Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2021) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

10th International Young Scientists Conference on Computational Science

A Context-Aware Approach for Extracting Hard and Soft Skills

Ivo Wings[a,*], Rohan Nanda[b,c], Kolawole John Adebayo[d]

[a]*School of Business and Economics, Maastricht University, Tongersestraat 53, 6211 LM Maastricht, The Netherlands*
[b]*Maastricht Law and Tech Lab, Maastricht University, Bouillonstraat 1-3, 6211 LH Maastricht, The Netherlands*
[c]*Institute of Data Science, Maastricht University, Paul-Henri Spaaklaan 1, 6229 EN Maastricht, The Netherlands*
[d]*Businesspoint Intelligence Solutions Ltd, 7A Butterfield Grove, Dublin and D14 Dublin, Ireland*

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 which can accurately and automatically extract hard and soft skills from candidates' resume 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 paper, we propose a context-aware sequence classification and token classification model for extracting both hard and soft skills. We utilized 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. Our results indicated that the best performing sequence classification model used BERT embeddings in addition with POS and DEP tags as input for a logistic regression classifier. The best performing token classification model used fine-tuned BERT embeddings with a support vector machine classifier.

Keywords: Natural language processing; Human Resources management; Skill extraction

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

* Corresponding author
E-mail address: i.wings@student.maastrichtuniversity.nl

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 up-skilling 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, if 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 or a sequence classifier which 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. State-of-the-art machine learning classifiers can learn to identify skills and automate the skill extraction process [1]. This paper aims to answer the following questions. First, 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 [1] [2] [3] [4]. 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. This brings us to the second question which asks what impact the context has in classifying terms as *soft/hard skill* or *not skill*? This paper proposes a context-aware sequence classification model for skills extraction implemented using linear machine learning algorithms and shallow classifiers. Furthermore, the paper attempts an extensive evaluation to compare our approach to previously used techniques. The significance of this research is that it presents an overview of the performance of difference word embedding algorithms, the creation of one robust model for extracting hard and soft skills, and finally the creation of a context aware system for skill extraction.

2. Methodology

This section is used to describe the methodology used to create the skill extraction pipelines.

2.1. Data collection

The first step in creating the skill extraction pipeline (hereafter pipeline) is data collection. We decided to use the publicly available Employment Scam Aegean Dataset (EMSCAD)[5]. The dataset contains *17,014* legitimate and 866 fraudulent job ads. We only utilized the legitimate jobs. The personal information in job ads was either removed or anonymized. After collecting the job descriptions, hard- and soft skill taxonomies were created. The first taxonomy is a soft skill taxonomy presented in [6]. 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 API[1]. A total of *28280* hard skills were added from EMSI skills API. In addition, a soft skill dataset was collected from [4]. This dataset contained annotated soft skills along with the context. This dataset contains *6721* soft skills.

2.2. Text Pre-processing and Dataset Preparation

This section describes the pre-processing step for the job descriptions, taxonomies and soft skills.

2.2.1. Job descriptions

The text cleaning process involves HTML tag removal and removing the pre-masked patterns of email addresses, phone numbers and URLs using custom RegEx. We also lower-cased the text and removed newline symbols and

[1] https://skills.emsidata.com/

whitespaces. The job descriptions are then sentence-tokenized using SpaCy and thereafter, any special characters were removed. These pre-processing steps yielded *302284* sentences.

2.2.2. Taxonomies

The taxonomy presented in [6] was read and transformed into a list of words. The taxonomy collected from EMSI [2] only contained hard skills which did not need any transformation in order to be used in the model.

2.2.3. 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 replaced with the soft skills, and after this step the text was lower-cased, and any special characters were removed.

2.2.4. Candidate skill and context selection with N-grams

After pre-processing, it is necessary to utilize n-grams to generate *candidate skills* from our job description collection. N-grams are sequences of words made from corpora and are used to turn sentences into smaller chunks better suited for analysis and annotation. In our case, they are also used to collect the context around a reference term in a sentence. 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 the average have up to 4 tokens. The list of n-grams will ordinarily include those which do not contain any skill term, therefore, such n-grams were removed. To achieve this, we used an integrated gazetteer which combines entries from the aforementioned taxonomies in addition to ESCO[3] and O*NET[4] skills taxonomies to filter out the noisy n-grams and retain only those contained in our integrated gazetteer. To ensure exact matches, we employed Flashtext as the n-grams filter . Subsequently, we perform de-duplication to remove redundant n-grams 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 our dataset will have the word 'Java' as the reference term. First, we check the gazetteer for the word 'Java' and if found, we extract the left context (be comfortable programming in) and right context (Python and other programming). 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 our 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 our job description collection which was found to be 15 tokens. Considering the average sentence length, we intuitively chose a window size = *5*.

2.2.5. Data Annotation

This study frames the skill extraction problem as a multi-class classification problem with three classes. We labelled the prepared dataset using the labels *Not Skill*; *Soft Skill*; and *Hard Skill*. Previous research [4] framed a soft skill extracting approach as a binary classification problem. The classes were soft skill or not skill. We extend these classes with the hard skill label to build one robust model. 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 [7] [8] 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. 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. Where the candidate-skill has fewer context words than the window size, we pad such context with an *Empty* tag.

[2] https://skills.emsidata.com/
[3] https://ec.europa.eu/esco/portal/home
[4] https://www.onetcenter.org/taxonomy.html

Gold-Standard annotation is usually manual and therefore labour-intensive. Given our limited resources, it would be difficult to annotate the whole samples in our dataset. To resolve this, we randomly selected *2000* samples which were then annotated by the authors and validated by an expert in the industry. To annotate, each candidate skill was only assigned an appropriate label with respect to its context. The Gold Standard dataset was then used to train selected machine learning classifiers to identify hard and soft skills. An example of the annotation is shown in Table 1. After the manual annotation process, the Gold-Standard dataset was supplemented with a list of annotated soft skills [4] (6721) resulting in a total corpus size of *8721* samples.

Table 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

2.3. Context-Aware Sequence Classification Model

In this section, we present the context-aware sequence classification model for identifying hard and soft skills from job descriptions.

2.3.1. Word embeddings

Previous studies on skill extraction have utilized bag-of-words and lexical approaches to model features for machine learning classifiers. We 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[9], FastText [10], and BERT [11] were implemented for performance evaluation. The main difference between these word embeddings is their training objectives and context-awareness. Pre-trained word embeddings were used because the number of job descriptions in the EMSCAD dataset were 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 was 300 dimensions. The English language model was used as pre-trained FastText model. This model was trained on Common Crawl and Wikipedia. The final dimensions of the word embeddings were *300* dimensions as well.

Contextual word embeddings: BERT has been used as a contextual word embedding method. Assuming the same input sentence presented in the previous paragraph. The entire sentence was loaded into BERT and afterwards the candidate skill embeddings were extracted from the model. Because BERT is a contextual word embedding method, each token gets a different embedding depending on the sentence. The 'bert-base-uncased' pre-trained BERT model 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.

2.3.2. 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 embedding of the tokens in the example sentence, the embedding for the context and candidate skills have been separated using a separator (222). In total each data sample had 902 dimensions. 300 for each of the context and candidate skills embeddings, separated by 2 separators.

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 the candidate skills could be up to 4 tokens. Therefore the embedding would have different dimensions which would result into issues further on. To create a uniform dimension, the tokens were zero-padded up to the maximum fixed length (4 tokens) and thus yielding a feature of 3075 dimensions, i.e., *768 * 4 + 3 separators*.

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 sentence 'Work closely with business clients' would be VERB, ADV, ADP, NOUN, and NOUN. After generating the tags, the tags were counted and added as features. An example of the dummy variable is given in Table 2. The term 'other tags' stands for other POS and DEP tags.

Table 2. POS and DEP example

VERB	ADV	ADP	NOUN	Other tags
1	1	1	2	0

2.3.3. Classifiers

After creating all the features which represent the n-grams, the classifiers were trained on the annotated data to make the predictions. Multiple classifiers have been used to compare performance. These include: Baseline, Logistic regression (LR), Gradient boosting classifier (GBC), Random Forest (RF), Support vector machine (SVM) and Multilayer Perceptron (MLP). 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 ration. This was important as there were imbalanced classes in the dataset.

2.3.4. 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 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 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 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 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.

2.4. Token classification

An alternative approach for the sequence classification model presented in 2.3 is to classify each individual token in the three proposed classes. This approach is described in this section.

2.4.1. Fine Tuning BERT

BERT is a neural network and can therefore be fine tuned towards solving specific problems. By fine tuning BERT we can increase the performance, because the model learns the specific task it has to perform. The 'bert-base-uncased' model was initially used. The dataset presented in Table 1 was transformed into the dataset depicted in Table 3 and used as input for the fine tuning process. This transformation was performed in order to label each token in the gold standard dataset individually. BERT was fine tuned using Huggingface and Tensorflow. The model is publically available and can be found at [5]

[5] https://huggingface.co/Ivo/emscad-skill-extraction-conference-token-classification

2.4.2. *Token classification model*

An alternative to the sequence classification model presented in 2.3 is to classify each individual token into not skill, soft skill, or hard skill. The dataset presented in Table 3 was used as input for the token classification model. BERT embeddings for each of the tokens were extracted. This approach has only been used for BERT because this word embedding technique is context aware, resulting in different embeddings for each of the tokens. These word embeddings and POS and DEP tags were then used as input for the classifiers. The parameters for the approach can be found in section 2.3.4. The results of the token classification model are presented in 3.

Table 3. Token classification dataset example

Token	Label
senior	not skill
Python	hard skill
programming	not skill

3. Results and analysis

This section is used to describe the results of the two pipelines: the context aware sequence classification model and the token classification model.

After the creation of the sequence classification model, multiple evaluation metrics have been used to evaluate the performance. The evaluation corpus metrics were calculated on a corpus size of *8721* labeled samples.The precision is presented in Figure 2, the recall is presented in Figure 1, and finally the F1-score is presented in Figure 3. The Figures present the average of the macro evaluation metrics after 10 fold cross validation. The highest precision is 0.83 achieved with BERT word embeddings, the use of POS & DEP tags, predicted by the logistic regression classifier. The highest recall is 0.84 achieved with BERT word embeddings, the use of POS & DEP tags, predicted by the logistic regression classifier. The highest F1-score achieved (0.83) is with BERT word embeddings, the use of POS & DEP tags, predicted by the logistic regression classifier. We also present the confusion matrices by fitting the classifier on 80% of the data while making the predictions on 20% of the data. These steps have been completed for the best performing classifier on the manually annotated dataset. The confusion matrix for the best performing classifier can be found in Figure 5. Next to the best performing confusion matrix, a baseline confusion matrix is plotted. This matrix is plotted to compare the pipeline against a baseline. This confusion matrix can be found in Figure 4. Finally, some example predictions are shown of the pipeline in Figure 8.

Fig. 1. Recall sequence/token classification model

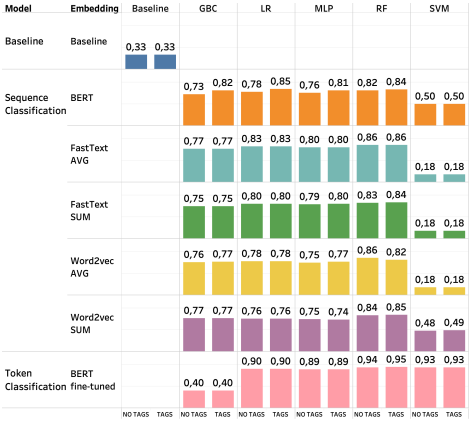

Fig. 2. Precision sequence/token classification model

The results indicate that the logistic regression classifier with BERT embeddings and POS & DEP tags features achieved the best performance. This was also largely expected due to the fact that BERT is a context-aware method

Fig. 3. F1-score sequence/token classification model

of assigning word embeddings to words whereas Word2Vec and FastText are not context aware. However, the F1-scores of FastText and Word2Vec are not that far off from BERT. This result was surprising because the method of capturing context with these two methods seems to provide positive results. A benefit of using Word2Vec or FastText is a reduced number of dimensions leading to a decreased fitting time of the classifiers. All of the classifiers achieved a higher performance than the baseline classifier.

The confusion matrix show 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. In the confusion matrix we can clearly see that there is a low number of annotated hard skills in the test set. A reason for the class imbalance is the use of the soft skills presented in [4]. This dataset contained 6721 annotated samples. This in comparison the the manually annotated samples(2000) would make the pipeline perform better in predicting soft skills than hard skills. Overall comparing the confusion matrix of the baseline in Figure 4 with the confusion matrix of the pipeline in Figure 5 we can see that the performance of the pipeline outperforms the baseline tremendously.

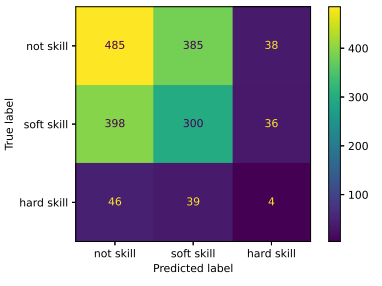

Fig. 4. Confusion matrix baseline sequence classification test data

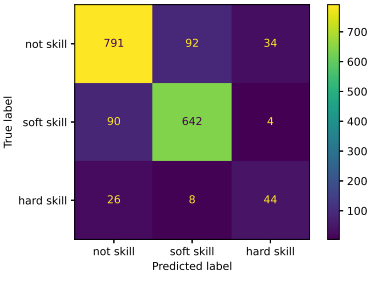

Fig. 5. Confusion matrix BERT POS DEP LR sequence classification test data

Figure 8 presents some example predictions made by the classifier on the test data. The first three examples in this Table are correctly classified. However, the last four examples are incorrectly classified as either a soft skill, hard skill, or not a skill. A possible reason for this is insufficient annotation.

We used the same performance metrics and approach for the token classification model. The fine-tuned BERT word embeddings as well as the POS and DEP tags were used as input for the classifiers. The precision scores are visualized in Figure 2, the recall scores are visualized in Figure 1, and the F1-scores are visualized in Figure 3. The highest precision achieved was *0.94* by the Random forest with POS and DEP tags. The highest recall achieved was *0.90* by the Multi-layer perceptron without POS and DEP tags. The highest F1-score achieved was *0.91* by the Support vector machine without POS and DEP tags. In Precision, Recall, and F1 score tables we can see that the addition of POS and DEP tags does not lead to a significant performance increase.

We also present the baseline confusion matrix in Figure 6 and the confusion matrix for the BERT token classification model with SVM in Figure 7. If we compare the baseline and the model we can see that the token classification model outperforms the baseline significantly. In the confusion matrix we can see that most of the tokens were labeled as soft skill. These were tokens that were in the context columns of the dataset. The model classified most of the soft skills accordingly and most of the hard skill accordingly as well. However, the same issues as the sequence classification dataset arise with a low number of annotated hard skills in the test dataset.

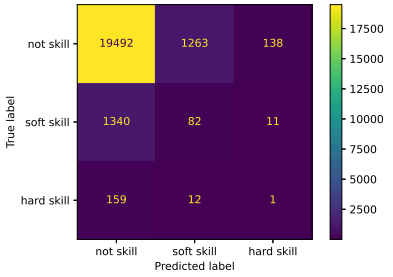

Fig. 6. Confusion matrix baseline token classification test data

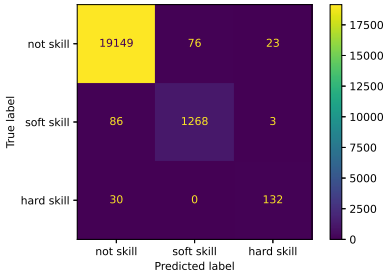

Fig. 7. Confusion matrix BERT SVM token classification test data

left_context	candidate_skill	right_context	label	predictions
the	person we are	looking to hire immediately	not skill	not skill
commercially aware logical and strategic thinker	selfconfident	and able to operate without	soft skill	soft skill
empty	voicexml	empty	hard skill	hard skill
introduce new product such a cobranded	loyalty	card and transactional business service	not skill	soft skill
programming experience solid	understanding of data	structures algorithms	not skill	hard skill
create clear and compel report base on	rigorous	analysis and an open mind	soft skill	not skill
empty	livelink	empty	hard skill	not skill

Fig. 8. Sequence classification model predictions

token	label	predictions
building	not skill	not skill
problemsolving	soft skill	soft skill
java	hard skill	hard skill
customisations	not skill	soft skill
organisational	not skill	hard skill
motivated	soft skill	not skill
dynamic	soft skill	hard skill
mongodb	hard skill	not skill
jquery	hard skill	soft skill

Fig. 9. Token classification model predictions

Figure 9 shows a sample of predictions made by the token classification model. The first 3 examples are correctly classified. However, the samples are not. This could be due to insufficient annotated samples.

If we compare the F1-scores of the Sequence classification model and token classification model we can see that the token classification outperforms the sequence classification model significantly. The sequence classification model achieved a maximum F1-score of *0.83* whereas the token classification model achieved *0.91*. However, the scores of the token classification model could be biased. For example, the soft skill: *ability to learn*. The sequence classification model would classify this as one soft skill, whereas the token classification model would classify this as three separate soft skills. This could bias the evaluation metrics.

We proposed a sequence classification model and a token classification model. A limitation of the sequence classification model is the need for determining the left and right context columns. We therefore also proposed the token classification model which is easier to implement in a production environment.

4. Related Work

This section is used to describe the main findings of the literature review for the paper. The presented literature can be classified into two different literature streams: skill extraction, and taxonomy creation. The work in [1] described a method for skill identification and normalization in resumes using NER and Named-entity normalization (NEN). A corpus of around 100 million resumes was utilized. Wikipedia was used to create a taxonomy of skills. A Word2vec based model was used to compute a relevance score between the candidate term and the elements in taxonomy to decide if the term would be labelled as a skill. The system solely relied on the relevance score and did not take the context or other semantic features into account.

The work in [12] described a system for skill sense disambiguation for resumes and job postings. Each document in the dataset contained the extracted skills. All documents are clustered based on their base skill form. By doing

this, different clusters are generated which are similar to another. After generating the clusters, the context of the skill is encoded using an n-gram model with Word2Vec. These vectors are then clustered using the Markov Chain Monte Carlo algorithm to determine the final sense. By using state of the art word embeddings, the clustering could be improved.

Another method for extracting skills was employed by using LinkedIn [2]. A folksonomy was created for the members to select their skills. The authors extracted the list of skills user use to summarize their skills. After skill extraction, a clustering algorithm was used to detect ambiguity between terms. A list of related phrased was generated for each skill phrase used for disambiguation. A clustering algorithm is used on the phrases, and the most dominant cluster is selected as the final sense. These are then evaluated using LinkedIn users. The results of the paper indicated that the system is able to predict in multiple languages.

In [3], authors presented a system that predicted skills using a Convolutional neural network(CNN) model. The problem was modeled as a multi-label classification problem for resumes. All the classes were normalized using a dictionary-based normalizer algorithm. Next, word embedding models were trained on the resumes. Finally, the labeled dataset is divided into class-specific sets on which independent CNN classifiers are trained to make the final predictions. The system performed the best for IT related resumes. It would be interesting to see more applications in different sectors.

Gaur et al.[13] presented a semi-supervised approach for extracting the education qualifications of candidates from a dataset of resumes. A small proportion of training data was manually annotated. The classification model consisted of a word embedding layer, CNN layer, and Bi-LSTM layer. The predictions made by the model were improved by using a correction module. The results could be improved by using manually annotated data as the input.

Sayfullina et al. [4] described a system used for detecting soft skills in unstructured text corpora. They formulated skill extraction as a binary classification problem. The dataset consisted of a corpus of job advertisements, resumes, and a publicly available soft skills list. Job advertisements were checked for exact matches with the soft skill list. The matches were highlighted in the sentences after which these sentences were shown to people and manually annotated. The annotated data was used as input for the LSTM, CNN, and HAN neural networks. The LSTM neural network had the best performance.

SkillNER is a named entity recognition (NER) system which uses a support vector machine model for extracting soft skills [14]. The system was trained on a corpus of 5000 scientific papers and used a soft skill taxonomy from O*NET. The system consists of two stages: clue extraction and skill extraction. The clue extraction stage consists of extracting patterns which would hint towards a soft skill. These extracted clues were used match with sentences throughout the corpus. These sentences were finally annotated in the skill extraction stage and used to train a support vector machine and a multi-layer perceptron model. The evaluation metrics show that the system could be improved.

Kivimäki et al. [15] presented a graph-based approach for skill extraction. They obtained a skill taxonomy from LinkedIn and used the hyperlink structure from Wikipedia. They created a module which associated Wikipedia pages to a given input text by using text similarity techniques such as TF-IDF and LDA. They further 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. These techniques could be improved by using state-of-the-art techniques. The TD-IDF approach had the best performance given this experimental setting.

Gugnani el al. [16] present a job recommender system which matches candidate resumes to job descriptions. The authors collected 1.1 million job descriptions as well as 1.314 resumes. The noun phrases from the input text were collected using NER. A rule based system was developed for detecting skills based on the semantic tree. A dictionary was used for additional skill matching. Word2vec was used to compare the vectors from the collected skills and the skills in dictionary. The results indicate that the system is able to predict a job description to an applicant with just 5 recommendations.

Previous work focused on extracting skills from corpora. Khaouja et al. [6] created a soft skill taxonomy using job openings which can be used for skill extraction. The dataset consisted of job descriptions. The authors established a first list of soft skills using the most common bi and tri-gram phrases that occurred at least ten times in the corpora as well as a list from previous works. After this step, the n-grams were converted into word vectors using Word2Vec. Additionally, n-grams were searched on DBpedia, where related words were extracted and stemmed. The cosine similarity between the n-grams and DBpedia terms is calculated and used to determine the hierarchy. An improvement for this paper could be the use of context aware word embeddings.

5. Conclusion and future work

This paper presented a context-aware sequence classification model as well as a token classification model for hard and soft skill extraction. We utilized N-grams to capture the context of the skills in job descriptions. The textual features were modelled using the pre-trained word embedding techniques and features from part-of-speech and dependency parsing. We utilized state-of-the-art machine learning classifiers to identify both hard and soft skills. The performance of the models were evaluated on the publicly available EMSCAD dataset of job descriptions. The best performing sequence classification model used BERT embeddings in addition with POS and DEP tags as input for a logistic regression classifier. The best performing token classification model used fine-tuned BERT embeddings with a support vector machine classifier.

Future research work may benefit by implementing a semi-supervised classifier to speed up the task of manual annotation. It would be interesting to see if this would further increase the performance. Another variation in feature modellings would be the use of different sentence forms (lemmatized, no stop words) and comparing their performance with the original sentence form. The number of word embeddings for comparison could be extended with ELMO[17] embeddings as these are also context dependent word embeddings. Finally, creating more granular classes instead of the three proposed classes might add even more value for the HR sector. An example of this would be language skills, programming skills, construction skills.

References

[1] Meng Zhao, Faizan Javed, Ferosh Jacob, and Matt McNair. Skill: A system for skill identification and normalization. In *Proceedings of the twenty-ninth AAAI conference on artificial intelligence*, pages 4012–4017, 2015.

[2] Mathieu Bastian, Matthew Hayes, William Vaughan, Sam Shah, Peter Skomoroch, Hyungjin Kim, Sal Uryasev, and Christopher Lloyd. Linkedin skills: large-scale topic extraction and inference. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 1–8, 2014.

[3] Kameni Florentin Flambeau Jiechieu and Norbert Tsopze. Skills prediction based on multi-label resume classification using cnn with model predictions explanation. *Neural Computing and Applications*, pages 1–19, 2020.

[4] Luiza Sayfullina, Eric Malmi, and Juho Kannala. Learning representations for soft skill matching. In *International conference on analysis of images, social networks and texts*, pages 141–152. Springer, 2018.

[5] Sokratis Vidros, Constantinos Kolias, Georgios Kambourakis, and Leman Akoglu. Automatic detection of online recruitment frauds: Characteristics, methods, and a public dataset. *Future Internet*, 9(1):6, 2017.

[6] Imane Khaouja, Ghita Mezzour, Kathleen M Carley, and Ismail Kassou. Building a soft skill taxonomy from job openings. *Social Network Analysis and Mining*, 9(1):1–19, 2019.

[7] Jiří Balcar. Is it better to invest in hard or soft skills? *The Economic and Labour Relations Review*, 27(4):453–470, 2016.

[8] Erus Angelo A Lumague. Relative value of hard skills and soft skills for hiring employees in manufacturing sector. *Journal of Business & Management Studies*, 3(1):1–5, 2017.

[9] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality, 2013.

[10] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information, 2017.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[12] Qinlong Luo, Meng Zhao, Faizan Javed, and Ferosh Jacob. Macau: Large-scale skill sense disambiguation in the online recruitment domain. In *2015 IEEE International Conference on Big Data (Big Data)*, pages 1324–1329. IEEE, 2015.

[13] Bodhvi Gaur, Gurpreet Singh Saluja, Hamsa Bharathi Sivakumar, and Sanjay Singh. Semi-supervised deep learning based named entity recognition model to parse education section of resumes. *Neural Computing and Applications*, pages 1–14, 2020.

[14] Silvia Fareri, Nicola Melluso, Filippo Chiarello, and Gualtiero Fantoni. Skillner: Mining and mapping soft skills from any text. *arXiv preprint arXiv:2101.11431*, 2021.

[15] Ilkka Kivimäki, Alexander Panchenko, Adrien Dessy, Dries Verdegem, Pascal Francq, Hugues Bersini, and Marco Saerens. A graph-based approach to skill extraction from text. In *Proceedings of TextGraphs-8 graph-based methods for natural language processing*, pages 79–87, 2013.

[16] Akshay Gugnani and Hemant Misra. Implicit skills extraction using document embedding and its use in job recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13286–13293, 2020.

[17] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations, 2018.