.58	0.51	0.54	55
12	0.00	0.00	0.00	4
accuracy			0.60	1231
macro avg	0.41	0.44	0.42	1231
weighted avg	0.57	0.60	0.57	1231

(11) Experiment Eleven
Ensemble + TF-IDF + FastText

Classification Report

	precision	recall	f1-score	support
0	0.53	0.51	0.52	184
1	0.64	0.86	0.73	198
2	1.00	0.02	0.04	49
3	0.64	0.62	0.63	120
4	0.80	0.65	0.72	127
5	0.39	0.49	0.44	85
6	0.60	0.72	0.65	162
7	1.00	0.50	0.67	8
8	0.61	0.18	0.28	92
9	0.47	0.76	0.58	122
10	0.00	0.00	0.00	25
11	0.52	0.31	0.39	55
12	0.00	0.00	0.00	4
accuracy			0.58	1231
macro avg	0.55	0.43	0.43	1231
weighted avg	0.59	0.58	0.55	1231

(12) Experiment Twelve
Ensemble + TF-IDF + Terms

Classification Report

	precision	recall	f1-score	support
0	0.71	0.38	0.50	184
1	0.64	0.85	0.73	198
2	0.75	0.31	0.43	49
3	0.70	0.72	0.71	120
4	0.72	0.67	0.69	127
5	0.49	0.65	0.56	85
6	0.64	0.71	0.67	162
7	0.00	0.00	0.00	8
8	0.50	0.61	0.55	92
9	0.53	0.70	0.61	122
10	0.00	0.00	0.00	25
11	0.60	0.49	0.54	55
12	0.00	0.00	0.00	4
accuracy			0.62	1231
macro avg	0.48	0.47	0.46	1231
weighted avg	0.62	0.62	0.60	1231

(13) Experiment Thirteen
Ensemble + FastText + Terms

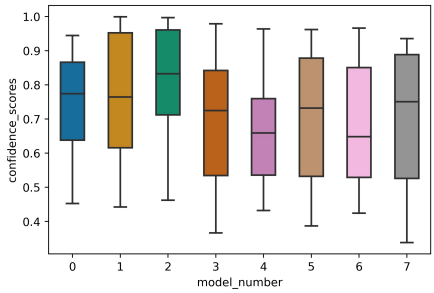
Classification Report

	precision	recall	f1-score	support
1	0.74	0.41	0.53	184
2	0.57	0.86	0.69	198
3	0.36	0.29	0.32	49
4	0.57	0.65	0.61	120
5	0.64	0.64	0.64	127
6	0.58	0.60	0.59	85
7	0.58	0.71	0.64	162
8	1.00	0.50	0.67	8
9	0.76	0.52	0.62	92
10	0.58	0.71	0.64	122
11	0.00	0.00	0.00	25
12	0.84	0.38	0.53	55
13	1.00	0.25	0.40	4
accuracy			0.61	1231
macro avg	0.63	0.50	0.53	1231
weighted avg	0.62	0.61	0.59	1231

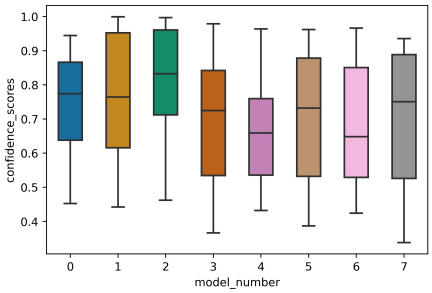
(14) Experiment Fourteen
Ensemble + TF-IDF + FastText + Terms

Appendix F

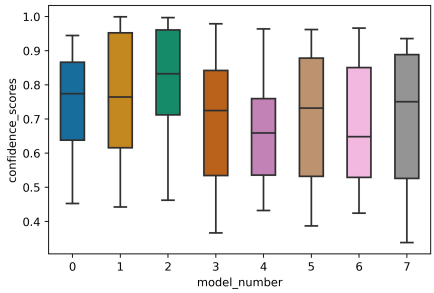
Results: Confidence Intervals



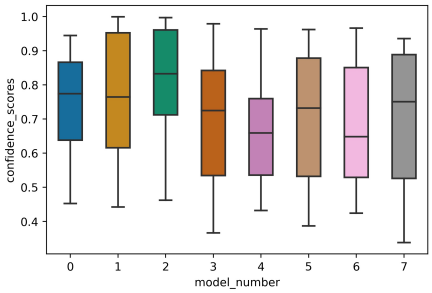
(1) Experiment One
SVM Raw



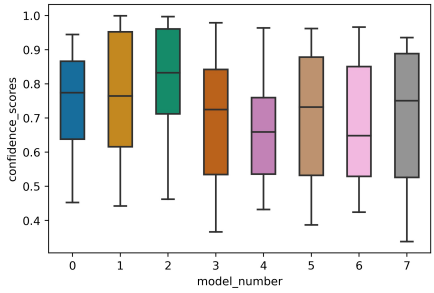
(2) Experiment Two
CNN Raw



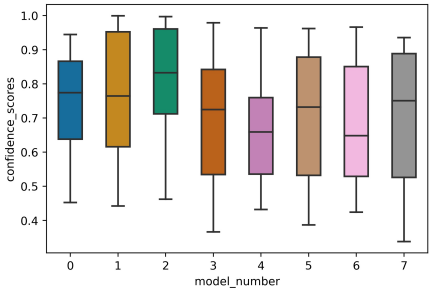
(3) Experiment Three
SVM + FastText



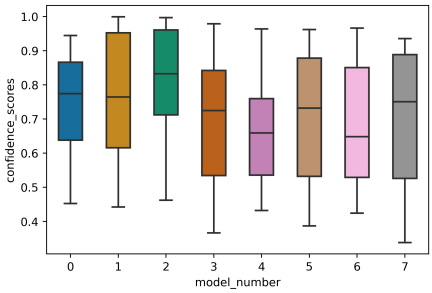
(4) Experiment Four
and CNN + FastText



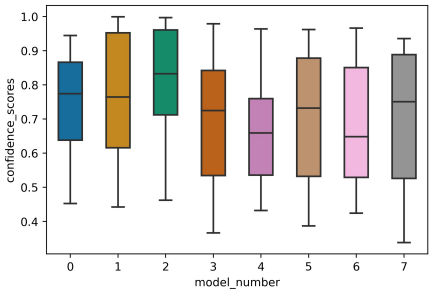
(5) Service



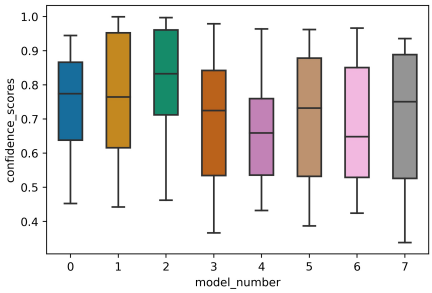
(6) Sales and Related



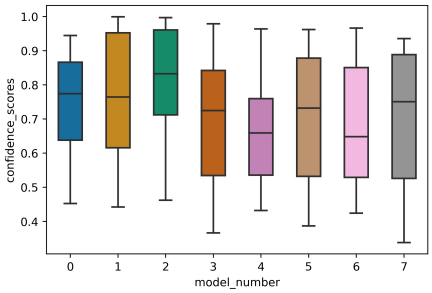
(1) Management, Business and Financial



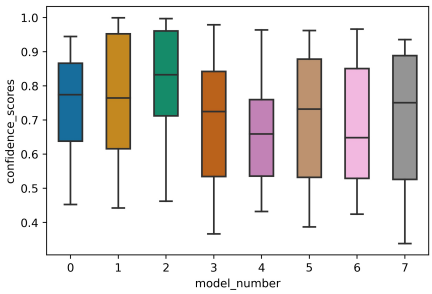
(2) Computer, Engineering, and Science



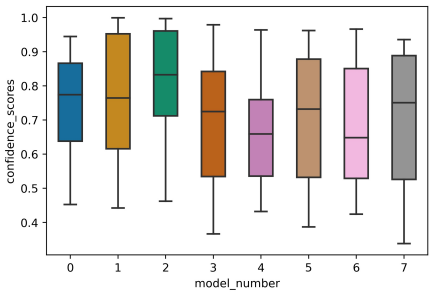
(3) Education, Legal, Community Service, Arts, and Media



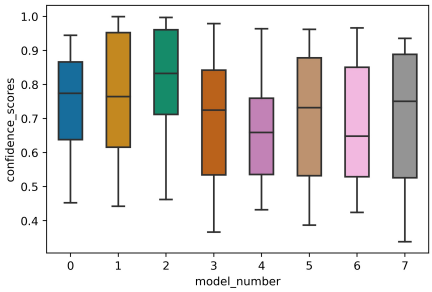
(4) Healthcare Practitioners and Technical



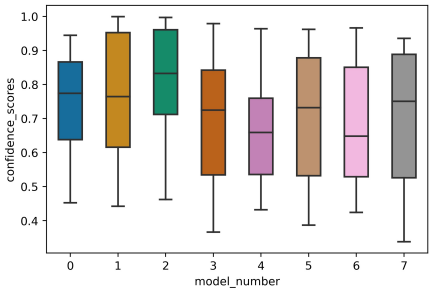
(5) Service



(6) Sales and Related



(7) Office and Administrative Support



(8) Farming, Fishing, and Forestry

Bibliography

Aktualisierung der KldB 2010 und der Einzelberufe. https://statistik.arbeitsagentur.de/nn_237808/Statischer-Content/Grundlagen/Klassifikationen/Klassifikation-der-Berufe/KldB2010/Arbeitshilfen/Aenderungen-Berufe/Aenderungen-Berufe.html. Accessed: 2020-06-05.

International Standard Classification of Occupations (ISCO). https://ilostat.ilo.org/resources/methods/classification-occupation/#isco08. Accessed: 2020-06-05.

Karrierebibel: (m/w/d) in Stellenanzeigen: Was bedeutet das? https://karrierebibel.de/m-w-d-in-stellenanzeigen/. Accessed: 2020-20-05.

Ontotext: What are Ontologies? https://www.ontotext.com/knowledgehub/fundamentals/what-are-ontologies/. Accessed: 2020-30-06.

S. Arora, Y. Liang, and T. Ma. A simple but tough-to-beat baseline for sentence embeddings. 2016.

N. Aussenac-Gilles and M.-P. Jacques. Designing and evaluating patterns for ontology enrichment from texts. In *International Conference on Knowledge Engineering and Knowledge Management*, pages 158–165. Springer, 2006.

Y. Babakhin, A. Sanakoyeu, and H. Kitamura. Semi-supervised segmentation of salt bodies in seismic images using an ensemble of convolutional neural networks. In *German Conference on Pattern Recognition*, pages 218–231. Springer, 2019.

A. Barushka and P. Hajek. The effect of text preprocessing strategies on detecting fake consumer reviews. In *Proceedings of the 2019 3rd International Conference on E-Business and Internet*, pages 13–17, 2019.

D. S. Batista, B. Martins, and M. J. Silva. Semi-supervised bootstrapping of relationship extractors with distributional semantics. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 499–504, 2015.

R. I. Ber and T. Haramaty. Domain adaptation in highly imbalanced and overlapping datasets. *arXiv preprint arXiv:2005.03585*, 2020.

T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific american*, 284(5):34–43, 2001.

R. Boselli, M. Cesarini, F. Mercorio, and M. Mezzanzanica. Classifying online job advertisements through machine learning. *Future Generation Computer Systems*, 86:319–328, 2018.

K. Bouton. Recruiting for cultural fit. *Harvard Business Review*, 17, 2015.

R. L. Brennan and D. J. Prediger. Coefficient kappa: Some uses, misuses, and alternatives. *Educational and psychological measurement*, 41(3):687–699, 1981.

O. Chapelle, J. Weston, and B. Schölkopf. Cluster kernels for semi-supervised learning. In *Advances in neural information processing systems*, pages 601–608, 2003.

L. Chiticariu, Y. Li, and F. Reiss. Rule-based information extraction is dead! long live rule-based information extraction systems! In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 827–832, 2013.

S. B. CHOI, J.-H. YOON, and W. LEE. The modified international standard classification of occupations defined by the clustering of occupational characteristics in the korean working conditions survey. *Industrial health*, pages 2018–0169, 2019.

F. Colas and P. Brazdil. Comparison of svm and some older classification algorithms in text classification tasks. pages 169–178, 2006.

J. Djumalieva, A. Lima, C. Sleeman, et al. Classifying occupations according to their skill requirements in job advertisements. Technical report, Economic Statistics Centre of Excellence (ESCoE), 2018.

A. Emmel and T. Cosca. Occupational classification systems: Analyzing the 2010 standard occupational classification (soc) revision. *Bureau of Labour Statistics*, 2010.

D. Faggella. What is machine learning?, 2020. URL `https://emerj.com/ai-glossary-terms/what-is-machine-learning/`.

J. Firth. A synopsis of linguistic theory. *Studies in Linguistic Analysis*, 1957.

M. Goindani, Q. Liu, J. Chao, and V. Jijkoun. Employer industry classification using job postings. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 183–188. IEEE, 2017.

Y. Grandvalet and Y. Bengio. Semi-supervised learning by entropy minimization. In *Advances in neural information processing systems*, pages 529–536, 2005.

S. T. Gries. Polysemy. *Cognitive Linguistics-Key Topics*, page 23, 2019.

S. Gupta. *Distantly Supervised Information Extraction Using Bootstrapped Patterns*. PhD thesis, Stanford University, 2015.

S. Gupta and C. D. Manning. Improved pattern learning for bootstrapped entity extraction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 98–108, 2014.

Z. S. Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

Hearst. Automatic acquisition of hyponyms from large text corpora. 1992.

J. H. Hoffmeyer-Zlotnik and U. Warner. *Harmonising demographic and socio-economic variables for cross-national comparative survey research*. Springer Science & Business Media, 2013.

L. Huang, J. Zhao, B. Zhu, H. Chen, and S. V. Broucke. An experimental investigation of calibration techniques for imbalanced data. *IEEE Access*, 8:127343–127352, 2020.

A. Iscen, G. Tolias, Y. Avrithis, and O. Chum. Label propagation for deep semi-supervised learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5070–5079, 2019.

N. Kerzazi and B. Adams. Who needs release and devops engineers, and why? In *Proceedings of the International Workshop on Continuous Software Evolution and Delivery*, pages 77–83, 2016.

J. Kim, Y. Ko, and J. Seo. A bootstrapping approach with crf and deep learning models for improving the biomedical named entity recognition in multi-domains. *IEEE Access*, 7:70308–70318, 2019.

Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

D.-H. Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 2, 2013.

M. Liebeck and S. Conrad. IWNLP: Inverse Wiktionary for Natural Language Processing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 414–418. Association for Computational Linguistics, 2015. URL `http://www.aclweb.org/anthology/P15-2068`.

B. Liu, X. Li, W. S. Lee, and P. S. Yu. Text classification by labeling words. In *AAAI*, volume 4, pages 425–430, 2004.

A. Mandelbaum and D. Weinshall. Distance-based confidence score for neural network classifiers. *arXiv preprint arXiv:1709.09844*, 2017.

J. L. Martinez-Rodriguez, A. Hogan, and I. Lopez-Arevalo. Information extraction meets the semantic web: a survey. *Semantic Web*, (Preprint):1–81, 2020.

M. L. McHugh. Interrater reliability: the kappa statistic. *Biochemia medica: Biochemia medica*, 22(3):276–282, 2012.

S. Muehlemann and H. Pfeifer. The structure of hiring costs in germany: Evidence from firm-level data. *Industrial Relations: A Journal of Economy and Society*, 55 (2):193–218, 2016.

A. Niculescu-Mizil and R. Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 625–632, 2005.

K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using em. *Machine learning*, 39(2-3):103–134, 2000.

A. Odena, A. Oliver, C. Raffel, E. D. Cubuk, and I. Goodfellow. Realistic evaluation of semi-supervised learning algorithms. 2018.

U. of Labor Statistics. *2018 Standard Occupational Classification User Guide*. U.S.Bureau of Labor Statistics.

I. Panda. Internet and web technology. 2020.

M. Pasca. Acquisition of categorized named entities for web search. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 137–145, 2004.

W. Paulus, B. Matthes, et al. The german classification of occupations 2010: structure, coding and conversion table. *FDZ-Methodenreport*, 8:2013, 2013.

E. Pintelas, I. E. Livieris, and P. Pintelas. A grey-box ensemble model exploiting black-box accuracy and white-box intrinsic interpretability. *Algorithms*, 13(1):17, 2020.

E. Riloff. Automatically generating extraction patterns from untagged text. In *Proceedings of the national conference on artificial intelligence*, pages 1044–1049, 1996.

S. Roller, D. Kiela, and M. Nickel. Hearst patterns revisited: Automatic hypernym detection from large text corpora. *arXiv preprint arXiv:1806.03191*, 2018.

S. S. Sathe, D. B. Davenport, H. Shah, and N. D. Kshirsagar. Ontology development for profile matching, Jan. 31 2017. US Patent 9,558,271.

H. Schwenk and M. Douze. Learning joint multilingual sentence representations with neural machine translation. *arXiv preprint arXiv:1704.04154*, 2017.

S. Sekine and E. Ranchhod. *Named entities: recognition, classification and use*, volume 19. John Benjamins Publishing, 2009.

Skills-OVATE. Skills-OVATE: Skills Online Vacancy Analysis Tool for Europe. `https://www.cedefop.europa.eu/en/data-visualisations/skills-online-vacancies/online-job-advertisements-providers`, 2020. Accessed: 2020-20-05.

R. Speer and J. Lowry-Duda. Conceptnet at semeval-2017 task 2: Extending word embeddings with multilingual relational knowledge. *arXiv preprint arXiv:1704.03560*, 2017.

M. Thelen and E. Riloff. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *Proceedings of the 2002 conference on empirical methods in natural language processing (EMNLP 2002)*, pages 214–221, 2002.

A. Turrell, B. J. Speigner, J. Djumalieva, D. Copple, and J. Thurgood. Transforming naturally occurring text data into economic statistics: The case of online job vacancy postings. Technical report, National Bureau of Economic Research, 2019.

T. Van Huynh, K. Van Nguyen, N. L.-T. Nguyen, and A. G.-T. Nguyen. Job prediction: From deep neural network models to applications. *arXiv preprint arXiv:1912.12214*, 2019.

H. Wang, Z. Xu, and W. Pedrycz. An overview on the roles of fuzzy set techniques in big data processing: Trends, challenges and opportunities. *Knowledge-Based Systems*, 118:15–30, 2017.

D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd annual meeting of the association for computational linguistics*, pages 189–196, 1995.

M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, and R. Kumar. Predicting the type and target of offensive posts in social media. *arXiv preprint arXiv:1902.09666*, 2019.

Z. Zhang and J. Iria. A novel approach to automatic gazetteer generation using wikipedia. In *Proceedings of the 2009 Workshop on The People's Web Meets NLP: Collaboratively Constructed Semantic Resources*, pages 1–9. Association for Computational Linguistics, 2009.

H. Zindler. Anglizismen in der deutschen presse nach 1945 (doctoral dissertation, universität kiel, 1959). *Kiel: UP*, 1959.

C. Züll. Coding of occupations (version 2.0). 2016.