

A Context Aware Approach for Extracting Hard and Soft Skills

Master Thesis

Regular thesis

M.S.C. Business Intelligence and Smart Services

School of Business and Economics

Author:

I.J.P. (Ivo) Wings

i6262955

i.wings@student.maastrichtuniversity.nl

Supervisors:

Dr. Rohan Nanda
Maastricht University

Dr. Kolawole John Adebayo
Businesspoint Intelligence Solutions Ltd

Second reader:

Dr. Niels Holtrop
Maastricht University



Maastricht, August 8, 2021

Acknowledgements

My research would have been impossible without the help and aid of Dr. Rohan Nanda and Dr. Kolawole John Adebayo. Both supervisors managed to make themselves available when I had questions about this research project. Which was sometimes difficult due to communication over zoom or teams during the COVID-19 crisis. The natural language processing field was relatively new to me at the start of this research project. However, both supervisors gave me the confidence and resources to develop myself further in this field for which I am very grateful.

I also want to express my gratitude towards my supervisors for pushing me to publish a paper in the Procedia Computer Science journal. This whole experience was completely new to me and taught me a lot about writing as well as the academic world in general.

Finally, I would like to thank my partner and family to guide me through this eight-month process and keep me focused.

Abstract

The continuous growth in the online recruitment industry has made the candidate screening process costly, labour intensive, and time-consuming. Automating the screening process would expedite candidate selection. In recent times, recruiting is moving towards skill-based recruitment where candidates are ranked according to the number of skills, skill's competence level, and skill's experience. Therefore it is important to create a system that can accurately and automatically extract hard and soft skills from candidates' resumes and job descriptions. The task is less complex for hard skills which in some cases could be named entities but much more challenging for soft skills which may appear in different linguistic forms depending on the context. In this thesis, a context-aware sequence classification and named entity recognition approach for extracting both hard and soft skills are proposed. These models utilize the most recent state-of-the-art word embedding representations as textual features for various machine learning classifiers. The models have been validated by evaluating them on a publicly available job description dataset. The results indicated that the best performing sequence classification model used fine-tuned BERT embeddings in addition to POS and DEP tags as input for a logistic regression classifier. The Transformer model outperformed the HashEmbedCNN models. The token classification model used fine-tuned BERT embeddings in addition to POS and DEP tags as input for a support vector machine classifier and outperformed the CRF model.

Keywords: Natural language processing; Human Resources management; Skill extraction; Named entity recognition

Contents

List of Figures	v
List of Tables	vii
1 Introduction	1
2 Literature review	5
2.1 Skill extraction	5
2.2 Taxonomy creation	9
2.3 Word embeddings	10
2.4 Summary	11
3 Methodology	13
3.1 Data collection	13
3.2 Text pre-processing and dataset preparation	14
3.2.1 Job descriptions	14
3.2.2 Taxonomies	16
3.2.3 Soft skills	16
3.2.4 Candidate skill and context selection with N-grams	16
3.2.5 Data Annotation	18
3.3 Context-Aware Sequence Classification Model	19
3.3.1 Word embeddings	19
3.3.2 Feature selection	21
3.3.3 Classifiers	23
3.4 Semi-supervised	24
3.5 Named Entity Recognition	25
3.5.1 Dataset preparation	25

3.5.2	HashEmbedCNN	26
3.5.3	Transformer	27
3.5.4	Conditional Random Fields (CRF)	28
3.5.5	Token Classification Model	28
3.6	Evaluation	29
3.6.1	Sequence classification model parameters	29
3.6.2	Named entity recognition parameters	31
3.7	Summary	32
4	Results and analysis	33
4.1	Context-Aware Sequence Classification Model	33
4.2	Semi-supervised Sequence Classification Model	38
4.3	Named Entity Recognition Models	39
4.3.1	HashEmbedCNN	39
4.3.2	Transformer	41
4.3.3	CRF	43
4.3.4	Token Classification Model	44
4.4	Summary	49
5	Conclusion and future work	51
5.1	Conclusion	51
5.2	Limitations	53
5.3	Future work	53
	Bibliography	55
A	Statement of original thesis	59

List of Figures

- 3.1 LDA topic distribution using the top 5 dominant topics 15
- 3.2 Non-contextual word embeddings feature selection 22
- 3.3 Contextual word embeddings feature selection 23

- 4.1 Precision sequence classification model 34
- 4.2 Recall sequence classification model 34
- 4.3 F1-score sequence classification model 35
- 4.4 Confusion matrix baseline sequence classification 20% test data 35
- 4.5 Confusion matrix BERT fine-tuned POS DEP LR sequence classification model
20% test data 35
- 4.6 Sequence classification model predictions 37
- 4.7 SHAP results BERT fine-tuned POS & DEP tags LR sequence classification
model 37
- 4.8 SHAP results FastText AVG POS & DEP tags RF sequence classification model 37
- 4.9 Semi-supervised sequence classification model predictions 39
- 4.10 Confusion matrix HashEmbedCNN accuracy 10% test data 40
- 4.11 Confusion matrix HashEmbedCNN efficiency 10% test data 40
- 4.12 Confusion matrix baseline 10% test data 40
- 4.13 DisplaCy example 41
- 4.14 Learned hard and soft skills HashEmbedCNN model 41
- 4.15 Confusion matrix baseline 10% test data 42
- 4.16 Confusion matrix Transformer 10% test data 42
- 4.17 Learned hard and soft skills Transformer model 42
- 4.18 Confusion matrix baseline 20% test data 43
- 4.19 Confusion matrix CRF 20% test data 43

4.20	Learned hard and soft skills CRF model	44
4.21	Precision token classification model	45
4.22	Recall token classification model	45
4.23	F1-score token classification model	46
4.24	Confusion matrix baseline token classification 20% test data	46
4.25	Confusion matrix fine-tuned BERT POS DEP SVM token classification model 20% test data	46
4.26	Token classification model predictions	47
4.27	Learned hard and soft skills token classification model	47
4.28	SHAP results BERT fine-tuned POS & DEP tags SVM token classification model	48

List of Tables

- 3.1 Manual annotation example 18
- 3.2 Class distribution 19
- 3.3 Fine tuning sequence classification 21
- 3.4 POS and DEP example 23
- 3.5 Tokenized dataset 26
- 3.6 Class distribution tokenized dataset 26
- 3.7 Token classification example 29

1 Introduction

With the continuous growth of the internet and the ubiquitous access it offers, it has become a central market place where jobs are advertised. For every advertised job, employers and recruiters receive a large number of resumes from applicants. These applications require screening in order to shortlist the most qualified candidates. Due to the costly and time-consuming nature of the screening process, there is a need for automation. On one hand, recruiting is drastically moving towards skills-based hiring. This is the case where candidates are assessed and ranked based on the skills they possess and how the skills align with the competency requirement of the job. On the other hand, the focus on skills is important since the skill set of employees dictates the level of innovation, profitability, and sustainability of an organization. To appraise employees skills or assess candidates' competency for skills-based hiring, it is important to put in place a system that can automatically identify skills, no matter how subtle, in human capital management (HCM) documents, e.g., curriculum vitae (CV), job advertisements, appraisal documents etc. This is important for automated systems that perform candidate search, match and rank; skills-gap analytics; employment market analysis; and re-skilling and workforce training objectives.

Skills are expressed in documents using human language. However, human language can be ambiguous, and exhibit the problem of polysemy and synonymy. For example, assuming that a HR employee is matching candidates to job descriptions related to software development. It would be necessary to differentiate between Java programming language (skill) and Java (geo-location) which is an island in Indonesia. In this example, the use of a Skill Gazetteer (list of skills) or a sequence classifier that does not take the context of each word into consideration would not be effective and also not generalize to previously unseen words or future skills. Natural language processing (NLP) is a form of computational linguistics used to learn, understand, and produce human language content[1]. NLP is used for nu-

merous purposes, such as machine translation or extracting actionable knowledge from text documents. Researchers used NLP techniques and state-of-the-art machine learning classifiers to learn and identify skills in order to automate the skill extraction process [2].

The objective of this master thesis is to create a state-of-the-art hard and soft skill extraction system in order to increase the efficiency of the HR recruitment process. This system needs to be context-aware in order to differentiate between actual skills or other tokens. Furthermore, the system needs to be able to learn new skills which were not in the training dataset. Therefore, the following main research question is defined: **How can we automatically extract hard and soft skills from job descriptions in order to increase the efficiency of the HR recruitment process?** There have been previous attempts to automatically extract hard and soft skills from job descriptions [2] [3] [4] [5] [6] [7]. However, these attempts could be improved by using the recent state-of-the-art word embedding techniques. Moreover, existing works framed the problem as a classification task without considering the context of each. The main research question is split up into three sub-questions. The first sub-question is: **What impact has the context in classifying terms as soft/hard skill or not skill?** This question encapsulates the importance of the context in order to differentiate skills from other tokens as shown in the previous example. The second sub-question is: **How to develop one state-of-the-art robust model for extracting both hard and soft skills?** Previous work focused on either hard or soft skill extraction from natural text. The second sub-question highlights the creation of one model using state-of-the-art techniques which can extract both hard and soft skills. The final sub-question is: **Can the state-of-art model learn new hard and soft skills?** The final sub-question is about the learning capabilities of the proposed approach. If the solution is unable to learn new skills, the use of a skill gazetteer might be sufficient. The main research question will be answered through the sub-questions. The first sub-question will be answered by comparing traditional techniques and state-of-the-art techniques. The second sub-question will be answered partially through a literature review as well as experimentation. The final sub-question will be answered by evaluating the results of the proposed systems.

This master thesis proposes multiple context-aware approaches for skill extraction. Further-

more, this thesis attempts an extensive evaluation and overview to compare state-of-the-art approaches to previously used techniques. Next to these contributions, the creation of one robust model for extracting hard and soft skills is presented.

This thesis is structured as follows. The second chapter describes the related works on skill extraction, taxonomy creation, and different traditional and state-of-the-art word embedding techniques. The third chapter provides insight into the methodology used to build the different skill extraction pipelines as well as the evaluation techniques used. The fourth chapter presents the results achieved from the different pipelines and the analysis of those results. To conclude, the fifth chapter presents the conclusion of the entire thesis in addition to the future works and limitations.

2 Literature review

The previous chapter introduced the topic, motivation, objectives, and the scope of this thesis. This chapter is used to describe the related works of other researchers tackling the skill extraction problem. It is essential to establish previously tried experiments in order to find gaps within these approaches. After determining the gaps, these can be closed by creating a new methodology. The first section in this chapter describes previous work on skill extraction. The second section describes taxonomy creation which is useful in order to identify potential skills within the text. Finally, section three describes various word embedding techniques that can be used to capture semantic meaning from text. The last section provides a summary of the findings in this chapter.

2.1 Skill extraction

The work in [2] describes a method for skill identification and normalization in resumes using NER and Named-entity normalization (NEN). The authors gathered around 100 million resumes from CareerBuilder and found that 90% of these used skills to summarize qualifications. This finding shows a critical demand for a skills taxonomy by which resumes and job descriptions can be analyzed. The authors searched the resumes for a specific skills section between the skill section's header and the subsequent section. If there was no skills section available, the entire resume was split by punctuations. The following step was to remove all noise in the document, such as stop words, adverbs, adjectives, country, and city names. After this, the results are searched on Wikipedia. If no result was found, the result was removed, as this result had a high probability of not being a skill. If a Wikipedia page was found, the category tags were extracted. These category tags are then used to create the skill taxonomy. The category tags are also searched in Google to remove the ambiguity. The highest-ranked result was selected as the final meaning of the term. After extracting

the skills for the resumes, a skills tagging system was built. Word2Vec was used to build this skills tagging system. The vectors are then used to calculate a relevancy score between the term and the taxonomy to determine the final meaning. The system solely relied on the relevance score and did not take the context or other semantic features into account. Overall this paper provided a good introduction to taxonomy creation and skill extraction using NLP techniques. The approach could be improved by using state-of-the-art word embedding techniques.

The work in [8] describes a system for skill sense disambiguation. The data the authors used was provided by CareerBuilder and consisted of resumes and job postings. Each document in the dataset contained the extracted skills set. All documents were clustered based on their base skill form. By clustering the skills on their base form, different clusters are generated which are related. After generating the clusters, the context of the skill is encoded using an N-gram model. The encoded skills are then used to train Word2Vec vectors. After training the Word2Vec vectors, the vectors are clustered using the Markov Chain Monte Carlo algorithm to determine the final sense. The end results are pruned in order to remove noisy skill terms. For this paper, skill sense disambiguation is not a focus. However, the usage of N-grams to capture skill within the text will be used. The authors used the average of the Word2Vec vectors in order to encode the skills. Resulting in a unique embedding for the entire skill sequence. By using state-of-the-art word embeddings, the clustering could be improved as the clusters better describe the meaning of the skill terms.

Another method for extracting skills was employed by using LinkedIn [3]. The authors created a folksonomy for members to select their skills. Most LinkedIn users summarize their skills in a comma-separated list on their page. The authors searched for this list and cleaned the list for other entities than skills using the phrase frequency in the specialty section between the frequency of other sections to create a ratio. After extracting the skills, a clustering algorithm was used to detect ambiguity between terms. A list of related phrases was generated for each skill phrase. This method is used because an ambiguous phrase has a different set of related phrases in each context. A clustering algorithm is used on the phrases, and

the most dominant cluster is selected as the final sense. Duplicates are removed through a crowdsourced approach. Wikipedia skill definitions are shown, and users can select the correct meaning of the skill. The results of the paper indicated that the system was able to predict skills in multiple languages. The method proposed in this paper was novel because a folksonomy was used instead of a traditional taxonomy.

In [4] the authors of the paper devised a system that predicted high-level from resumes even if these are not specifically mentioned with the resume. The authors used a Convolutional neural network(CNN) model for this research problem. The problem was handled as a multi-label classification problem. The authors used resumes collected from the internet. Each resume was labeled with the specific skills of the uploader. The first step was normalizing all the classes using a handwritten dictionary-based normalizer algorithm. Next, word embedding models were trained on the resumes. After training the word embedding model, each word in a resume was turned into a vector and stored in a matrix. Finally, the labeled dataset is divided into class-specific sets on which independent CNN classifiers are trained to make the final predictions. Training different classifiers for each of the classifiers worked very well in this example. But in the context of this paper, it would be interesting to make a more robust model which could in a later stage make predictions on job descriptions as well. The system also had the best performance for IT-related resumes. It would be interesting to see more applications in different sectors.

Previous work used supervised approaches for extracting skills, Gaur et al.[9] propose the use of a semi-supervised approach for extracting the education qualifications of candidates. The dataset consisted of 550 resumes. A small proportion of training data was manually annotated. After this, the classification model was trained. The classification model consisted of a word embedding layer, CNN layer, and Bi-LSTM layer. The predictions that the model produced are corrected using a correction module. This module was a list of institute names and degree names. Fuzzy wuzzy search is employed to correct the difference between the predictions and the dictionary. After correction, the data is added to the training set, and the process is repeated until the subsets are exhausted. The semi-supervised approach is in-

interesting because it helps alleviate some of the manual annotation that is necessary to train a classifier. The results could be improved by using manually annotated data as the input as some skills are context-dependent which cannot be captured using gazetteer annotation.

Sayfullina et al. [5] described a system used for detecting soft skills in unstructured text corpora. The authors formulated the problem as a binary classification problem where the positive class refers to the candidate's skills while the negative class relates to others. The dataset consisted of a corpus of job advertisements, resumes, and a publicly available soft skills list. The training dataset was created using a crowdsourced approach. Job advertisements are checked for exact matches with the soft skill list. The matches are highlighted in the sentences after which these sentences are shown to people and manually annotated. The annotated data is used as input for the LSTM, CNN, and HAN neural networks. The authors used an extensive evaluation process for the annotations in order to make sure that only correct annotations would be used by the classifiers. The LSTM neural network had the highest performance.

Related skill extraction research created a system called SkillNER [6]. SkillNER is a Named entity recognition system using a support vector machine trained for soft skill extraction. The system was trained on a corpus of 5000 scientific papers and used a soft skill taxonomy from O*NET. The system contains two stages: clue extraction and skill extraction. The clue extraction stage consisted of extracting patterns before entities that would hint towards a soft skill. These extracted clues were used to match with sentences throughout the corpus. These sentences were finally annotated in the skill extraction stage. This dataset was used to train and evaluate a support vector machine and a multi-layer perceptron classifier. Finally, the support vector machine was implemented as a SpaCy model. The approach of the paper is novel. However, the results of the paper indicated that these could be improved.

Kivimäki et al. [10] presented a graph-based approach for skill extraction. The authors obtained a skill taxonomy from LinkedIn and used the hyperlink structure from Wikipedia. They created a module that associated Wikipedia pages to a given input text by using text

similarity techniques such as TF-IDF and LDA. The authors then extracted the Wikipedia skill pages from the list of associated Wikipedia pages. The authors used traditional text similarity functions such as TF-IDF and LDA. The TD-IDF approach had the best performance given this experimental setting. The results of the paper could be improved by using state-of-the-art techniques.

Another paper [11] presented a review on deep learning for named entity recognition. In the paper, the authors present results that indicate that current state-of-the-art named entity recognition systems, use neural network architectures. In particular, BERT is highlighted as a high performer. However, the authors show that BERT needs to be fine-tuned in order to further improve the models performance.

Gugnani et al. [7] present a job recommender system that matches candidate resumes to job descriptions. The main assumption was that skills are the most important features in the matching process. The authors mined job descriptions from open sources and collected 1.1 million job descriptions as well as 1.314 resumes. Skills were collected from O*NET, ComputerHope, and Wikipedia. The noun phrases from the input text were collected using NER. The authors also devised a rule-based system for detecting skills based on the semantic tree. Finally, a dictionary was used for additional skill matching. Word2Vec was used to compare the vectors of the collected skills and the skills in the dictionary. The results indicate that the system is able to predict a job description for an applicant with just 5 recommendations. The researchers presented a combination of different approaches and combined the results into one system. This approach, therefore, takes advantage of multiple approaches each with its advantages and disadvantages.

2.2 Taxonomy creation

Previous work focused on extracting skills from corpora. Khaouja et al. [12] created a soft skill taxonomy using job openings which can be used for skill extraction as well. The dataset included job postings from different boards such as CareerBuilder, Apec, and Keljob. The authors established the first list of soft skills using the most common bi and tri-gram phrases

that occurred at least ten times in the corpora. The N-grams are searched in DBpedia, and if a page is associated with the N-gram, the value was stored. The list of N-grams was supplemented with a list of soft skills from previous research. The N-grams are also checked for an association with an enterprise or job description. A naive Bayes text classifier was trained on manually labeled data to determine this. After this step, the N-grams were converted into word vectors using Word2Vec. Parallel to this, the N-grams were searched on DBpedia, where related words are extracted and stemmed. Finally, the cosine similarity between the N-grams and DBpedia terms is calculated and used to determine the hierarchy. The results of the paper show a soft skill taxonomy in English and in the French language.

2.3 Word embeddings

Mikolov et al. [13] proposed a system that could transform natural text into word vectors. The system is called Word2Vec. Word2Vec is an efficient method to learn vectors representations from large corpora. The vector representations can be used to encode linguistic regularities and patterns. These can then in turn be used to compare words mathematically. The authors show that the vector for 'Madrid' is closer to Spain than to France. The authors also show that by vector summation new meanings can be derived for the word vectors. For example, the summation of the vectors for 'Germany' and 'capital' closely resembles 'Berlin'. This further showcases the capabilities of the Word2vec in capturing semantic information. The Word2Vec model was trained on news articles. An improvement for this paper would be the evaluation of Word2Vec performance on different data sources.

An improved method for generating word embeddings is presented in [14]. The authors build upon the Word2Vec system described in the previous section. They devised a system that could enrich the word vectors with morphology. The authors modeled morphology as subword units. By incorporating this morphology the word embedding include more linguistic meaning in the word embeddings. The system is called FastText. FastText was evaluated on Wikipedia data. The results show that FastText outperforms Word2Vec. It would have been interesting to evaluate the performance on different datasets instead of only using

Wikipedia data.

Devlin et al. [15] created a novel approach for word embeddings. The authors create a context-aware word embedding technique. BERT (Bidirectional Encode Representations from Transformers) is a neural network that generates context-aware word embeddings. The model reads the entire corpora bidirectional simultaneously. After reading the entire corpora unique embeddings are assigned to tokens. Therefore, capturing more semantic information in the word embeddings. Another benefit of BERT is that it can be easily fine-tuned on specific tasks to further increase performance. Overall, the paper shows that BERT has superior performance over non-contextual word embeddings.

Similar to BERT is Embeddings from Language Models (ELMo) [16]. ELMo uses all the stages of the biLM structure in order to create meaningful contextual embeddings. It concatenates the forward LM and backward LM stages to create the embeddings. The difference with BERT is that ELMo uses a unidirectional LSTM. This causes the model to capture less semantic information. The results of the paper indicate that ELMo outperforms non-contextual word embeddings.

2.4 Summary

The literature review shows that many researchers attempted to solve the skill extraction problem with different methods. All researchers used traditional non-contextual word embeddings together with different feature selection techniques. The literature also shows that the summation of non-contextual word embeddings leads to new contextual meanings. Next to these findings, is the usage of taxonomies to initially find skills within the corpora. Researchers either created the taxonomies within the system or used taxonomies collected from different sources. The most important finding is that most of the researchers did not take the context of the skills into account. However, the context could drastically change the meaning of the text resulting in errors in the skill extraction process.

3 Methodology

The literature review showed that previous work did not take the context of skills into consideration while building the skill extraction pipelines. Furthermore, the literature review showed numerous word embedding techniques that can be used to capture linguistic meaning from text. Finally, taxonomies can be used to initially find skills within the text. This chapter describes the methodology used to create the context-aware skill extraction pipelines. The first section describes the dataset used for the skill extraction process as well as the collected hard and soft skill taxonomies. The second section describes the various pre-processing steps in order to make the corpora ready for analysis. The third section describes the first approach, the context-aware sequence classification model. This model builds upon the insights from the literature review and uses novel context and skill selection as well as non-contextual and contextual word embedding techniques. The fourth section describes a variation of the best performing sequence classification model with the usage of a semi-supervised approach in order to alleviate the need for an increased amount of manual annotation. The fifth section describes various approaches for NER. Different state-of-the-art NER techniques are compared and created in order to find the best performing NER solution. Finally, the hyperparameters and evaluation techniques for all the proposed models are described.

3.1 Data collection

The data collection section describes all the collected data used to create the skill extraction pipeline. The job description dataset is described as well as the skill taxonomies. Finally, an Latent Dirichlet Allocation topic modeling graph is shown to provide insights into the main topics of the job descriptions.

The publicly available Employment Scam Aegean Dataset (EMSCAD)[17] was used as job

description set. The dataset contains 17,014 legitimate and 866 fraudulent job ads. Only the legitimate job descriptions were used from the dataset. After collecting the job descriptions, hard- and soft skill taxonomies were collected. Two taxonomies have been created. The first taxonomy is a soft skill taxonomy presented in [12]. This soft skill taxonomy is created using job descriptions and thus a perfect match for this use case. It contains 579 soft skills after removing duplicates. Because this taxonomy only contains soft skills, it was supplemented with hard skills from the EMSI skills APIⁱ as well as ESCOⁱⁱ and O*NETⁱⁱⁱ taxonomies. A combined total of 28280 hard skills were collected from these sources.

In addition, a soft skill dataset was collected from [5]. This dataset contained annotated soft skills along with the respective context. These soft skills are used alongside the manually annotated samples. Overall, 7220 annotated soft skills were collected.

Some Latent Dirichlet Allocation (LDA) topic modeling has been used to provide insights into the topics of the job descriptions. The five most dominant topics are visualized in Figure 3.1. These topics are named empirically. The two major topics in the JD set are Design and Learning.

3.2 Text pre-processing and dataset preparation

After collecting all the necessary data, the data needs to be pre-processed in order to make it suitable for analysis. These steps are described in this section. Next to the pre-processing steps, the data annotation process is thoroughly described.

3.2.1 Job descriptions

The first step in cleaning the job descriptions was removing the HTML tags. Removing the HTML tags has been done using the Python package BeautifulSoup. After removing the HTML tags the 'description' and 'requirements' columns have been concatenated. The description column contained general information about the company as well as some soft

ⁱ<https://skills.emsidata.com/>

ⁱⁱ<https://ec.europa.eu/esco/portal/home>

ⁱⁱⁱ<https://www.onetcenter.org/taxonomy.html>



Figure 3.1: LDA topic distribution using the top 5 dominant topics

skills and the requirements column contained pre-extracted hard and soft skills. Because this paper focuses on skills extraction, the columns have been concatenated in order to create realistic job descriptions. The next step included removing the pre-masked patterns of email addresses, phone numbers, and URL's. These were masked as Hash patterns and have been removed using RegEx expressions. The job description dataset was further manually checked. During this step, it became clear that there were still names, phone numbers, and email addresses in the job description set. Therefore further anonymization steps needed to be taken. Presidio^{iv} was used to further anonymize the names, phone numbers, and email addresses in the dataset. Next to Presidio, a large dictionary of names^v was used to search the job descriptions. Any matches with this dictionary were removed. After using the list, custom RegEx patterns were created to remove any remaining phone numbers with less tra-

^{iv}<https://github.com/microsoft/presidio>

^v<https://github.com/philipperemy/name-dataset>

ditional formats. The final anonymization step included the usage of SpaCy^{vi}. The NER module was used and any tokens with the 'PERSON' label were removed. After completely anonymizing the job descriptions dataset, all the text was lower-case, newline symbols removed, and finally, the leading and trailing whitespaces were removed. The punctuation was kept in order to transform the job descriptions into sentences. The job descriptions were then sentence-tokenized using SpaCy sentencizer and thereafter, any special characters were removed to standardize the text. These pre-processing steps yielded 302284 sentences.

3.2.2 Taxonomies

The taxonomy presented in [12] was read and transformed into a list of words. Each item in the list was then lower-cased and any special characters were removed in order to standardize the text. The same procedure was performed for the ESCO and O*NET Taxonomies. The taxonomy collected from ^{vii} only contained hard skills which did not need any transformation in order to be used in the model.

3.2.3 Soft skills

The soft skill dataset consisted of a list of pre-annotated soft skills along with their contexts. The original location of the soft skill was masked in the context using 'xxx'. For example, 'you will require xxx to manage your own' the soft skill originally in 'xxx' was 'initiative'. The 'xxx' was used to split the context column into left context and right context. This methodology is also used later in this paper. After creating the context columns, the 'xxx' was replaced with the soft skill. After replacing the soft skill, all the text was lower-cased, and any special characters were removed. After standardizing the text, each annotation was manually checked for correctness.

3.2.4 Candidate skill and context selection with N-grams

After performing the pre-processing steps in the preceding section, it is necessary to utilize N-grams to generate *candidate skills* from the job description collection. N-grams are

^{vi}<https://spacy.io/>

^{vii}<https://skills.emsidata.com/>

sequences of words made from corpora and are used to turn sentences into smaller chunks better suited for analysis and annotation. In this case, the N-grams are also used to capture the context around a reference term in a sentence. This is especially useful in the case of soft skills. All the sentences of the job descriptions were transformed into N-grams up to four-grams. The decision was made to go up to four-grams because most skills on average are composed of 1-4 constituent words. The N-grams are created using the NLTK^{viii} package in Python. After transforming the sentences into N-grams, a large dataset was created. This dataset also contained N-grams which did not contain any skills, therefore, such N-grams were removed. To achieve this, an integrated gazetteer was used which combines entries from the aforementioned skill taxonomies to filter out the noisy N-grams and retain only those contained in our integrated gazetteer. To ensure exact matches, Flashtext^{ix} was employed as the N-grams filter. Subsequently, any redundant N-grams were removed resulting in a total of 323600 distinct N-grams. The next step is to generate the context for each candidate skill. Consider the sentence *'The candidate must be comfortable programming in Java, Python, and other programming languages'*. Using a uni-gram reference word and a context with window size = 4 (i.e., four-grams context), a sample in the dataset will have the word 'Java' as the reference term. First, the gazetteer checks for the word 'Java' and if found, the left context (be comfortable programming in) and right context (Python and other programming) are collected. It is imperative to note that providing the context aids disambiguation which helps the classifier to resolve polysemous terms. For instance, a candidate skill even if found in the gazetteer may not necessarily be a skill as this depends on the context in which the term is used. To buttress this point, consider the term *'systematic'*. This term is contained in the skill gazetteer and can be categorized as a soft skill given the context *'be able to demonstrate a systematic and analytical approach to'*. However, this would not be the case if the context changes to *'applicant should be familiar with systematic invest, and trade future'*. The context window size was derived based on the average length of all sentences in the job description collection which was found to be 15 tokens. Considering the average sentence length, the context length was intuitively chosen with a window size of 5.

^{viii}<https://www.nltk.org/>

^{ix}<https://github.com/vi3k6i5/flashtext>

3.2.5 Data Annotation

This thesis frames the skill extraction problem as a multi-class classification problem with three classes. The prepared dataset is labeled using the labels *Not Skill*; *Soft Skill*; and *Hard Skill*. Previous research [5] framed a soft skill extraction approach as a binary classification problem. The classes were soft skill or not skill. This approach extends these classes with the hard skill label to build one robust model for skill extraction. An additional advantage is that the classes are more granular. Meaning that the results better inform HR employees about the skill distribution of potential employees. Other research [18] [19] shows that most business sectors identify hard and soft skills as key aspects during the hiring process. Further showcasing the importance of the three classes. In this thesis, hard skills are defined as *skills that are teachable and straightforward to quantify*. Soft skills are defined as *skills that are related to interaction and difficult to quantify*. The dataset is formatted in the following pattern: Left-Context, Candidate-Skill, Right-Context, and Label. The candidate-skill can belong to one of the three classes. Gold-Standard annotation is usually manual and therefore labour-intensive. Given the limited resources, it would be difficult to annotate the whole sample collection (over one million samples) in the dataset. To resolve this issue, 13616 samples from the dataset were randomly selected for annotation. To annotate, each candidate skill was only assigned an appropriate label with respect to its context. An example of the annotation is shown in Table 3.1. To facilitate the annotation process, Label Studio^x was used. After the manual annotation process, the Gold-Standard dataset was supplemented with a list of annotated soft skills [5] with 7220 samples, resulting in a total corpus size of 20836 samples.

Table 3.1: Manual annotation example

Left context	Candidate skill	Right context	Actual label
and creative assistant director of	interactive	marketing and new media to	not skill
is expected to be	self motivated	set goals and meet	soft skill
expect an individual with senior	Python	programming experience in healthcare	hard skill

^x<https://labelstud.io/>

After the data annotation steps, the class distribution is presented in Table 3.2. The table shows that there is a class imbalance. This could be explained by the usage of the skill taxonomies. For example, one skill in the taxonomies was *Team*. This led to many matches with N-grams containing the word *Team*. However, most of these matches were labeled as *Not Skill*.

Table 3.2: Class distribution

Class	Counts
Not Skill	16.073
Soft Skill	4.121
Hard Skill	642

3.3 Context-Aware Sequence Classification Model

This section describes the first context-aware skill extraction pipeline: the context-aware sequence classification model. This model utilized the dataset described in the previous section. Furthermore, the feature selection process is described as well as the machine learning classifiers.

3.3.1 Word embeddings

Previous studies on skill extraction have utilized bag-of-words and lexical approaches to model features for machine learning classifiers. This thesis utilized recent state-of-the-art word embedding models for semantic representation of texts as they have shown to outperform lexical and bag-of-words approaches. The major types of word embeddings, such as Word2Vec[13], FastText [14], ELMo [16], and BERT [15] were implemented for performance evaluation. The main difference between these word embeddings is the method used to assign word embeddings. Pre-trained word embeddings were used because the number of job descriptions in the EMSCAD dataset was not sufficient to train word embeddings with an accurate semantic representation.

Non-contextual word embeddings

Word2vec and FastText have been used as non-contextual word embeddings. Assuming an input sentence with multiple words: 'Have good coding skills in Python'. The candidate skill in this sentence would be coding. The phrasal embedding for the context and candidate skill is derived through either vector averaging or summing of the constituent words and the candidate skill(s). The 'glove-wiki-gigaword-300' pre-trained Word2Vec model was used. This model has been trained on a combination of Wikipedia data from 2014 and the Gigaword 5 corpus. It contains 400,000 unique tokens. All of these tokens were lowercased. The final dimensions of the vectors were 300 dimensions. The English language model was used as pre-trained FastText model. This model was trained on Common Crawl and Wikipedia data. The final dimensions of the word embeddings were 300 dimensions as well.

Contextual word embeddings

BERT and ELMo have been used as contextual word embedding methods. Assuming the same input sentence presented in the previous paragraph. The entire sentence was loaded into BERT and ELMO after which the candidate skills embeddings were extracted from the model. Because BERT and ELMo are contextual word embedding methods, each token gets a different embedding depending on the sentence. The 'bert-base-uncased' pre-trained BERT model from ^{xi} was used. This model was trained on BookCorpus and Wikipedia. BookCorpus is a corpus consisting of 11.038 unpublished books. The final dimensions of the pre-trained model are 768 dimensions. The 'small' ELMo model from ^{xii} was used. This model was trained on the 1 Billion Word Benchmark. This Benchmark contains approximately 800 million tokens of news data. The dimensions of this model are 1024.

Fine Tuning BERT sequence classification

BERT is a neural network and can therefore be fine-tuned towards solving specific problems. By fine-tuning BERT the performance of the model is increased as the model learns the specific task it has to perform. The 'bert-base-uncased' model already learned what the English

^{xi}<https://huggingface.co/models>

^{xii}<https://allennlp.org/elmo>

language is, and by fine-tuning, the model learns what skills are in the text. BERT has been fine-tuned two times. Once for sequence classification and once for token classification (described in 3.5.5). The context columns and candidate skill columns were concatenated to create one sequence per label. An example of this is shown in Table 3.3.

Table 3.3: Fine tuning sequence classification

Sequence	Actual label
and creative assistant director of interactive marketing and new media to	not skill
is expected to be self motivated set goals and meet	soft skill
expect an individual with senior Python programming experience in healthcare	hard skill

This dataset was then used as input for the fine-tuning process. BERT has been fine-tuned using Huggingface ^{xiii} and Tensorflow ^{xiv} on the Google Colab platform ^{xv}. The Google Colab platform was used for GPU access during training. After fine-tuning, the model was published on Huggingface. The model is publically available and can be found at ^{xvi}. The Results section (4) shows a comparison of the base model and the fine-tuned model.

3.3.2 Feature selection

All the features needed to be carefully extracted and organized in order to be used by the classifiers. The non-contextualized word embeddings have been prepared using the following method. After extracting the word embeddings of the tokens in the example sentence, the embeddings for the context and candidate skills have been separated using a separator (5). In total each data sample had 902 dimensions. 300 for each of the context and candidate skills columns, separated by 2 separators. The feature selection procedure for the non-contextual word embeddings is visualized in Figure 3.2.

The contextual embeddings have been prepared as follows. After loading the entire sentence into BERT, only the word embeddings for the candidate skills were extracted. After extracting the embeddings of the candidate skills these were appended to a list. The dimensions of

^{xiii}<https://huggingface.co/models>

^{xiv}<https://www.tensorflow.org/>

^{xv}<https://research.google.com/colaboratory/>

^{xvi}<https://huggingface.co/Ivo/emscad-skill-extraction>

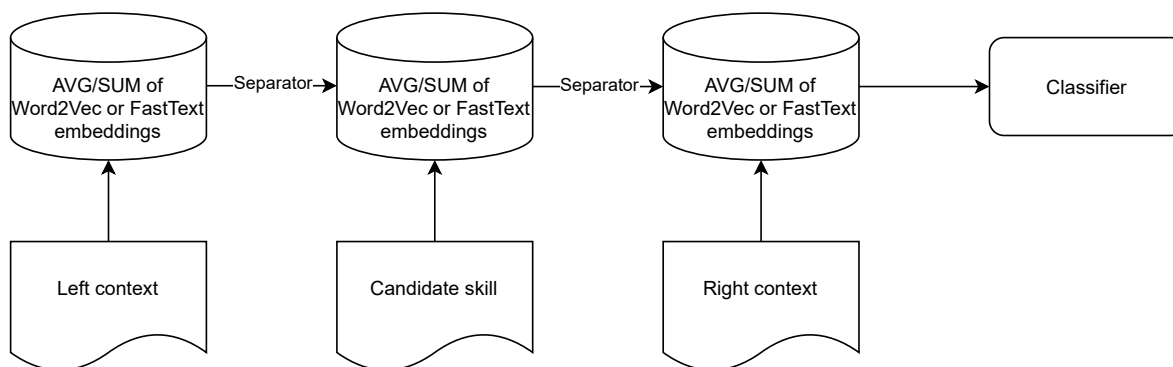


Figure 3.2: Non-contextual word embeddings feature selection

the candidate skills could be up to 4 tokens. Therefore the embeddings would have different dimensions which would result in issues further on. In the case there were less than 4 tokens, the resulting dimensions were padded with zeros. This created a uniform dimension. After following these steps the dimensions for the BERT embeddings were 3075 dimensions, consisting of a max of 768 embedding columns * 4 tokens + 3 separators. For ELMo, one embedding for each token has a dimension of 1024. Therefore the final dimensions for ELMo are $1024 * 4 + 3$. The ELMo embeddings were padded using the same methodology as BERT. Three separators were used for the contextual word embeddings because the embeddings for each of the candidate skill tokens were extracted. The number of tokens in the candidate skill columns could be up to four. This resulted in four embeddings for the candidate skill tokens, separated by three separators. Comparing this to the non-contextual word embeddings, the difference lies in the way the context is captured. In the case of the non-contextual word embeddings, the context was captured by summation or averaging all the word embeddings in the context columns as well as the candidate skill column. This resulted in the following format: three embeddings for the entire context columns and candidate skill column, separated by two separators. The feature selection process for the contextual word embeddings is visualized in Figure 3.3.

Another added feature was part-of-speech tagging (POS) and dependency parsing tagging (DEP). These two features were added in order to provide extra features to the model and comparing the performance before and after adding. The POS and DEP tags were generated for the candidate skills. For example, the POS tags for the candidate skill 'ability to actively

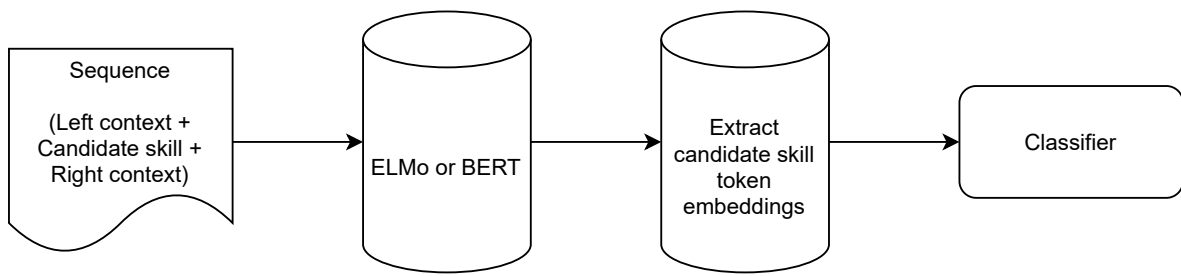


Figure 3.3: Contextual word embeddings feature selection

listen' would be NOUN, PART, ADV, and VERB. After generating the tags, the tags were counted and added as features. The DEP tags were captured using the same method. An example of the dummy variable is given in Table 3.4. The term 'other tags' stands for other POS and DEP tags.

Table 3.4: POS and DEP example

NOUN	PART	ADV	VERB	Other tags
1	1	1	1	0

3.3.3 Classifiers

After creating all the features which represent the N-grams, as well as the POS & DEP tags, the classifiers were trained on the annotated data to make the predictions. Multiple classifiers have been used to compare performance. The classifiers are listed below:

- Baseline
- Logistic regression (LR)
- Gradient boosting classifier (GBC)
- Stochastic gradient descent (SGD)
- Random Forest (RF)
- Support vector machine (SVM)
- Multi-layer Perceptron (MLP)

The hyperparameter settings for each of the classifiers are described in section 3.6.1. The baseline classifier was the 'DummyClassifier' from sci-kit learn. This classifier was used with the 'stratified' option. By using the stratified option, the classifier would make predictions according to the class balance ratio. This was important as there were imbalanced classes in the dataset. Each classifier was modeled with all four word embedding models.

3.4 Semi-supervised

In order to alleviate some of the annotation necessary to create the skill extraction pipelines, a semi-supervised variation of the context-aware sequence classification model is presented in this section.

Due to a large number of instances in the dataset (over a million), it was not possible to manually label each instance. One method to lower the burden of manual annotation is the use of a semi-supervised approach. The semi-supervised approach is called psuedo-labeling. Psuedo-labeling is the process of training a classifier on a manually labeled dataset and using the classifier to make predictions about the unlabeled dataset. Only predictions with a probability higher than 99% are kept and added to the training set. The 99% threshold was set in order to mitigate the effects of any mislabeling made by the classifier. After adding the predictions the classifier was trained again in order to make new predictions based on the originally annotated data and the predicted labels from the previous iteration. This process was repeated five times using the 'SelfTrainingClassifier' from sci-kit learn. A subset of the complete dataset was taken due to computational limitations. In total 100.000 samples were used. 20.836 samples of this subset were annotated. The other samples are used for psuedo-labeling. The highest performing solution of the sequence classification model is selected for the semi-supervised approach. One reason to use the semi-supervised approach instead of a skill-gazetteer is that the semi-supervised classifier is capable of learning representations for other words as well. For example, the semi-supervised classifier can learn what other hard skills are, even if these are not initially annotated. A skill-gazetteer cannot learn in this sense, it can only match with words in its taxonomy. The results are presented in section 4.

3.5 Named Entity Recognition

Named entity recognition (NER) is the process of identifying entities in natural text. These identities can consist of many types. For example, locations, person names, or organizations [20]. Named entity recognition models can be trained on specific tasks. The dataset preparation for the NER models is described in this section as well as the different models used for NER.

3.5.1 Dataset preparation

The first step for training the custom NER models was creating the dataset. The same dataset as described in Table 3.1 was used. Except this dataset needed to be transformed into a different format. The original format was left context, candidate skill, right context, and label. However, the custom NER models require data to be labeled per token. The dataset has been transformed by splitting the candidate skill column on the whitespaces between the tokens. Afterwards, a colon + the label was concatenated to the tokens. After adding the label to the tokens, the left context column + tokens with labels + right context columns were concatenated. This concatenated column now contained all the tokens + the label. After combining the tokens + labels, the column was tokenized and turned into a separate row. Now that the rows were created, the column was split on the colon, and the resulting column was the label column. Any rows in this column with NaN values were replaced with the *O* label which represents the not skill class. All these tokens with NaN values were tokens from the context columns. Other research [9] used BILOU encoding for the NER annotations. The acronym stands for beginning, inner, last, outside, and unit. These terms refer to the position of the tokens within the sentence. By using BILOU encoding more semantic information is stored within the annotations. BILOU encoding was not used for this dataset due to the structure of the dataset. All possible skill tokens are in the candidate skill column. This leads to the same BILOU encoding for each of the tokens, thus not adding semantic value for the classifiers. A sample of the output of the transformed dataset is presented in Table 3.5. This example shows the transformation from the second row in Table 3.1 to the tokenized data form. This dataset is used as input for the custom NER models. The class distribution

of the tokenized dataset is presented in Table 3.6.

Table 3.5: Tokenized dataset

Token	Label
is	O
expected	O
to	O
be	O
self	Soft Skill
motivated	Soft Skill
set	O
goals	O
and	O
meet	O

Table 3.6: Class distribution tokenized dataset

Class	Counts
Not Skill (O)	217.487
Soft Skill	7.764
Hard Skill	831

3.5.2 HashEmbedCNN

The HashEmbedCNN is a NER model created by SpaCy. The model consists of a Multi-HashEmbed layer which takes the subword features into consideration. The model also includes a MaxOutWindowsEncoder layer which consists of a Convolutional neural network layer and a layer-normalized maxout activation function [21]. SpaCy is a popular NLP library in Python used for many different use cases. The library supports many different languages, as well as different pipelines leading to different results. One advantage of SpaCy is that the library can be used to train specific custom NER models. The first step in training the HashEmbedCNN model was transforming the dataset described in 3.5.1 into the SpaCy format. SpaCy was recently updated from version 2 to 3. This meant that the dataset first needed to be transformed into the format for version 2 and then into the format for version 3.

Version 2 expects dictionaries that contain the sentence, keys to the entities, and the numerical position of the entities in the sentence. As an example, the sentence 'also have excellent write and *oral communication skills* demonstrate excellent analytical skill and' would have accompanying entities entries: (30, 34 Soft Skill); (35, 48 Soft Skill); and (49, 55 Soft Skill). The numbers point to locations of the three tokens: oral, communication, and skills. After converting the dataset to the SpaCy 2 format the dataset was transformed into the SpaCy 3 format. This format is binary and not readable for humans. Therefore, no example of the output is given. After preprocessing the dataset for SpaCy, separate models were trained and evaluated.

Two HashEmbedCNN models were trained, one using the SpaCy accuracy pipeline, and a different model using the SpaCy efficiency pipeline. The efficiency pipeline is optimized for efficiency. This results in faster inference, smaller model size, and a lower memory consumption. The accuracy pipeline is the opposite of the efficiency pipeline. Meaning that the accuracy pipeline is potentially larger and slower. However, the accuracy model has better performance. The performance metrics for the customer models are presented in 4.3.

Another benefit of using SpaCy is displaCy^{xvii}. DisplaCy can be used to visualize entities in the text and is straightforward to implement. Therefore, it is a suitable solution to be run in production environments. DisplaCy is used to present examples of the best performing pipeline.

3.5.3 Transformer

The following NER model is based on the transformer architecture and implemented using NERDA [22]. NERDA is another Python library that can be used for NER. NERDA is built on the Huggingface, Transformers, and Pytorch framework and is a state-of-the-art tool for NER. NERDA is unique because it supports many of the transformer architectures provided by Huggingface as well as pre-trained custom models. Another benefit of using NERDA is that it makes fine-tuning transformer architectures (BERT,ELMo, etc.) straightforward.

^{xvii}<https://spacy.io/usage/visualizers>

NERDA expects a different format than the dataset created in 3.5.1. The dataset was transformed into two lists, one which contains a list per sentence, and another list that contains the labels per sentence. For example, the sentence 'and *oral communication skills* demonstrate', would result in one list containing: (and, oral, communication, skills, demonstrate) and one list containing: (O, Soft Skill, Soft Skill, Soft Skill, O). These lists were used as input for the custom Transformer model. NERDA supports training on GPU. Therefore, the Transformer model was trained on the Google Colab Platform ^{xviii}. The results of the Transformer model are presented in 4.3.

3.5.4 Conditional Random Fields (CRF)

Another method for extracting entities from text is conditional random fields (CRF). CRF models work by assuming that data features are dependent on each other. For example, the CRF model checks the previous and next words of the candidate skill and learns that a token is a skill if it is between those words. The CRF model used the following features: the three tokens before and three tokens after the candidate skill as well as the length and suffix of these tokens. These features have been selected empirically. The CRF model has been implemented using the `sklearn-crfsuite` ^{xix} Python package. The dataset created in 3.5.1 was transformed into lists of sentences that contain tuples containing the token and label. For example, the sentence 'and *oral communication skills* demonstrate' would result in [(and, O), (Oral, Soft Skill), (communication, Soft Skill) etc.]. The performance of the CRF model is visualized in 4.3.

3.5.5 Token Classification Model

An alternative to the context-aware sequence classification model presented in 3.3 is to classify each individual token into not skill, soft skill, or hard skill. The dataset presented in Table 3.1 was used as input for the token classification model. Each token was transformed into embeddings using the aforementioned word embedding techniques. The non-contextual word embedding techniques have been implemented to compare the performance with the contextual word embedding techniques. The word embeddings and POS and DEP tags were

^{xviii}<https://research.google.com/colaboratory/>

^{xix}<https://sklearn-crfsuite.readthedocs.io/en/latest/index.html>

then used as input for the classifiers. An example of the input can be found in Table 3.7. The results of the token classification model are presented in section 4.3.

Table 3.7: Token classification example

Multiple embedding columns	Label
Numerical values	Not Skill
Numerical values	Soft Skill
Numerical values	Hard Skill

This dataset was the input for the classifiers presented in 3.3.3.

Fine Tuning BERT

BERT was fine-tuned for token classification as well. The 'bert-base-uncased' model was selected as the base model. The dataset created in 3.5.1 was transformed into lists of sentences and lists of labels. These lists were the input for the BERT fine-tuning process. BERT was fine-tuned using Huggingface and Tensorflow. The fine-tuning process was performed on Google Colab due to the accessibility of GPUs on the platform. The resulting model is publically available and can be accessed at ^{xx}.

3.6 Evaluation

This section describes the different evaluation tools used for the approaches. As well as the evaluation parameters. The first section describes the evaluation methods for the sequence classification model and the second section describes the evaluation methods for the NER models.

3.6.1 Sequence classification model parameters

The training dataset size was 80%. The test dataset size was 20%. The performance metrics used to evaluate the performance were precision, recall, and F1-score. The reasoning behind these metrics instead of using the accuracy is that most of the samples in the job descriptions dataset were not skills. Meaning that if the model would predict only the label 'not

^{xx}<https://huggingface.co/Ivo/emscad-skill-extraction-token-classification>

skill' the accuracy would be high but these results would not be useful. The macro values of the performance metrics were calculated for each of the iterations in the ten-fold cross-validation process. Afterwards the average was taken from these macro values. The ten-fold cross-validation steps have been taken in order to provide more realistic performance metrics while also reducing the risk of overfitting on the data. The following section describes the hyperparameters of the various classifiers. The LR classifier had the solver set to lbfgs. The GBC classifier had the `n_estimators` set to 100, `learning_rate` set to 1, and the `max_depth` set to 1. The SGD classifier had the loss function set to hinge and penalty to l2. The SVM classifier had the `decision_function` set to ovo. The MLP classifier had the solver set to lbfgs, `alpha` to 1e-5, `hidden_layer_sizes` to 15. The `random_state` for each of the classifiers was set to 456, and finally the `n_jobs` were set to unlimited.

Next to the evaluation metrics, and cross-validation steps, SHAP^{xxi} was used to check the feature importance of the sequence classification model. SHAP provides explainability for various machine learning classifiers. This helps to reduce the 'black box' surrounding machine learning predictions. SHAP is used to present the feature importance for two models: the model using FastText embeddings in addition with POS & DEP tags as input for the RF classifier and the model using fine-tuned BERT embeddings in addition with POS & DEP tags as input for the LR classifier. This decision was made because these models capture context differently. The former captures context by averaging the word embeddings (left context + candidate skill + right context) of the surrounding tokens. While the latter captures the context by scanning the entire sequence before assigning embeddings. Because the input of the two models consists of the word embeddings and POS & DEP tags, only the feature importance is shown. SHAP can also provide insights into each individual prediction. But these results only show which embedding column provided the most evidence to move the prediction to a specific class and this information is not insight full. SHAP was implemented by running the application on a sample of the dataset (1000 samples). SHAP is also used to show the feature importance of the best performing NER model. The results of SHAP are presented in section 4.

^{xxi}<https://github.com/slundberg/shap>

3.6.2 Named entity recognition parameters

Some of the NER approaches used neural networks at their core. Therefore the dataset needed to be split up into three parts: training, validation, and test. The training dataset size was 80%, the validation dataset size was 10%, and the test dataset size was also 10%. The training dataset was used for model training. The validation dataset was used for tuning the models hyperparameters. During the tuning of the hyperparameters, information of the validation set leaks into the model [23]. This can cause overfitting on the validation dataset. Therefore the final models were evaluated on the test dataset which the model had not seen during training. The same evaluation metrics were used as the sequence classification model.

The HashEmbedCNN pipelines were trained using the default hyperparameters collected by running the SpaCy config file. The HashEmbedCNN training procedure needed to be started from a terminal, therefore the HashEmbedCNN model could not be cross-validated. The models were only evaluated on the test dataset using the identical evaluation metrics as the sequence classification model.

The 'bert-base-uncased' model from Huggingface was used for fine-tuning the Transformer model. The number of epochs was set to 3, the warm-up steps were set to 500, the training batch size to 13, and finally, the learning rate was set to 0.0001. These parameters were set through experimentation. The model was evaluated using ten-fold cross-validation with the same evaluation metrics as the sequence classification model.

The algorithm of the CRF model was set to lbfgs, the max_iterations to 100, and all possible transitions was set to True. By setting this parameter to True, the CRF model can generate transitional features. The optimal values for the c1 and c2 parameters were determined using the RandomizedSearchCV function from sci-kit learn. The RandomizedSearchCV function updates the parameters of the coefficients of the L1 and L2 regularization according to the F1 scores during the cross-validation process. The output values were 0.07631744316674878 for the c1 parameter and 0.04351002093874303 for the c2 parameter. The model was finally evaluated using ten-fold cross-validation with the identical approach

as the sequence classification model.

3.7 Summary

The methodology chapter presented all the data collection, pre-processing, and the various approaches which are implemented for the skill extraction process. The EMSCAD dataset is used as job description dataset. The skill taxonomies are collected from EMSI, ESCO, and O*NET. Four word embedding techniques are compared in order to evaluate the performance of non-contextual and contextual word embeddings. POS & DEP tags are added as extra features for the models. Two main groups of approaches are implemented. The context-aware sequence classification model and the NER models. Both groups of approaches are evaluated using precision, recall, F1-score where the macro values are averaged during the ten-fold cross-validation process. SHAP was also implemented to show the feature importance of the classifiers.

4 Results and analysis

The methodology chapter provided insights into the collected datasets and pre-processing steps. Next to these two steps, two groups of approaches were presented: the context-aware sequence classification models and the NER models. Finally, the evaluation metrics and hyperparameters were described. This section is used to describe the results and analysis of those results for different pipelines. The first section describes the results and analysis of the context-aware sequence classification model. The second section shows if the semi-supervised approach is capable of learning skills that were not initially annotated. The third section presents the results of the various NER approaches.

4.1 Context-Aware Sequence Classification Model

After the creation of the context-aware sequence classification model, multiple evaluation metrics have been used to evaluate the performance. The evaluation corpus metrics were calculated on a corpus size of 20.836 labeled samples. The precision is presented in Figure 4.1, the recall is presented in Figure 4.2, and finally, the F1-score is presented in Figure 4.3. The Figures present the average of the macro evaluation metrics after ten-fold cross-validation. The highest precision achieved is 0.93 with fine-tuned BERT embeddings and the random forest classifier. The highest recall achieved is 0.86 with fine-tuned BERT embeddings and the logistic regression classifier. Finally, the highest F1-score achieved is 0.86 with fine-tuned BERT embeddings and the logistic regression classifier. The addition of the POS & DEP does improve the precision, recall, and F1-scores. However, this difference is very slight.

The baseline confusion matrix is presented in Figure 4.4, and the confusion matrix for the model using fine-tuned BERT embedding, POS & DEP tags, and the logistic regression classifier is presented in Figure 4.5. The confusion matrices were created by fitting the classifiers

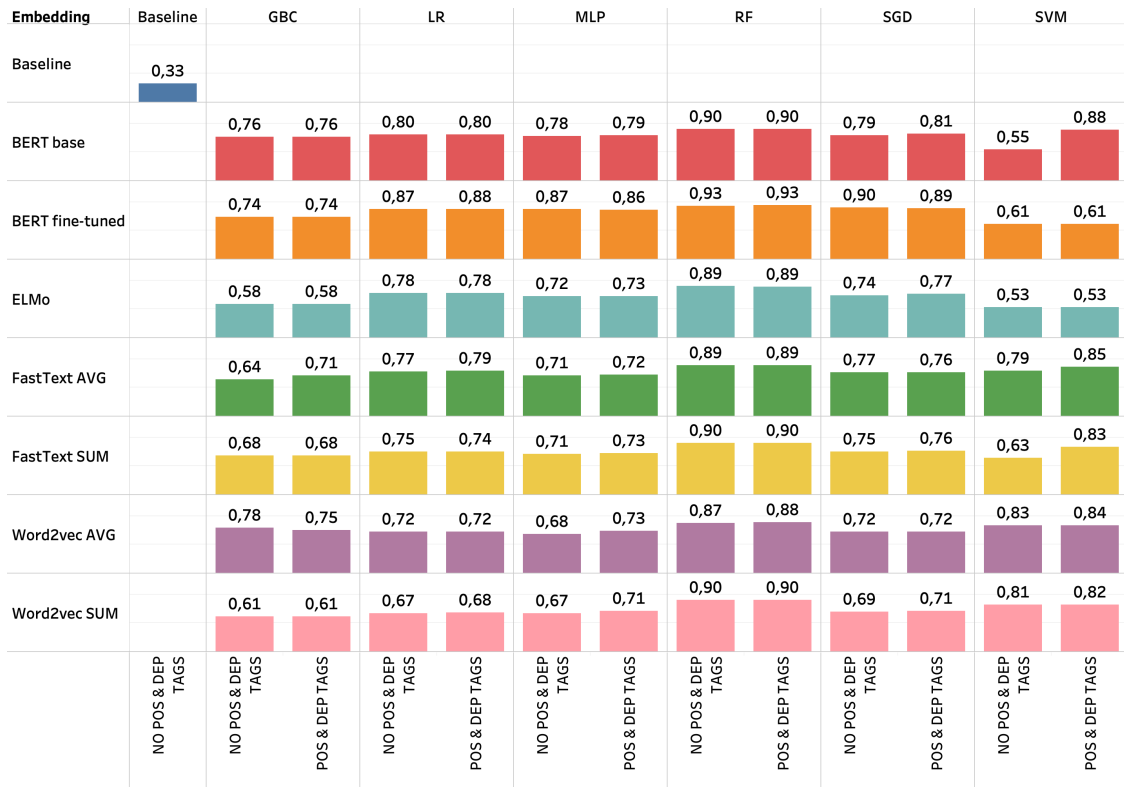


Figure 4.1: Precision sequence classification model

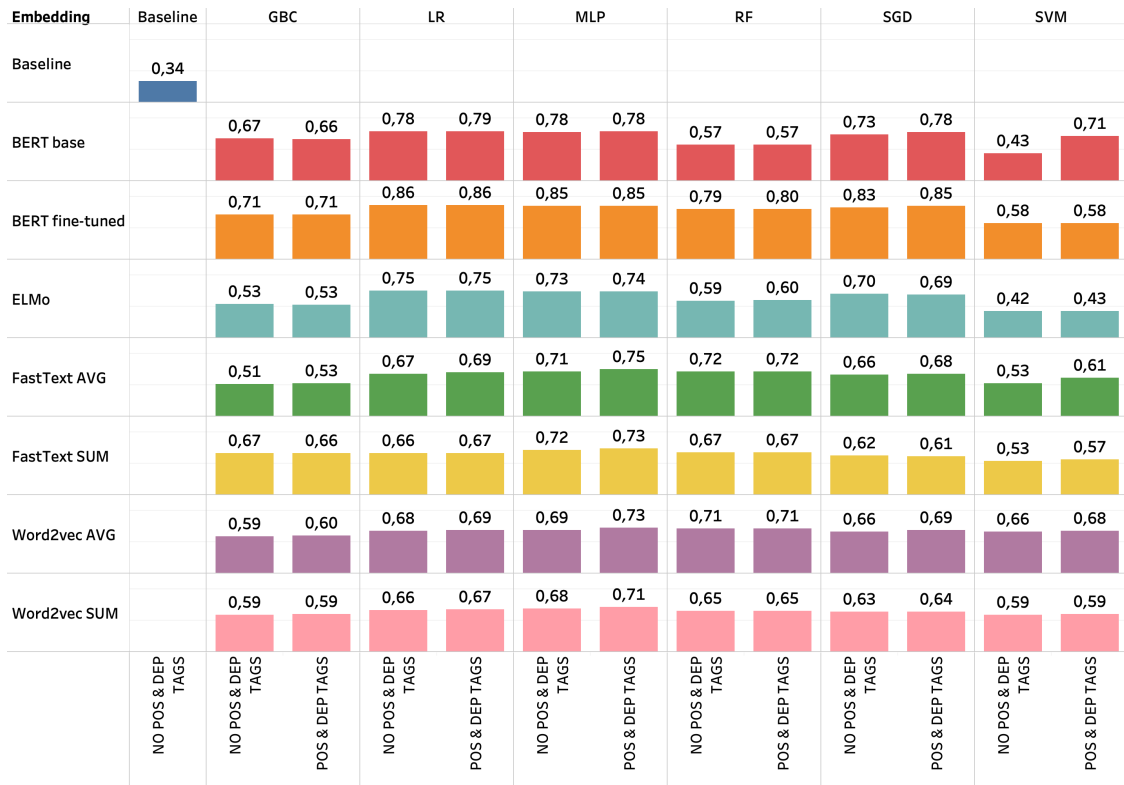


Figure 4.2: Recall sequence classification model

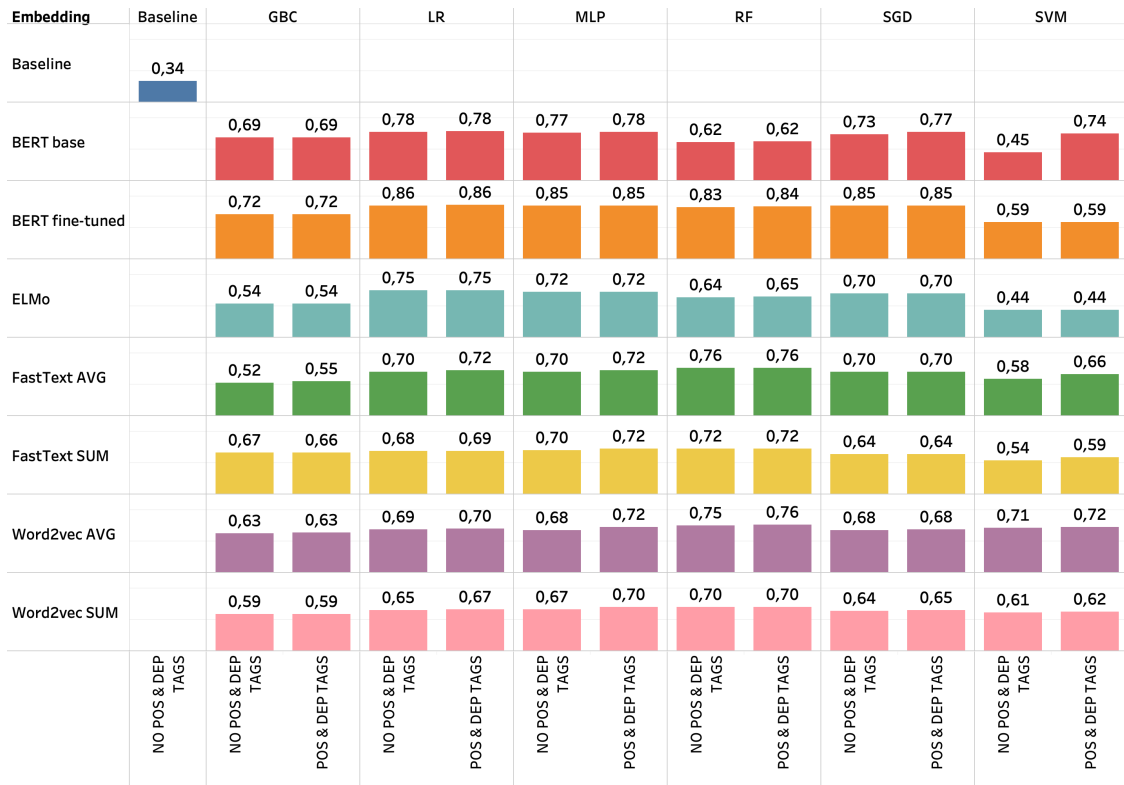


Figure 4.3: F1-score sequence classification model

on 80% of the data while making the predictions on 20% of the data. Only the confusion matrix for the best-performing model is shown next to the baseline model. The baseline model is used to compare the performance of the sequence classification model to a stratified random guessing model. Finally, example predictions are shown of the sequence classification model in Figure 4.6.

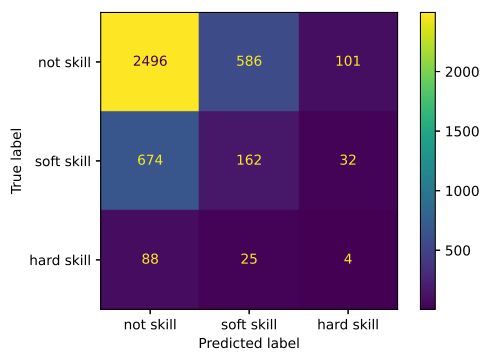


Figure 4.4: Confusion matrix baseline sequence classification 20% test data

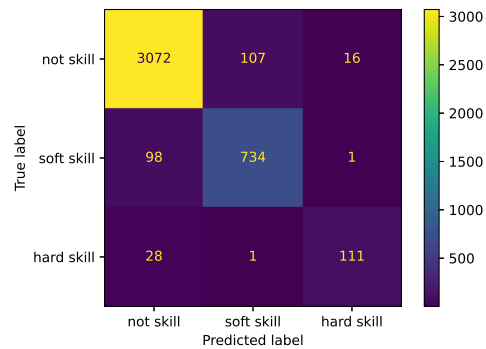


Figure 4.5: Confusion matrix BERT fine-tuned POS DEP LR sequence classification model 20% test data

The results indicate that the logistic regression classifier with fine-tuned BERT embeddings and POS & DEP tags features achieved the best performance. The results also indicate that fine-tuning BERT improves the performance of the model significantly. What is interesting to see is that the ELMo performance is very similar to the traditional methods (Word2Vec and FastText). This could be explained by the usage of the small ELMo model. Using the medium or large model might improve the performance of ELMo. All of the approaches outperform the baseline classifier. This shows that the sequence classification model adds value to the skill extraction process.

The confusion matrix shows that the two most correct predictions made are for not skills and soft skills. These two classes are also the largest classes in the manually annotated dataset. The confusion matrix clearly shows a small number of annotated hard skills in the test dataset. A reason for the class imbalance is the use of the soft skills presented in [5]. This dataset contained 7220 annotated samples. This in comparison to the manually annotated samples(13616) would make the sequence classification model perform better in predicting soft skills than hard skills. Overall comparing the confusion matrix of the baseline in Figure 4.4 with the confusion matrix of the pipeline in Figure 4.5 we can see that the performance of the pipeline outperforms the baseline tremendously.

Figure 4.6 shows a sample of predictions made by the sequence classification model. The first three examples are correctly classified. However, the fourth and fifth samples are not. This could be due to insufficient annotated samples. The final two examples have label -1. This means that these were not originally annotated. These examples show that the model is able to learn soft and hard skills which were not originally in the annotated dataset as well as the taxonomy.

Figure 4.7 and Figure 4.8 show the feature importance results from SHAP. Figure 4.7 shows the top 20 features with the most predictive power for the three classes. The results of the graph indicate that the BERT embeddings have the most predictive power for predicting the classes. These results are however difficult to interpret because the embeddings in each of

left_context	candidate_skill	right_context	label	predictions
agency recruiter	having credibility	and strong	not skill	not skill
good knowledge of microsoft application excell...	communication skills	good keyboard speedsregistering with	soft skill	soft skill
cms	drupal	or joomla useful	hard skill	hard skill
as a senior	java	developer you will be part	hard skill	not skill
proactively	communicating	with owners and tenants	soft skill	not skill
winning mind set and	entrepreneur spirit	to fill entry level	-1.0	soft skill
dysfunction knowledge of	behavioral management techniques	general knowledge of	-1.0	hard skill

Figure 4.6: Sequence classification model predictions

the embedding columns are influenced by other tokens in the sequence. One insight derived from this graph is that the POS & DEP tag features do not add significant value. This insight was also visible from the evaluation metrics, however, these results further strengthen this conclusion. Figure 4.8 shows the top 20 features with the most predictive power for the FastText AVG in addition to POS & DEP tags as input for the RF classifier results. The results indicate that the most predictive power is in the embeddings from the candidate skill tokens. Therefore the method of capturing the context to the left and right of the candidate skill tokens is not the most significant in predicting the three classes. Similar to the SHAP results for the LR model, the graph shows that the addition of the POS & DEP tags does not have significant predictive power as seen in the evaluation metrics.

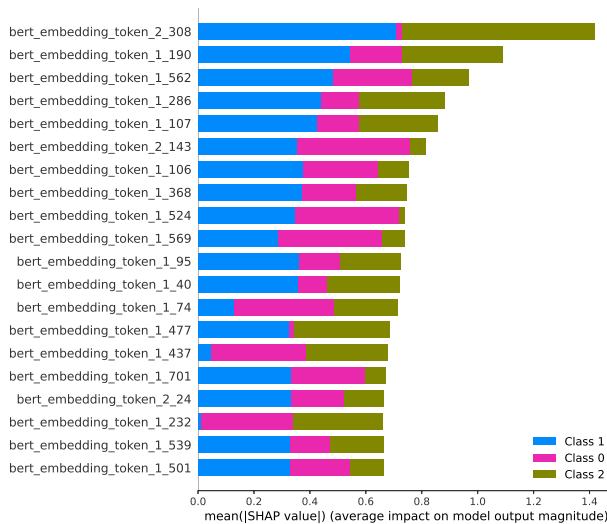


Figure 4.7: SHAP results BERT fine-tuned POS & DEP tags LR sequence classification model

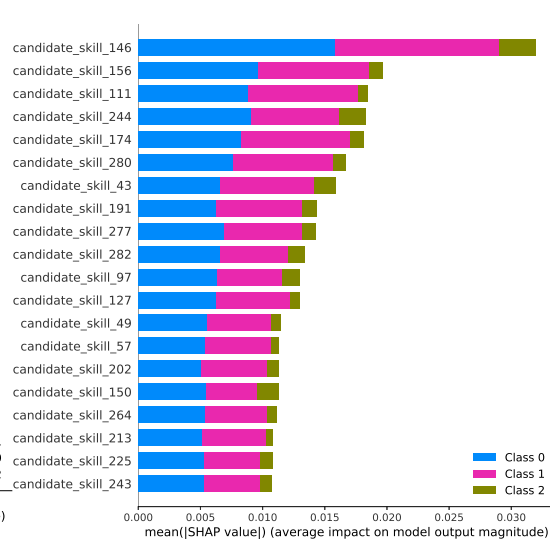


Figure 4.8: SHAP results FastText AVG POS & DEP tags RF sequence classification model

To implement the top-performing context-aware sequence classification model as a continuous pipeline various steps need to be taken. The steps are summarized in the following

list.

1. Pre-process the job description
2. Create N-grams of the job description
3. Determine the context of the N-grams
4. Create fine-tuned BERT embeddings of the candidate skills N-grams using the entire sequence (concatenated context columns and candidate skill column)
5. Add the POS and DEP Tags
6. Classify each sequence using the LR classifier

The results of the sequence classification models indicate that the state-of-the-art models are able to predict hard and soft skills from the EMSCAD dataset. This answers the second sub-research question: **How to develop one state-of-the-art robust model for extracting both hard and soft skills?**. The results also show that the best performing sequence classification model is able to learn skills which were not in the taxonomies as well as in the annotated dataset. These results are very insight full as it showcases the benefits of implementing this model in comparison to a skill gazetteer. This answers the third sub-research question: **Can the state-of-art model learn new hard and soft skills?**

4.2 Semi-supervised Sequence Classification Model

This section describes the results of the semi-supervised approach. The semi-supervised approach was only tested with the best performing sequence classification model: fine-tuned BERT embeddings with POS & DEP tags and a logistic regression classifier. A subset of 100,000 samples was used due to computational limitations. The main reason for experimenting with a semi-supervised approach was investigating if the sequence classification model could learn new skills which were not annotated in the Gold standard dataset. Figure 4.9 shows predictions made by the semi-supervised sequence classification model. The -1 in the label column indicated that these samples were not annotated from the beginning.

Whilst the predictions column shows the predicted label made by the sequence classification model.

left_context	candidate_skill	right_context	label	predictions
highly professional pharmacist with a	forward thinking	approach we invite you to	-1	soft skill
positive and energetic	phone skills	excellent listening skills	-1	soft skill
excellent problem solving skills strong	ability to multi task	self motivated strong interest in	-1	soft skill
to read and understand	plc logic	and electrical diagrams or	-1	hard skill
knowledge in linux and	web logic	10 3 and 11excellent	-1	hard skill
programming skills in	objective c	java or both	-1	hard skill
strategies and do benchmarking on	proof of concept	solutions	-1	soft skill
a bookkeeping company		located	-1	hard skill

Figure 4.9: Semi-supervised sequence classification model predictions

Figure 4.9 shows that the first six samples are classified correctly. The candidate skills were manually checked to see if these were in the manually annotated dataset which they were not. This shows that the sequence classification model is able to automatically learn other skills. A downside of the semi-supervised approach is that the model will most likely learn errors as well. This is highlighted in the last two samples. Both these samples are not skill.

The model was not further evaluated with the evaluation metrics and the confusion matrices because all of the data was used to train the classifier. Therefore testing on the same data would not showcase the performance on real-world data.

4.3 Named Entity Recognition Models

This section describes the results and analysis of the HashEmbedCNN models, Transformer model, and the token classification model.

4.3.1 HashEmbedCNN

The first NER approach for skill extraction was the HashEmbedCNN model. Two HashEmbedCNN pipelines were trained: accuracy and efficiency pipelines. The evaluation metrics are calculated using the macro approach based on the 10% test data. The precision of the

accuracy pipeline was 0.66 . The recall of the accuracy pipeline was 0.62 . The F1-score of the accuracy pipeline was 0.64 . The precision of the efficiency pipeline was 0.65 . The recall of the efficiency pipeline was 0.58 . The F1-score of the efficiency pipeline was 0.60 .

Next to the evaluation metrics, the confusion matrices are plotted. Figure 4.10 show the confusion matrix of the accuracy pipeline. Figure 4.11 shows the confusion matrix of the efficiency pipeline. The baseline model is plotted in Figure 4.12. Both the pipelines outperform the baseline model significantly. The accuracy model seems to outperform the efficiency model very slightly. The performance of both HashEmbedCNN pipelines needs to be improved for the hard skill class. The pipelines classify most hard skills incorrectly as not skill.

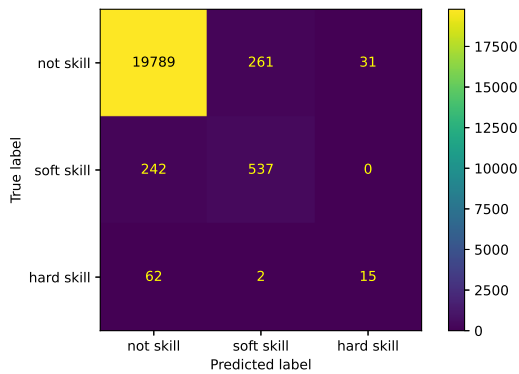


Figure 4.10: Confusion matrix HashEmbedCNN accuracy 10% test data

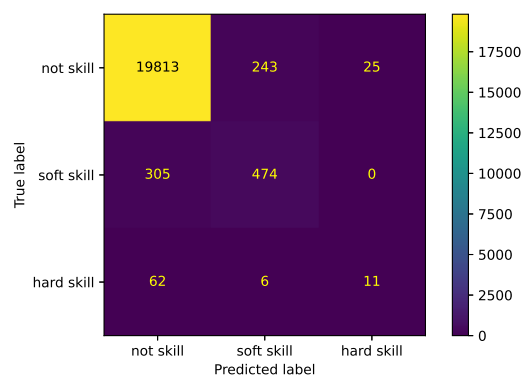


Figure 4.11: Confusion matrix HashEmbedCNN efficiency 10% test data

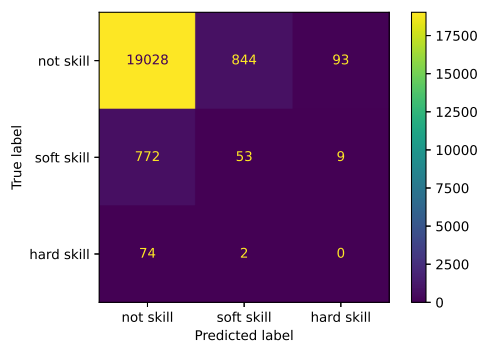


Figure 4.12: Confusion matrix baseline 10% test data

Another benefit of SpaCy is DisplaCy. The predictions made by the pipelines can be visualized in the actual text. An example of this is visualized in Figure 4.13. DisplaCy makes implementing the trained models in a production environment a straightforward process.

Example Job Description:
 Knowledge of **statistics Hard_Skill** such as skewness, kurtosis, probability density function, data normalization techniques, correlation techniques
 Translate complex concepts into implications for the business via excellent **communication Soft_Skill** **skills Soft_Skill**, both verbal and written

Figure 4.13: DisplaCy example

Next to the example of DisplaCy is an example of hard and soft skills learned by the accuracy HashEmbedCNN model. These are visualized in Figure 4.14. Both the soft skill and hard skill were not present in the taxonomy and in the annotated dataset. This showcases that the HashEmbedCNN model is able to learn new skills.

you will be warm and have a **welcoming Soft_Skill** **personality Soft_Skill**
 Knowledge of **RDBMS Hard_Skill** concepts

Figure 4.14: Learned hard and soft skills HashEmbedCNN model

4.3.2 Transformer

The second NER approach for skill extraction was the Transformer model. The Transformer model was fine-tuned on the dataset to increase the performance. The evaluation metrics are calculated on 10% of the test data. While the model was trained and validated using the macro variant of the evaluation metrics with ten-fold cross-validation. The precision of the Transformer model was 0.76 . The recall was 0.74 , and the F1-score was 0.75 .

The confusion matrix of the Transformer model is plotted in Figure 4.16. This confusion matrix was created on 10% of the test data. The baseline confusion matrix of the NER model is plotted in Figure 4.15.

These results indicate that fine-tuning a state-of-the-art word embedding technique on a specific task drastically increases the performance. The Transformer model achieved an F1-score of 0.75 whilst the HashEmbedCNN model achieved an F1-score of 0.64 . The Trans-

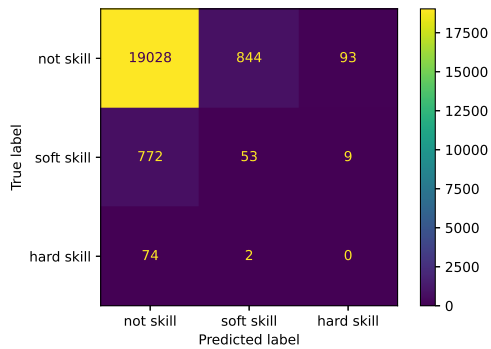


Figure 4.15: Confusion matrix baseline 10% test data

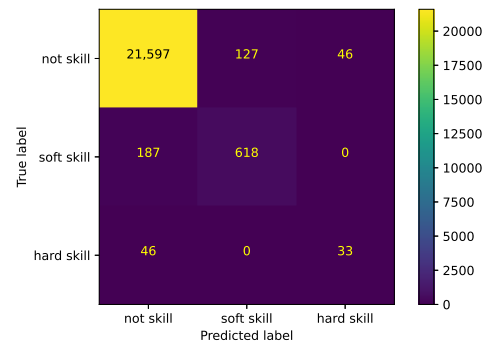


Figure 4.16: Confusion matrix Transformer 10% test data

former model predicted most of the soft skills correctly. However, the model has difficulty predicting hard skills. 46 hard skills are predicted as not skills whilst 33 hard skills are predicted correctly. This could be due to insufficient hard skill samples. Comparing the Transformer model to the HashEmbedCNN does show that the Transformer model outperforms the HashEmbedCNN in all three classes. A downside of the Transformer model is that there is no implementation of DisplaCy. Making it slightly more difficult to implement in a production environment.

Figure 4.17 shows some predictions made by the Transformer model. Both these hard skills and soft skills were not in the annotated dataset or taxonomies showcasing the learning capabilities of the model. The figure also shows, for example, that the Transformer model incorrectly predicted storage arrays as not skill. This is however a hard skill. Increasing the number of annotated hard skills could resolve this issue.

token	predictions	token	predictions
knowledge	O	check	O
of	O	Ability	Soft_Skill
Storage	O	to	Soft_Skill
Arrays	O	effectively	Soft_Skill
SANS	Hard_Skill	work	Soft_Skill
VMware	O	individually	Soft_Skill
Administration	O	or	O
Project	O	in	O

Figure 4.17: Learned hard and soft skills Transformer model

4.3.3 CRF

The third NER approach created was the CRF model. The CRF model was validated using the macro variant of the evaluation metrics as well as ten-fold cross-validation. The CRF model achieved a 0.61 precision score, 0.49 recall score, and 0.52 F1-score. The confusion matrix of the CRF model is visualized in Figure 4.19. The baseline confusion matrix is visualized in Figure 4.18.

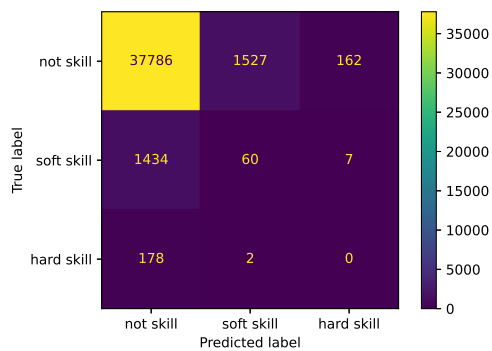


Figure 4.18: Confusion matrix baseline 20% test data

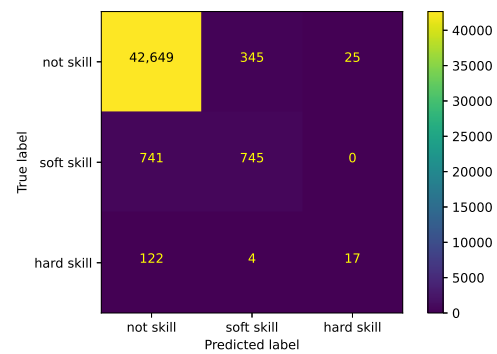


Figure 4.19: Confusion matrix CRF 20% test data

Comparing the CRF model to the baseline model, the results indicate that the CRF model is better in predicting hard and soft skills than the baseline model. Similar to the other models, the CRF model has difficulty in correctly predicting hard skills. Comparing the CRF model to the token classification model, the CRF model seems to have low performance. The CRF model is a traditional approach, and the results show that the state-of-the-art techniques have higher performance, making these methods more appropriate for use in production.

Figure 4.20 shows a sample of predictions made by the CRF model. The hard skill *Puppet* and soft skill *pitching* were not in the annotated dataset or in the taxonomies. Therefore, the CRF model learned that these tokens are skills. The figure also shows that MS SQL is classified as not skill. This is however a hard skill.

token	predictions	token	predictions
.NET	O	success	O
MS	O	Experience	O
SQL	O	of	O
JavaScript	Hard_Skill	creating	O
Puppet	Hard_Skill	pitching	Soft_Skill
to	O	and	O
name	O	converting	O
platform	O	new	O
where	O	business	O
ad	O	Ownership	O

Figure 4.20: Learned hard and soft skills CRF model

4.3.4 Token Classification Model

The following section describes the final NER approach, the token classification model. The same approach is used to present the results as the context-aware sequence classification model. The precision scores are visualized in Figure 4.21. The recall scores are visualized in Figure 4.22. The F1-scores are visualized in Figure 4.23. The metrics are calculated using ten-fold cross-validation of the macro variant of the performance metrics. The highest precision achieved was 0.95 with the BERT base model embeddings, POS & DEP tags, and a random forest classifier. The highest recall achieved was 0.87 with fine-tuned BERT embeddings, POS & DEP tags, and a multi-layer perceptron classifier. The highest F1-score achieved was 0.88 with fine-tuned BERT embeddings, POS & DEP tags, and a support vector machine classifier. The precision, recall, and F1-score figures show that the addition of the POS & DEP tags does not lead to a significant performance increase. However, the performance does increase slightly.

The baseline confusion matrix of the token classification model is shown in Figure 4.24 and the confusion matrix of the token classification model with fine-tuned BERT embeddings, POS & DEP tags, and the support vector machine classifier is shown in Figure 4.25. In comparison to the baseline model, the token classification model outperforms it significantly. The model classifies most of the soft skills correctly. However, the same issue as the context-aware sequence classification model arises with a low number of annotated hard skills in the test dataset.

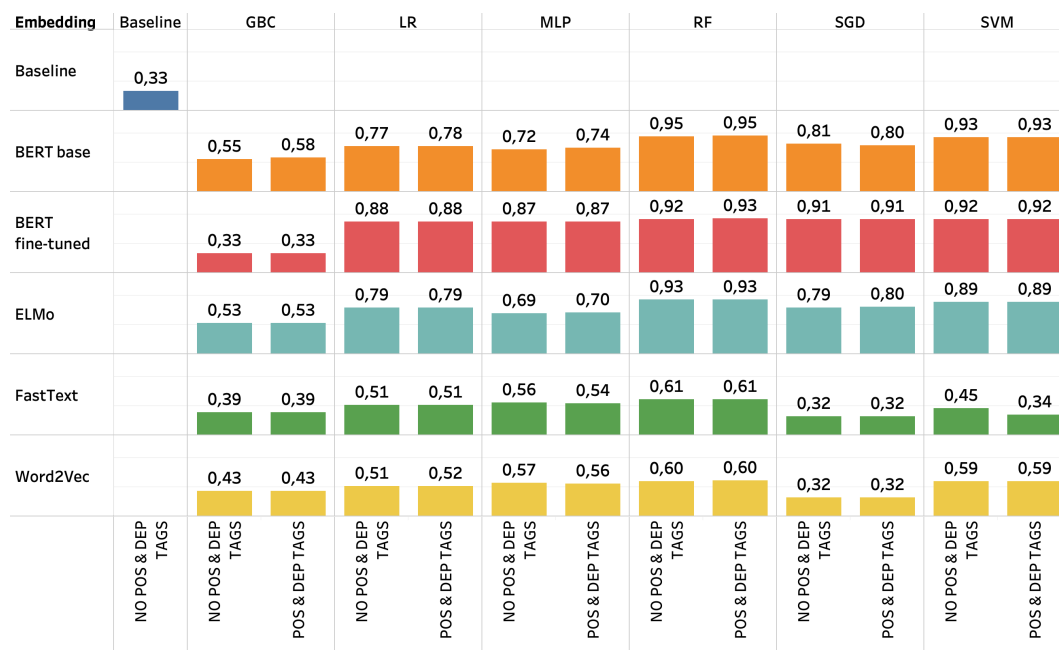


Figure 4.21: Precision token classification model

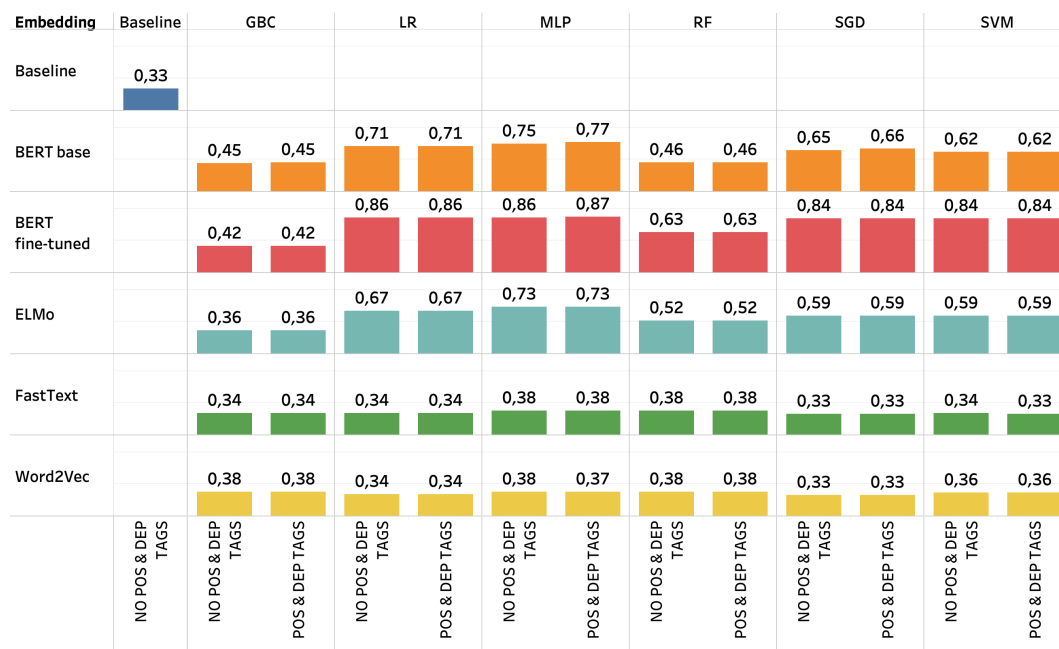


Figure 4.22: Recall token classification model

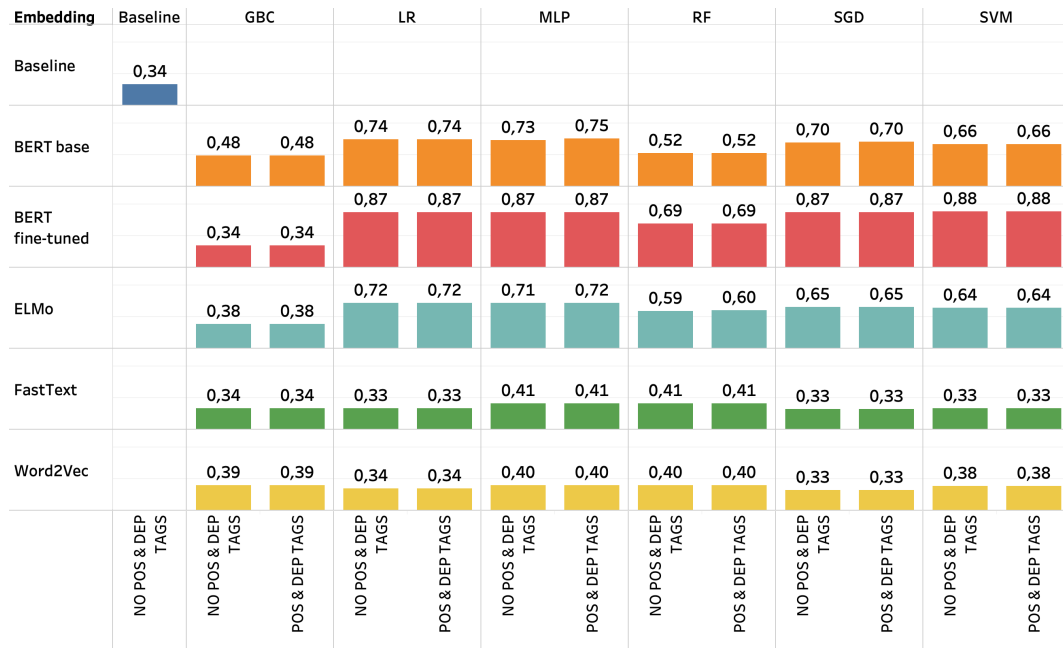


Figure 4.23: F1-score token classification model

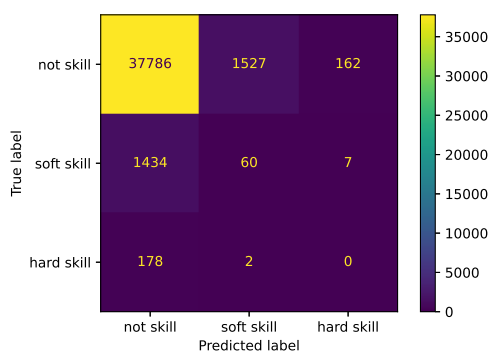


Figure 4.24: Confusion matrix baseline token classification 20% test data

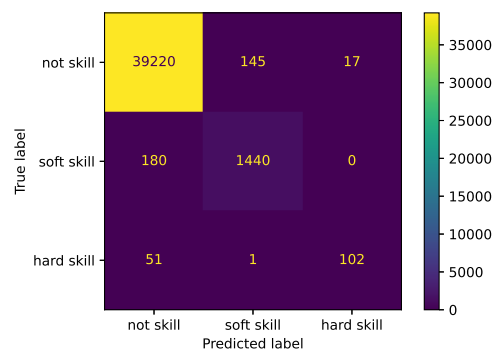


Figure 4.25: Confusion matrix fine-tuned BERT POS DEP SVM token classification model 20% test data

Figure 4.26 shows a sample of predictions made by the token classification model. The first three examples are correctly classified. However, the other samples are not. This could be due to insufficient annotated samples or the transformation of the dataset. Because all the tokens in the left context and right context were labeled as not skill. However, these columns could contain skill tokens, but these tokens were labeled as *not skill*.

token	label	predictions
consistent	not skill	not skill
enthusiasm	soft skill	soft skill
mint	hard skill	hard skill
enthusiastic	not skill	soft skill
authentication	not skill	hard skill
dynamic	soft skill	not skill
jquery	hard skill	not skill
crisis	hard skill	soft skill

Figure 4.26: Token classification model predictions

Next to the example predictions shown in the previous figure, Figure 4.27 shows a sample of learned hard and soft skills. These skills were not in the taxonomies or in the annotated dataset. This showcases that the token classification model is able to learn skills outside of the training data.

token	predictions	token	predictions
team	not skill	team	not skill
knowledge	not skill	members	not skill
of	not skill	the	not skill
bpm	hard skill	ability	soft skill
concepts	not skill	to	soft skill
experience	not skill	stay	soft skill
on	not skill	on	soft skill
presentations	not skill	task	soft skill
to	not skill	and	not skill
an	not skill	work	not skill

Figure 4.27: Learned hard and soft skills token classification model

As for the sequence classification model, SHAP was implemented to show the feature importance of the best performing token classification model. The results are presented in Figure 4.28. The results show the top 20 features with the most predictive power for the three classes. The graph shows that the fine-tuned BERT embeddings have the most predictive power. Similar to the sequence classification models, the graph shows that the addition of the POS & DEP tags does not lead to a significant performance increase. This was also shown in the comparison of the evaluation metrics.

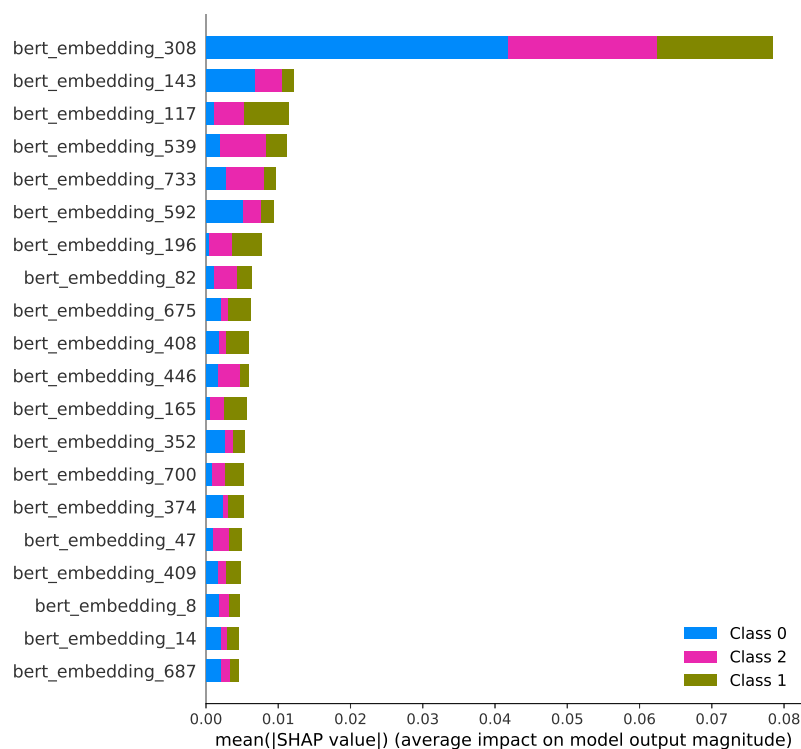


Figure 4.28: SHAP results BERT fine-tuned POS & DEP tags SVM token classification model

An advantage of the token classification approach is that implementing the solution in a production environment is straightforward. The steps needed to implement the top-performing NER model in a production environment are summarized in the following list:

1. Pre-process the job description
2. Create fine-tuned BERT embeddings for each of the individual tokens
3. Add the POS and DEP Tags

4. Classify each token with an SVM classifier

The results of the token classification model clearly show the comparison of non-contextual and contextual aware models. The comparison shows that by using a context-aware system the evaluation metrics are greatly improved. The best performing non-contextual model achieved an F1-score of *0.41* whereas the best performing contextual model achieved an F1-score of *0.88*. This result answers the first sub-research question: **What impact has the context in classifying terms as soft/hard skill or not skill?** Another insight is that all the NER models are capable of predicting both hard and soft skills. This answers the second sub-research question: **How to develop one state-of-the-art robust model for extracting both hard and soft skills?** However, the performance for the hard skill class could be improved. Furthermore, the results also show that the HashEmbedCNN, Transformer, CRF, and token classification models are able to learn hard and soft skills which were not in the taxonomies and in the annotated dataset. These results answer the third sub-research question: **Can the state-of-art model learn new hard and soft skills?**

4.4 Summary

The results and analysis section highlighted all the results achieved with the various approaches. The best-performing context-aware sequence classification model was built using fine-tuned BERT embeddings, additional POS & DEP tags, as input for an SVM classifier. This model achieved an F1-score of *0.86*. The model was also able to learn new hard and soft skills.

The NER approaches cannot be directly compared. The HashEmbedCNN and Transformer models were evaluated on 10% of the data. While the CRF and token classification model were evaluated on 20% of the test data. Comparing the HashEmbedCNN models and the Transformer model, the Transformer model has the highest performance, achieving an F1-score of *0.75* in comparison to the F1-score of *0.64*. Comparing the CRF model and the token classification model, the token classification model with fine-tuned BERT embeddings, POS & DEP tags, as input for an SVM has the highest performance (*0.88* F1-score). The context-

aware sequence classification model and token classification model cannot be directly compared. For example, the soft skill: *ability to learn* would be classified as one soft skill in the sequence classification model whereas the token classification model would classify three tokens as soft skill. This would bias the evaluation metrics. Furthermore, the results of the token classification answer the first sub-research question. By taking the context into consideration the performance of the model is significantly improved showcasing the importance of the context in correctly classifying tokens. The results also show that building one robust model for predicting both hard and soft skills can be achieved by careful feature selection and using state-of-the-art techniques. Finally, the results show that all the proposed models are able to learn hard and soft skills which were not in the taxonomies or in the annotated dataset.

5 Conclusion and future work

This final chapter presents the conclusion as well as the future work and limitations. The goal of this thesis was to develop a context-aware state-of-the-art model that could predict hard and soft skills from job descriptions. The previous chapter showed sequence classification and NER approaches that can be used for this task.

5.1 Conclusion

The dataset used for this thesis was the EMSCAD dataset. The dataset consisted of fraudulent and legitimate job descriptions where only the legitimate job descriptions were used. The first step consisted of pre-processing the text in order to make it suitable for analysis. After the preprocessing steps, the job descriptions were transformed into N-grams in order to capture the context of the candidate skill tokens as well as the candidate skills. The candidate skills tokens were first selected by using a skill gazetteer with skills from various sources. After building the dataset, the dataset was manually annotated and supplemented with a list of soft skills collected from [12]. The total dataset size was 20.836 labeled samples.

After creating the dataset, various approaches were tested and evaluated for the skill extraction process. The approaches can be grouped into two groups: sequence classification and NER. The sequence classification utilized the annotated dataset described in the previous paragraph and utilized both contextual and non-contextual word embedding techniques in order to represent the linguistic meaning of the tokens in a numerical format. The non-contextual word embedding techniques consisted of Word2Vec and FastText. The contextual word embedding techniques consisted of ELMo and BERT. In order to capture the context as well as the candidate skills with the non-contextual word embedding techniques, all word vectors were either summed or averaged. The context was captured with the contextual

techniques by loading the entire sequences into the models and only extracting the word embeddings for the candidate skill tokens. In addition to the word embeddings, POS & DEP tags were added in order to provide more information to the state-of-the-art machine learning classifiers. In total, six classifiers were compared and evaluated. Finally, the top performing sequence classification model was used in a semi-supervised approach in order to see if the model could learn new skills this way as well.

The presented NER models were trained on a transformed version of the N-gram dataset. The N-gram dataset was transformed into a tokenized form where each row consisted of a token and its appropriate label. Four models were created and evaluated on this dataset: HashEmbedCNN, Transformer, CRF, and the token classification model. Two HashEmbedCNN models were created and evaluated. One model trained for accuracy and one model trained for efficiency. The Transformer model was based on fine-tuned BERT embeddings. The CRF model utilized the three tokens before and after a given candidate skill as well as the length and suffix of these tokens. Finally, the token classification model used the identical word embedding techniques, addition of POS & DEP tags, and classifiers as the sequence classification model.

The results of the sequence classification model indicate that the best performing sequence classification model utilized fine-tuned BERT embeddings, in addition to POS & DEP tags, with a logistic regression as classifier. The model achieved a *0.88* precision score, *0.86* recall score, and *0.86* F1-score. These metrics were calculated using the macro variant after ten-fold cross-validation. The results of the HashEmbedCNN, Transformer, CRF and token classification model could not be directly compared due to different test dataset sizes. However, the Transformer model outperformed the HashEmbedCNN model and the token classification model outperformed the CRF model. The Transformer model was based on fine-tuned BERT embeddings and achieved a *0.76* precision score, *0.74* recall score and *0.75* F1-score. The token classification model utilized fine-tuned BERT embeddings, in addition to POS & DEP tags as input for a support vector machine. This model achieved a precision score of *0.92*, recall score of *0.84* and a F1-score of *0.88*. Similar to the sequence classi-

fication model these metrics were calculated using ten-fold cross-validation of the macro variant of the evaluation metrics. The best performing sequence classification model and token classification model cannot be compared directly as the results are biased. The sequence classification model would classify *ability to learn* as one soft skill whereas the token classification model would classify as three soft skills.

5.2 Limitations

This section is used to present the limitations of the proposed models. The results of the proposed models indicate that the models are capable of extracting hard and soft skills from job descriptions. However, the models also have limitations. The best performing sequence classification model utilized the N-gram dataset to capture context and make predictions accordingly. Implementing this model in a production setting would be computationally expensive as the text first needs to be thoroughly pre-processed. This in comparison to the token classification model where the text only needs to be tokenized. The performance for predicting the hard skill class for both the sequence classification model and token classification model needs to be improved in order to successfully use these in a production setting. One reason for explaining this drop in performance is insufficient annotation. Only a sample of the created dataset was manually annotated due to time constraints. By increasing the number of annotations the performance of the hard skill class could be improved.

Another limitation that arises with the use of machine learning systems is that of ethical and privacy limitations. The models could predict some sequences or tokens which contain personal information as hard or soft skill. In a production setting, this could lead to personnel seeing sensitive information instead of extracted skills.

5.3 Future work

This section describes further interesting research and applications for the models proposed in this thesis. Future research work may benefit from the usage of different sentence forms such as lemmatized forms or forms without stop words. These results could then be com-

pared to the results in this thesis and evaluated if this increases or decreases the performance. Furthermore, since there was a class imbalance in the dataset, different sampling techniques could be used to resolve this issue. The performance of this system could then be compared to the systems in this thesis.

Another interesting area would be to increase the granularity of the classes. The classes could, for example, be increased to language skills, programming skills, construction skills etc. This would provide more information for the candidate selection and matching process and could lead to better matches with potential employees and employers.

Similar to the previous statement, different neural networks could be used to predict the respective classes. The performance could then be compared to the performance achieved in this thesis.

Taking a broader perspective, the systems proposed in this thesis could be one building block towards a fully automated recruitment process. By incorporating multiple systems, multiple tasks could be automated. This thesis focused specifically on skill extraction. However, other researchers created systems for education information extraction [9]. Another application for automating the recruitment process was presented in [24]. These researchers created a system that could classify a resume into a specific class. The following paper [25] created a system that incorporated the resume of an applicant and their social media presence in order to background check the resume. By combining all these systems into perhaps a job recommender system, a lot of the recruitment tasks could be automated. A different issue that arises in the screening and matching process is bias towards certain demographics [26]. By carefully creating an automated system, these issues could perhaps be resolved. This could in turn create a lot of value for society in general and lead towards a more fair job market.

All the code used in this thesis can be found at the following public GitHub page: https://github.com/iwmaas/emscad_skill_extraction

Bibliography

- [1] J. Hirschberg and C. D. Manning, “Advances in natural language processing,” *Science*, vol. 349, no. 6245, pp. 261–266, 2015.
- [2] M. Zhao, F. Javed, F. Jacob, and M. McNair, “Skill: A system for skill identification and normalization,” in *Proceedings of the twenty-ninth AAAI conference on artificial intelligence*, pp. 4012–4017, 2015.
- [3] M. Bastian, M. Hayes, W. Vaughan, S. Shah, P. Skomoroch, H. Kim, S. Uryasev, and C. Lloyd, “Linkedin skills: large-scale topic extraction and inference,” in *Proceedings of the 8th ACM Conference on Recommender systems*, pp. 1–8, 2014.
- [4] K. F. F. Jiechieu and N. Tsopze, “Skills prediction based on multi-label resume classification using cnn with model predictions explanation,” *Neural Computing and Applications*, pp. 1–19, 2020.
- [5] L. Sayfullina, E. Malmi, and J. Kannala, “Learning representations for soft skill matching,” in *International conference on analysis of images, social networks and texts*, pp. 141–152, Springer, 2018.
- [6] S. Fareri, N. Melluso, F. Chiarello, and G. Fantoni, “Skillner: Mining and mapping soft skills from any text,” *arXiv preprint arXiv:2101.11431*, 2021.
- [7] A. Gugnani and H. Misra, “Implicit skills extraction using document embedding and its use in job recommendation,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 13286–13293, 2020.
- [8] Q. Luo, M. Zhao, F. Javed, and F. Jacob, “Macau: Large-scale skill sense disambiguation in the online recruitment domain,” in *2015 IEEE International Conference on Big Data (Big Data)*, pp. 1324–1329, IEEE, 2015.

- [9] B. Gaur, G. S. Saluja, H. B. Sivakumar, and S. Singh, "Semi-supervised deep learning based named entity recognition model to parse education section of resumes," *Neural Computing and Applications*, pp. 1–14, 2020.
- [10] I. Kivimäki, A. Panchenko, A. Dessy, D. Verdegem, P. Francq, H. Bersini, and M. Saerens, "A graph-based approach to skill extraction from text," in *Proceedings of TextGraphs-8 graph-based methods for natural language processing*, pp. 79–87, 2013.
- [11] J. Li, A. Sun, J. Han, and C. Li, "A survey on deep learning for named entity recognition," *IEEE Transactions on Knowledge and Data Engineering*, 2020.
- [12] I. Khaouja, G. Mezzour, K. M. Carley, and I. Kassou, "Building a soft skill taxonomy from job openings," *Social Network Analysis and Mining*, vol. 9, no. 1, pp. 1–19, 2019.
- [13] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," 2013.
- [14] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with sub-word information," 2017.
- [15] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018.
- [16] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," 2018.
- [17] S. Vidros, C. Koliass, G. Kambourakis, and L. Akoglu, "Automatic detection of online recruitment frauds: Characteristics, methods, and a public dataset," *Future Internet*, vol. 9, no. 1, p. 6, 2017.
- [18] J. Balcar, "Is it better to invest in hard or soft skills?," *The Economic and Labour Relations Review*, vol. 27, no. 4, pp. 453–470, 2016.
- [19] E. A. A. Lumague, "Relative value of hard skills and soft skills for hiring employees in manufacturing sector," *Journal of Business & Management Studies*, vol. 3, no. 1, pp. 1–5, 2017.
-

- [20] J. Li, A. Sun, J. Han, and C. Li, "A survey on deep learning for named entity recognition," *IEEE Transactions on Knowledge and Data Engineering*, 2020.
- [21] SpaCy, "Model architectures," 2021. <https://spacy.io/api/architectures#parser>, Last accessed on 2021-08-07.
- [22] L. Kjeldgaard and L. Nielsen, "Nerda," GitHub, 2020.
- [23] F. Chollet, *Deep learning with Python*. Simon and Schuster, 2017.
- [24] P. K. Roy, S. S. Chowdhary, and R. Bhatia, "A machine learning approach for automation of resume recommendation system," *Procedia Computer Science*, vol. 167, pp. 2318–2327, 2020.
- [25] H. Suman, H. Tamkiya, and A. S. Kushwah, "Candidate background verification using machine learning and fuzzy matching," 2020.
- [26] K. M. Neckerman and J. Kirschenman, "Hiring strategies, racial bias, and inner-city workers," *Social problems*, vol. 38, no. 4, pp. 433–447, 1991.
-

A Statement of original thesis

Official statement of original thesis

By signing this statement, I hereby acknowledge the submitted thesis (hereafter mentioned as “product”), titled:

A Context Aware Approach for Extracting Hard and Soft Skills

to be produced independently by me, without external help.

Wherever I paraphrase or cite literally, a reference to the original source (journal, book, report, internet, etc.) is given.

By signing this statement, I explicitly declare that I am aware of the fraud sanctions as stated in the Education and Examination Regulations (EERs) of the SBE.

Place: Maastricht

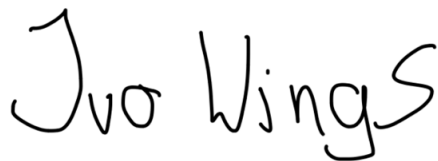
Date: 8-8-2021

First and last name: Ivo Wings

Study programme: Business Intelligence & Smart Services

Course/skill: Master Thesis

ID number: i6262955

A handwritten signature in black ink that reads "Ivo Wings". The letters are cursive and slightly slanted to the right.

Signature:
