Expertise
and insight
for the future

Mariia Chernova

Occupational skills extraction with FinBERT

Metropolia University of Applied Sciences

Master of Engineering

Information Technology

Master's Thesis

25 November 2020

Author Title Number of Pages Date	Mariia Chernova Occupational skills extraction with FinBERT 49 pages 25 November 2020
Degree	Master of Engineering
Degree Programme	Information Technology
Instructor(s)	Ville Jääskeläinen, Principal Lecturer

Job search market is highly competitive even in a small country such as Finland. Oikotie Työpaikat is a platform, where recruiters post jobs, candidates search and apply for open positions. In order to stay among the leaders in this race, Oikotie Työpaikat desires to put personalization features on the next level. Therefore, it was important for the platform to obtain a tool that allows to extract skills from job postings. In order to build in the future applications for better user experience.

The main objective of this master thesis was to develop framework that extracts skills from unstructured text such as job description. In the initial phase, the study explored the variety of currently used skills extraction systems and compared the possible options for implementing the framework. The next was to investigate different NLP techniques that take into account the context of words. These techniques include Self-Attention mechanism, RNN, LSTM, Transformer and BERT algorithms. Since the extraction system must be able to process words in Finnish, it was decided to leverage Google's open-source BERT model for the Finnish language (FinBERT) developed by Turku University. This version of BERT outperformed a previous multilingual model in a wide range of tasks, especially in classification problems. One of this task is NER, which can be easily applied to extract entities such as skills from unstructured texts, and it is utilized in this study.

The implementation process started with data cleaning and pre-processing an input for BERT model. The dataset provided by Oikotie Työpaikat contained about 300 000 job advertisements. 100 JDs were randomly selected from the pre-processed data. This dataset was labeled utilizing the web-based tool for NLP text annotation called TagTog. The architecture of the developed model contains the main block based on FinBERT and the additional layer was chosen as a simple Dense layer with a softmax activation function. This Dense layer was fine-tuned for the NER task. The developed model was trained and validated. The model performance was evaluating using confusion matrix and its based different evaluation metrics such as accuracy, precision, recall, and F1-score.

The developed skill extraction framework achieves noticeable results. Extracted skill phrases included soft skills, hard skill and different qualification certificates. Moreover, the developed framework has a potential to become a basis for various user and business applications, examples of which are also presented in this master thesis.

Keywords	Skills Extraction, NLP, NER, BERT, FinBERT

Contents

List of Figures

List of Tables

List of Abbreviations

ANN	Artificial Neural Network.
API	Application Programming Interface.
BERT	Bidirectional Encoder representations from Transformers.
CNN	Convolutional Neural Network.
CV	Curriculum Vitae.
FinBERT	A version of open-source BERT model for the Finnish language developed by Turku University.
JD	Job Description.
LSTM	Long short-term memory.
ML	Machine Learning.
MLM	Masked Language Modeling.
NER	Named Entity Recognition.
NLP	Natural Language Processing.
NSP	Next Sentence Prediction.
PoS	Part of Speech.
RNN	Recurrent Neural Network.
seq2seq	Sequence-to-sequence model.

1 Introduction

Nowadays job search services contain a huge number of open positions and not fewer job seekers. However, the unemployment rate is not getting lower. Recent research from McKinsey [1] demonstrated that European employers are facing a growing crisis of not finding people with indispensable skills to fill even entry-level positions. At the same time, the European Union has 5.6 million young people without jobs. One of the main reasons for this is a mismatch between the skills required in the Job Description (JD) and the skills listed in the CV. A general step in reducing the gap is accurate skills extraction and their comparison in the CV and job advertisement.

Job skills extraction is a big challenge and often solved by traditional techniques such as matching against a taxonomy skills dictionary [2]. However, this approach does not extend to new and emerging skills. The dictionary updating can be manual and tedious, and also requires plenty of time of domain experts to identify correct skills that map to a particular field. In addition to that skill extraction needs serious consideration, for instance, the term «Java» can be an island in Indonesia or an object-oriented programming language. Moreover, CVs and JDs usually contain unstructured text that also may include tables, bulleted lists, etc.

Although general-purpose search engines have made an immense progress, job search services have experienced rather modest development. Considering all of these, and also other factors, automated job search engines require research investment to make them reliable and improve their performance. Therefore, to analyze and reduce the skills mismatch for job search service it is crucial to have an automated framework that can extract skills from JDs and CVs. The resulting data can be leveraged as the foundation for labor market analysis and can also be used in job matching and recommendation systems to better match candidates to jobs and reduce unemployment.

1.1 Research Goals and Attendant Company

The main objective of this research is to propose an algorithm to extract skills from unstructured texts in order to overcome some of the above-mentioned challenges. The fundamental premise this research builds upon is that skills are one of the most important

aspects while matching CVs to JDs, and play a major role in recommending JDs which are the best match for a certain CV.

This research aims to archive the following goals:

- identify trends in the industry based on current state analysis,
- study several natural language processing (NLP) techniques such as Transformer, Bidirectional Encoder Representations from Transformers (BERT), and etc.,
- develop an algorithm for robust skills extraction from unstructured texts,
- pre-process data and implement the idea.

The topic of this thesis attracted attention of Oikotie Työpaikat. It is a job search service, that is one of the key players on the market in Finland. On this platform, recruiters post jobs, candidates search and apply for open positions. In addition, the service also aggregates jobs from external sources. There are about 5 thousand active jobs at a time on the platform. The service is quite popular and it is visited over 2 million times in a month. However, Oikotie Työpaikat has at least 10 competitors in Finland. That is why it is constantly looking for ways to stand out in the market.

1.2 Method and Material

A qualitative method was used to gather the information to define the scope of the research question. Requirements for the solution came from observations from the current job search process and system, interviews and feedback from potential customers and business owners of Oikotie Työpaikat.

The solution was created and evaluated using the quantitative method. The model performance was evaluated by traditional metrics such as confusion metrics, precision, etc. For this research Oikotie Työpaikat provided two datasets, that included CVs and JDs. The obtained dataset was preprocessed and labeled for further model training.

1.3 Thesis Structure

The rest of the master thesis is organized as follows: In Section 2, related work from the area of skills extraction and its analysis are briefly described. The theoretical background used in this work and the proposed approach are explained in Sections 3 and 4,

respectively. The result of the proposed method is presented in Section 5 followed by conclusions.

2 Current State Analysis

This Section presents the core techniques used nowadays to extract entities such as skills from JDs and CVs. It explores various industry-standard practices and strategies in fetching entities, including previous common research that focuses on the taxonomy dictionary. This section also detects the key weaknesses regarding the existing solutions. Moreover, to come up with answers to the research questions, information is analyzed and compared from publications, journals, websites, and books. The knowledge presented here lays the ground for understanding, scoping, and designing the new model of extracting skills that this thesis aims to implement. NLP techniques mentioned in this section and essential for the current research are described are rigorously explained in Section 3.

2.1 Skills Folksonomy/Taxonomy and Graph-Based Approaches

One of the common approaches for skills extraction offers methods that rely on the construction of folksonomy or taxonomy of entities [5]. Both terms are types of controlled dictionaries, but they contain several key differences presented in Table 1.

Table 1. The comparison of taxonomy and folksonomy.

Taxonomy	**Folksonomy**
Hierarchical - Parent/child & sibling relationship	Flat - No levels, no order, no explicit relationship
Exclusive - The same item cannot be in few distinct categories	Not Exclusive - An item can be associated with many tags
Top-down - Established by experts	Bottom-up - Created by users

One of the extraction systems based on skills taxonomy is ESCO [4]. It is the multilingual classification of European Skills, Competences, Qualifications and Occupations. It works as a dictionary and aspires to provide semantic interoperability between labor markets, education and training programs. Unfortunately, there is no available information on what techniques and methodologies were used to build the ESCO taxonomies. However, this dictionary is public and can be useful for building taxonomy in this research.

Nowadays the world leader among job search systems is LinkedIn. In 2014, the team at LinkedIn built a massive skills extraction framework. The framework was developed on top of a constructed folksonomy of skills and expertise and provided an inference for a recommender system [2]. The folksonomy-building system consisted of discovery, disambiguation, and deduplication steps. The first discovery step was based on the observation that most of the users create a list of comma-separated skills in the specialty section of their profile. This feature was used for fetching entities as potential skills. The second step of disambiguation aimed to remove uncertainty in skill phrases that had multiple meanings depending on their context. The clustering based on the co-occurring phrases was used to solve the disambiguating skills issue. Some skills were tagged with many senses belonging to different clusters with an industry label. The last step was to eliminate semantic duplicates such as “Python development” and “Python programming”. To solve this problem, researchers applied crowdsourcing approach that involved LinkedIn users. The users were asked to tag a skill phrase with the most relevant Wikipedia page from a suggested list. This skills extraction system also contained a skills inference component, which leveraged profile attributes such as company, title, and industry as features. The Naıve Bayes classifier was trained on the constructed feature set. However, the following researches demonstrated the superiority of graph-based model over Naïve Bayes classifier.

The approach for automated skills extraction from free written text documents suggested in Kivimäki et al. [3]. The main idea of this system was based on the hyperlink graph of Wikipedia and skills folksonomy obtained from LinkedIn. At first, the system computed similarities between an input document and the texts in Wikipedia pages and then applied the Spreading Activation algorithm on the Wikipedia graph to associate the input document with skills. This system was able to extract both inferred and explicitly stated skills from the text.

An approach similar to Kivimäki et al. was presented by Wang et al. [6]. The system also applied skills folksonomy from LinkedIn and graph-based model using textual data resided in the skills and expertise sections, personal profile connections (shared majors, titles, companies, and universities), and skills connections (skills that co-occur together). While the approach constructed on skills connections outperformed the one that used only profile connections, the mutual system that utilized both connection types demonstrated the best results [6].

In a recent research Phuong et al. [7] revealed a new approach to skill taxonomy generation. As in LinkedIn [2], this work included the discovery, disambiguation, and deduplication steps, but with different approaches to implementation. In order to build taxonomy system, they collected skill-related content from candidate resumes and job descriptions available at the Career Builder web-site. The collected text data was split by punctuation marks and cleaned from noise, such as stop words, additional adverbs and other predefined terms by domain expertise. Essentially, the cleaning phase excluded words that bring no or very little semantic value to a constructed skill taxonomy. For normalization and deduplication researchers applied Wikipedia API [12]. Another important step in building a taxonomy of skills was validation, which used the Standard Occupational Classification (SOC) system to ratify the returned Wikipedia category tags. In this research, the issue of the word sense disambiguation (WSD) was addressed using the Google Search API. For example, if the skill term had multiple senses, the system chose the one with the highest Google Search relevancy ranking. Nevertheless, this approach demonstrated the explicit weakness in not considering semantic context. As a result, the researchers developed the Skill Tagging system, which is presented in more details in Figure 1. This skill taxonomy included 39,000 raw terms mapped to 26,000 normalized skill entities.

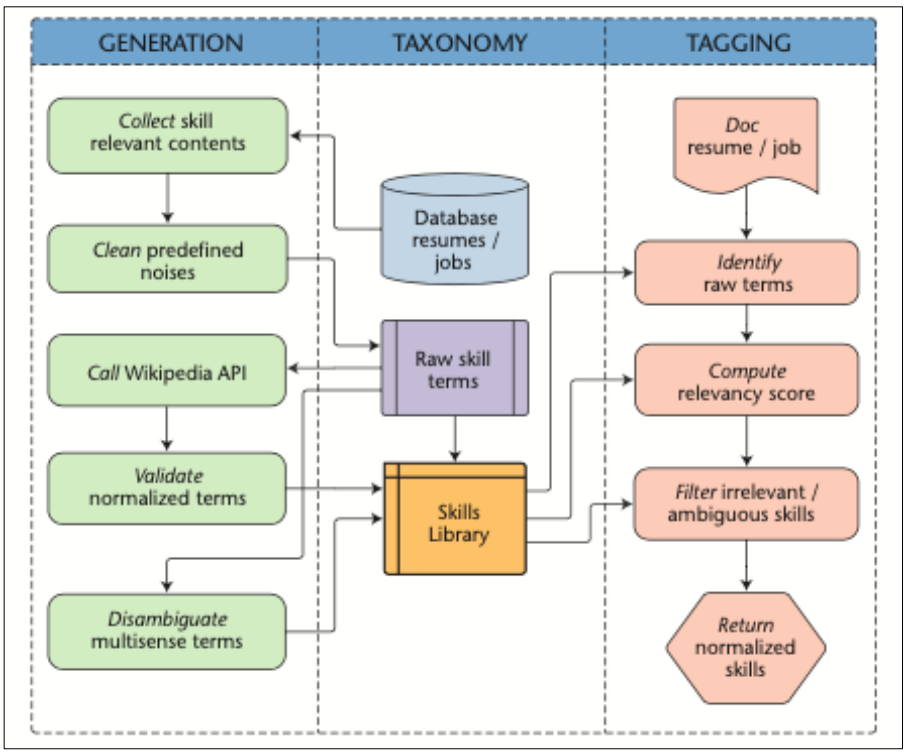

Figure 1 Architecture of the SKILL System [7].

2.2 Machine Learning Approaches

This section presents the most advanced approaches for extracting skills using Machine Learning (ML) techniques, such as natural language processing (NLP), deep learning networks etc.

In 2018 Nikita Sharma compared different approaches [8] based on unsupervised and self-supervised learning techniques to extract the relevant skills from free written text. The models were trained on small dataset of job advertisements in Data science category and then extended to other cross categories. The first two models were based on the following techniques:

- Topic Modelling is unsupervised technique to fetch abstract topics. This approach showed solid understanding of the context. However, the extracted key words mismatched with the relevant set of skills identified in problem statement.
- Word2Vec is a self-supervised neural network that is able to identify words used in similar contexts. This approach was applied to extend on the top of Topic modeling. Extracted key words by top modeling method were used for training Word2Vec model. Word2Vec demonstrated good results at recognizing skills. However, Word2Vec extracted a lot of noise and separating valuable skills from noise was a quit tedious work.

The other two models described in the research apply supervised learning. The training dataset was labeled manually and basically included only noun phrases as skills. Bellow a comparison in more details of those two approaches:

- The first was a simple word embedding based classifier, which contained a convolutional layer and was trained on the labeled dataset. This approach extracted a lot of useful skills from the job descriptions. Test accuracy was 0.6803.
- The second was the combination of word embedding and Long short-term memory (LSTM) that improved the accuracy of the skills classifier and also extracted a lot more keywords. This approach showed the best results with test accuracy 0.7658. The presence of noise was also reduced as compared to other models.

The LSTM and Word embedding model was able to provide decent results by training on a very small dataset. However, only noun phrases were used for model training, but a lot of job postings explain required skills in the form of verb phrases and other grammatical structures. Therefore, the initial dataset needs to be extended with the richer set of labeled examples, and only then a new model trained.

In 2020 Akshay Gugnani and Hemant Misra [9] presented a skills extractions framework, that was based on several natural language processing (NLP) techniques. The skills extraction framework was composed of four main submodules:

- Named Entity Recognition (NER) is usually used to identify keywords and concepts, extract entities, such as names of persons, organizations, locations, expressions of times, quantities, monetary values, percentages, etc. Researchers leveraged NER to extract skills from JDs and had noticed great results. They applied Watson NLU 3 services [13] to extract skills as entities. After that the extracted list of skills was classified as a set of “Probable Skills”. On the step the processed skills got assigned relevance scores by means of Word2Vec.
- Part of Speech (PoS) Tagger is the process of labeling of each word in a text as a corresponding part of speech. Five domain experts manually processed few hundreds of JDs and labeled words or phrases as skills. They noticed that skills vary depending on not only a job industry, but a subjective opinion of a person labeling it. The set of JDs data was also processed through the Stanford Core NLP Parser and PoS Tagging to identify part of speech. Based on the observations they defined rules and patterns for identifying potential or new skill-terms, that were not presented in skill dictionary or taxonomy. For instance, if there was a comma separated list of nouns in a sentence, and few nouns are skills, then the other nouns should probably be skills. Similar rules were programmed in the system to identify skill-terms.
- Word2Vec (W2V) [10] in a nutshell is used to represent word as a vector. The input data is usually a large corpus of text, which is used by W2V to create a vector space, typically, of several hundred dimensions. Each unique word is mapped to a corresponding vector in the vector space. Typically words from common contexts are located close to each other in the vector space. The W2V model tokenizes text using white spaces, therefore a single-word skill is easy to extract in most cases. However, the difficulty arises when the skill is a phrase such as “Hard Working”, “Web Development”, etc. To solve this issue,

researchers suggested to represent a skill phrase by a vector, which is an average of individual vectors composing the skill phase. Moreover, there was a skill dictionary, that was used to compared every potential skill-term in the embedded W2V space. The researchers also leveraged users' feedback mechanism to learn new skills and improve performance. The presented model was trained on the text corpus of 1.1 Million JDs from over 50 different categories.

- Skill Dictionary was needed to identify a word or phrase as a skill. In order to create this skill dictionary researches applied the same approach as proposed by Gugnani et al [11]. Skills were mined from public resources, such as Onet, Hope, and Wikipedia. Then the team of three experts validated these terms. The created skill dictionary contained 53,293 soft and hard-skills in different categories.

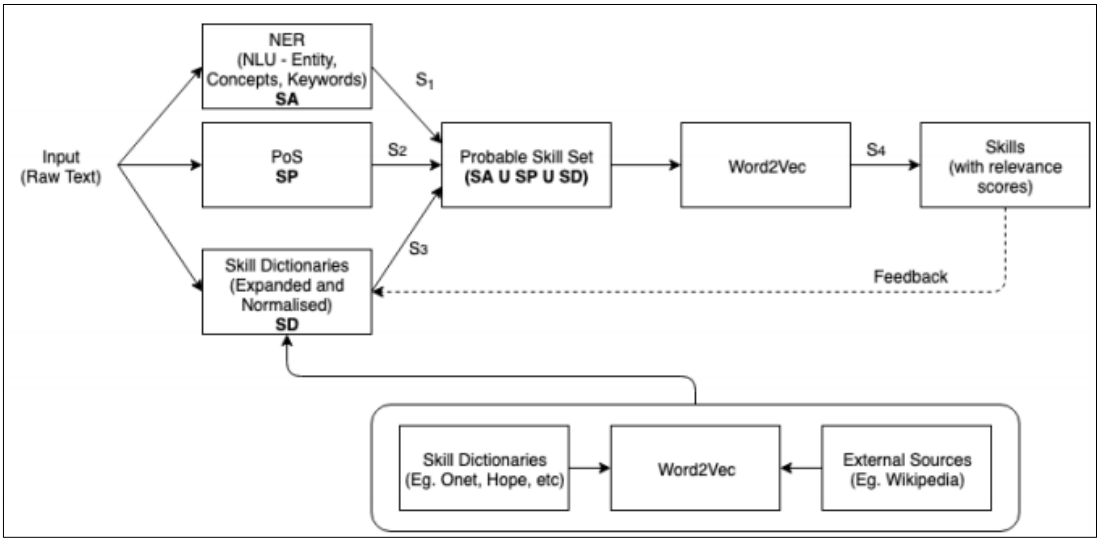

Figure 2. Skill Identification Flow Diagram [9].

Described above modules of skill extraction system is shown in Figure 2. The input data is a raw text, from which the first three modules extract a set of phrases and assign them a module-specific "score". Further, the combination of scores specifies how likely an identified phrase is a skill.

2.3 Limitations

There are many approaches to extract skills from a free written text and but this thesis concentrates only the most advanced and successful ones above. This thesis continues the development of this topic towards the defined focus taking into account the requirements and limitations of Oikotie Työpaikat.

First of all, Oikotie Työpaikat is a platform where most of the CVs and JDs are posted in Finnish language. As a result, the final solution should be able to work with texts in Finnish language.

Secondly, building skills folksonomy/taxonomy requires the involvement of a team of experts and users to solve deduplication and disambiguation issues. Unfortunately, in this thesis it was not possible to involve additional people to this work.

In addition, the considered ML approaches are based on supervised learning and the dataset for the training model must be labeled which requires time and extra resources. Therefore, the developed model should be trained on a relatively small dataset and show satisfactory results.

On top everything else the available computational resources are of high importance as well. In fact, the analysis of huge textual data and advanced ML methods such as BERT often demand powerful GPUs or advanced multicore CPU machines. As a result, the performance of one laptop may not be enough to train and test the developed skills extraction model.

2.4 Summary

Based on the described limitations, this section includes an analysis of different approaches for skills extraction and make conclusions whether or not they can be used for this work and why. The results are presented in a Table 2. Analysis of different approaches to skill extraction.

Table 2. Analysis of different approaches to skill extraction.

Approach name	**Conclusion**
Skills folksonomy/taxonomy and Graph-Based Approaches	
Bastian et al. LinkedIn, LinkedIn Skills: Large-Scale Topic Extraction and Inference [2]	Firstly, the dataset provided by the Oikotie is free written texts of CVs and JDs, where it is not possible to build a template for extracting entity. Secondly, the model used a crowd assisted approach to eliminate semantic duplicates. It is also impossible in this study, because Oikotie does

	not have an interface for such user interaction. Thirdly, this model used a Bayesian classifier, which is an obsolete approach.
Kivimäki et al. A Graph-Based Approach to Skill Extraction from Text [3]	This approach is based on skills folksonomy obtained from the LinkedIn. Unfortunately, the folksonomy is built in English and is not suitable for this research. However, the Wikipedia graph based model, that is used to associate the input document with skills, may be applied to this work.
Wang et al. Skill Inference with Personal and Skill Connections [6]	This system is similar to Kivimäki et al [3], but it applies user profile data to improve performance. This study did not have access to the user data of the Oikotie and this approach cannot be used in this research.
Phuong et al. Large-Scale Occupational Skills Normalization for Online Recruitment [7]	This approach demonstrates an explicit weakness, because of ignoring a semantic context. Moreover, this approach is difficult to implement, and it takes a lot of human resources to create such a taxonomy.
ML approaches	
Sharma. Job Skills extraction with LSTM and Word Embeddings [8]	The approach, that is a combination of word embedding and LSTM technics, showed the best result in extraction skills from unstructured texts. Also this method is relatively easy to implement and could be used in this study. However, LSTM part is better to change to newer NLP method such as BERT.
Gugnani et al. Implicit Skills Extraction Using Document Embedding	One of the modules is based on NER to identify keywords and extract entities as skills. This approach can be useful in this master thesis work.

and Its Use in Job Recommendation [9]	

The building of a skill folksonomy or taxonomy is a complex and laborious work and is out of focus of this research. In addition, previously created dictionaries contain skills only in English, but the main goal of this work is to develop a skills extraction system for Finnish language.

The most appropriate approach for this research is the model developed by Nikita Sharma [8], where Word embedding and LSTM are combined together. The proposed model allows to train a neural network on a small dataset and perform excellent results. In this master thesis a similar approach was developed, but the NLP part exploits a more modern language processing technique called BERT. Also, the problem of skills extraction from unstructured text can be considered as a NER problem. This paradigm was used in the work of Gugnani et al. [9].

3 Theoretical Background

The goal of this chapter is to explore the most common techniques of Natural Language Processing that have been developed to understand human language by computers. The narration continues with the concise introduction of state-of-the-art methods such as Bidirectional Encoder Representations from Transformers (BERT), its operation principle and Named Entity Recognition problem which is solved in the current thesis.

3.1 Natural Language Processing

Nowadays a wide range of tasks for text processing in a natural language is in high demand in different spheres. However, teaching machines to understand natural human language is quite tedious work. The field of computer science where machine learning and text analysis overlap is known as Natural Language Processing (NLP).

There is a set of common NLP tasks such as language translation, text classification, named entity recognition and others. In contrast to the image processing field, regular NLP tasks have been solved using classic machine learning algorithms such as Bag of words or Stemming, and showed results that were not too inferior to state-of-the-art solutions. Classic solutions required thorough consideration of architecture and manual collection and processing of features. Nevertheless, a while ago neural networks began to defeat classic models and formed a general approach for solving NLP problems.

The implementation of a complex NLP task usually requires building a pipeline consisting of multiple steps. In most of the cases the feature set and processing steps are almost the same. Only the last steps are different and involve a neural network. Thus, it forms a uniform pipeline, that includes the following processes:

- Text cleaning is the removing of unnecessary signs and symbols such as line breaks, html tags and etc.
- The next two steps of the pipeline are segmentation and tokenization that is the process of splitting text into sentences or individual words (tokens) respectively.
- The step of calculating the representation of each token usually occurs in either of the following forms:
  - The first one is to calculate context-independent representations of tokens, which include different word embedding models, Part of Speech tagging etc.

  - The second one is context-sensitive tokens representations that contain information not only about the token, but also about its neighbors. These representations are usually defined utilizing RNN, LSTM, GRU etc.

- The last step is fine-tuning a model depending on the goal. For example, a model for classifying or generating new texts.

The given example of a pipeline is not the only one possible. To solve a specific task, some steps can be excluded or new ones added such as stop words removal or parsing dependencies. However, this pipeline contains the most general steps and approaches that allow to derive practical value from given data with the help of NLP.

Natural human language presents an abundance of polysemy and complex semantics. Depending on the context a word may have completely different meanings. For instance, the word "organ" may be understood as part of a human's body or as a large musical instrument. Thus, the same word may be assigned different representations in various contexts. In this case, context-sensitive representations have obvious advantages, especially for the skills extraction task that this thesis aims to implement. Therefore, this section continues with the exploration of models that take context into account. The original contextual models such as RNN and LSTM are investigated and their drawbacks are exposed. Afterwards, BERT, its building blocks and strengths over previous models are explained.

3.1.1 Sequential Models and Problem of Long-Term Dependencies

Models that perform sequence transduction [14] are called Sequence-to-sequence (seq2seq). Sequence transduction is a process that transforms input sequences to output sequences. Seq2seq models are rather versatile and utilized in a variety of NLP tasks, such as machine translation, speech recognition, question-answering system, and others.

For seq2seq models, it is important to have some kind of a memory, that allows information about dependencies and connections in a sentence to persist. For that purpose, a Recurrent Neural Network (RNN) utilizes loops. The part of RNN is presented on the in Figure 3. On the left hand side, the network A is processing the input x_t and output h_t. At the same time the loop may be represented in an unrolled form as a set of multiple copies of the same network, each passing a message to a successor. This approach allows an RNN to transfer the information of previous word to the next network, which can utilize and process this data about context.

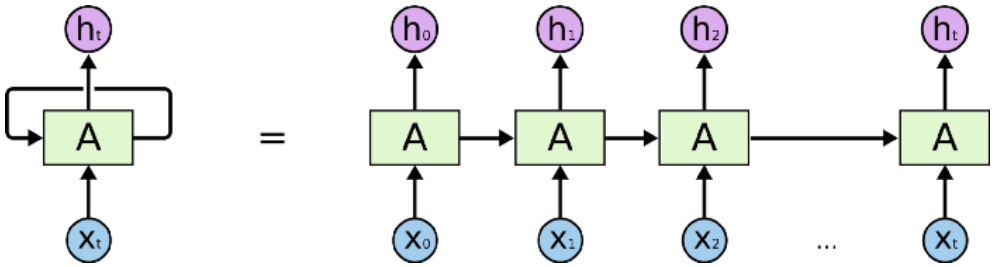

Figure 3. An unrolled recurrent neural network [15].

The Figure 4 demonstrates the basic architecture of a seq2seq model that is based on two RNNs. The top row of the blocks corresponds to an encoder and the bottom one is a decoder. Let us consider this model using a machine translation task as an example. The encoder receives a sentence in language A as an input, that contains words x_i, and compresses it into a hidden state vector h_i. The hidden state is known as the context vector, that contains information about the input sequence. The decoder receives the last hidden state of the encoder and generates words y_i in language B as output.

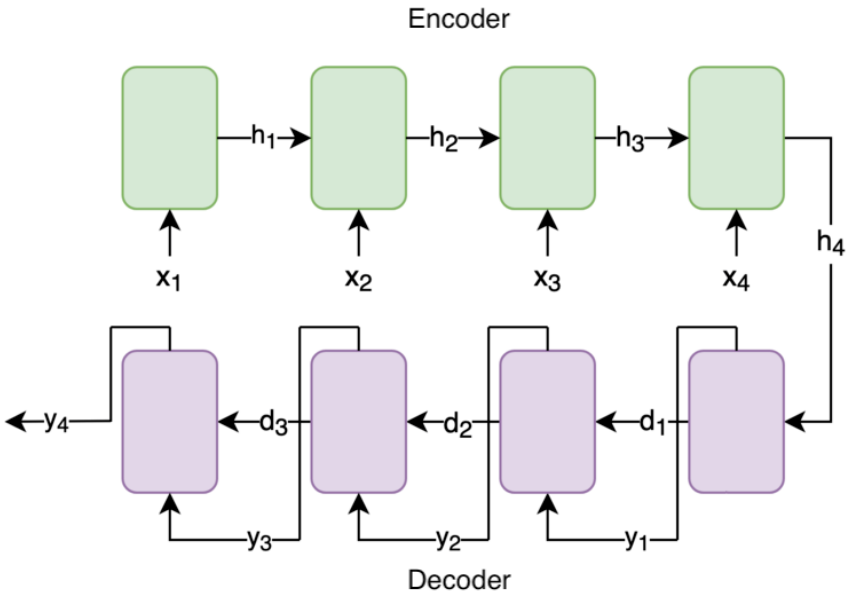

Figure 4. Example of basic seq2seq model.

In the situation when the distance between a word and relevant text around it is not too long, RNN is able to memorize the context. However, in cases that require more context, this gap becomes much bigger and RNN is very ineffective. This is due to the fact that the longer the input sequence, the more likely context information will be lost at one step in the sequence. In theory, RNNs could learn this long-term dependencies, but in practice, it is quite difficult task [16].

Long-Short Term Memory (LSTM) is a particular type of RNN that attempts to solve the problem of long-term dependencies. LSTM has a mechanism that is able to selectively remember or forget significant and insignificant context. This mechanism is called cell states and presented in the Figure 5 as a horizontal line running through the top of the cells. In our case each cell receives a word as input x_t, the state and the output of the previous cell. The cell processes these inputs and then based on them, it generates a new cell state and output. Due to the gates mechanism an LSTM is able to remove or add information to the cell state. Gates consist of a sigmoid neural network layer and a pointwise multiplication operation. The outputs of sigmoid layer are values from 0 to 1, that describe how much of each component should be let through. A value of zero means "let nothing through," while a value of one means "let everything through". An LSTM has three of these gates to protect and control the cell state. Because of this cell state mechanism, the context that is important can be transmited from one word to another during the processing of a text or a sentence.

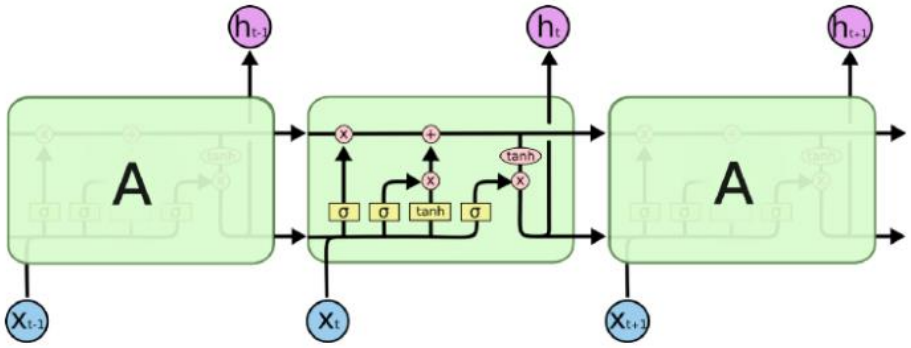

Figure 5. Long-Sort Term Memory network [15].

Despite the cell state mechanism, when a sentence is too long LSTM faces the same problem as RNN. Moreover, the probability of preserving the context for a word that is far from the current word being processed decreases exponentially with a distance from it.

Seq2seq models also meet another problem, that rises from the sequential nature of the model architecture. The model processes a sentence word by word, which prohibits parallelization. In addition to that, there is no explicit modeling of long and short range dependencies.

3.1.2 Attention and Self-Attention

The Attention mechanism was developed to solve problems of seq2seq models described above. In broad terms, the main idea of this technique is to allow the model to focus on certain elements of the input and output sequences when processing the data.

There are two different types of attention mechanism:

- General Attention takes into account the dependences between the elements of input and output.

- Self-Attention constructs interdependences between only the input elements.

The General Attention easily solves the problem of long-term dependencies that occurs in consequence that the last hidden state of the encoder is used as the context vector for the decoder. The attention mechanism allows the decoder to use information obtained not only from the last hidden state, but also from any hidden state of any element of the sequence. In this case the decoder is able to selectively distinguish certain elements from input sequence to produce the output.

The diagram in Figure 6 shows an attention mechanism added between the RNN Encoder and Decoder. The attention mechanism is an ordinary single-layer neural network that uses hidden states $h_t, t = 1 \dots m,$ as input, as well as a vector d that contains a certain context that depends on a specific task. In the case of seq2seq models, the vector d will be the hidden state d_{i-1} of the previous decoder iteration. The output of attention layer will be the score vector s that is estimated based on the hidden state h_i which acquired the most "attention". The $softmax$ [18] is used to normalize s values and has the following properties: $\forall s$: $\sum_{i=1}^{n} softmax(s)_i = 1$, $\forall s, i$: $softmax(s)_i >= 0$. The result of the attention layer is $c = \sum_{i=1}^{m} e_i h_i$, which contains information about all hidden states of h_i in proportion to the e_i score.

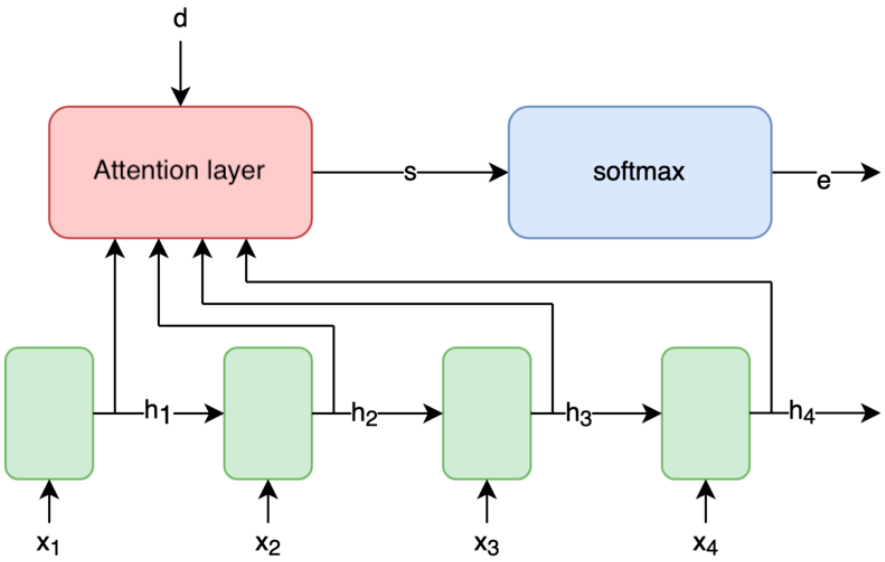

Figure 6. General attention mechanism in RNN.

Using the attention mechanism, the decoder is focused on certain hidden states. In cases of machine translation this feature helps a decoder to pay attention to a correct context of a given word while translating the text from language A to language B.

The main difference of Self-Attention from General Attention is that the former one draws conclusions about dependencies solely between input data. This approach is more effective in machine translation tasks. Self-Attention made it possible to abandon the use of RNNs and replace them with conventional neural networks in combination with the Self-Attention mechanism in the transformer architecture. As an example, let us consider the following sentence: "The animal didn't cross the street because it was too tired". The result of the Self-Attention algorithm for the word "it" is shown in Figure 7. The resulting vector corresponds to the relationship of the word "it" with all other words in the sentence. It can be easily seen that the Self-Attention mechanism found the strongest relationship between the words "it" and "animal". This conclusion can be intuitively explained from a human point of view, which allows machine learning algorithms using this approach to better solve the problem taking into account contextual relationships.

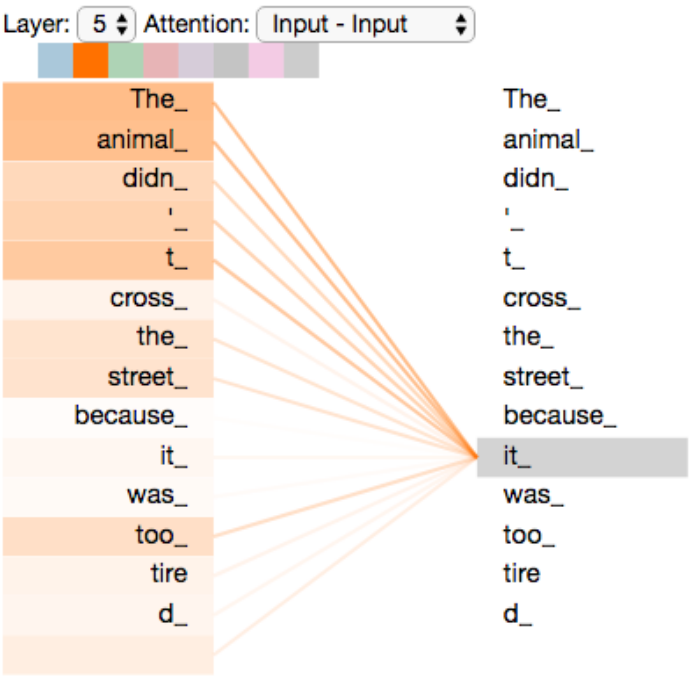

Figure 7. Self-Attention example.

Self-Attention improves performance because the model is able to remember a position of each word in the input sentence. In addition, self-attention allows the model to look at the context for clues that can help to improve the encoding of words. This technique is able to achieve even greater results as part of the Transformer architecture.

3.1.3 Transformer

Self-Attention mechanism perfectly solves the problem of long and short dependencies. However, the issue of processing inputs (words) in parallel remains unresolved. For a large corpus of text, this issue increases the time spent processing the text. In its turn, Convolutional Neural Network (CNN) allows each word in the input to be processed simultaneously. The same idea was implemented in the Transformer architecture.

Transformer architecture is based on encoder-decoder components and self-attention mechanisms, dispensing with recurrence and convolutions entirely. Transformer typically consists of six encoders and six decoders. Architecture of all encoders and decoders are similar and presented in Figure 8.

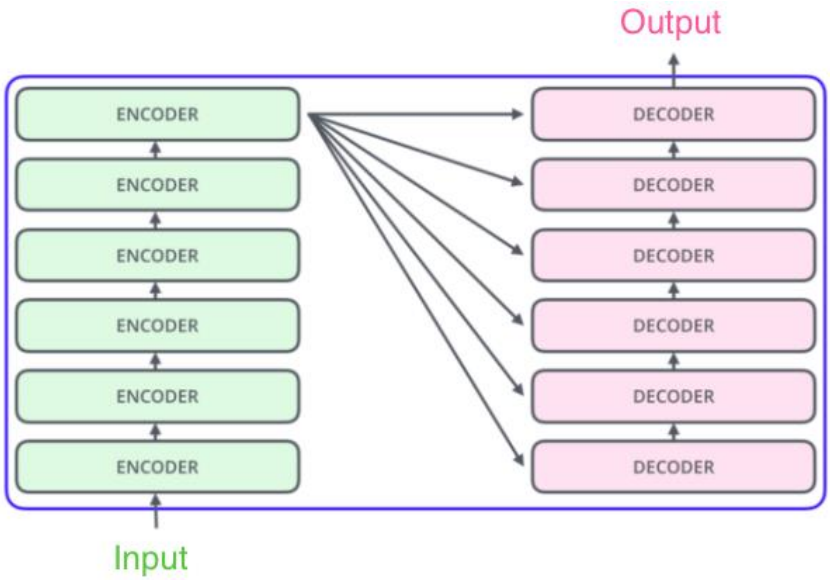

Figure 8. Transformer architecture [17].

Each encoder encompasses of two layers: Self-attention and a Feed-forward Neural Network as shown in Figure 9. The first encoder accepts input data as vectors. Usually words are encoded into vectors by an embedding algorithm. Also a first encoder takes positional information. It is necessary for the Transformer to be aware of the order of the sequence, because no other part of the Transformer makes use of this [17].

The self-attention mechanism adopts a set of input encodings from the previous encoder and weighs their relevance to each other to generate a set of output encodings. In other words, the self-attention helps the encoder look at other words in the input sentence as it encodes a specific word. Then the Feed-forward Neural Network processes each output encoding separately.

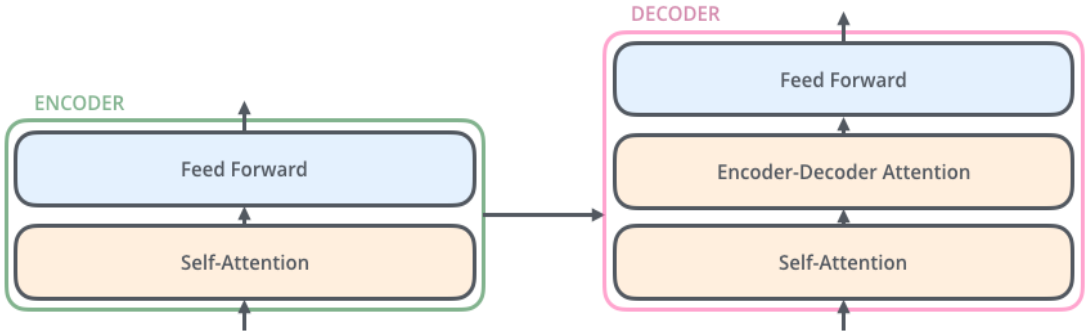

Figure 9. Transformer encoder-decoder architecture [17].

The decoder features similar architecture to the encoder, but there is an additional layer. This layer is essentially an attention mechanism over the encodings (Encoder-Decoder Attention), which draws relevant information from the encodings generated by the encoders. The Encoder-Decoder Attention helps the decoder to focus on relevant parts of the input sentence.

Although, there are some obvious advantages of Transformers over seq2seq models, but Transformers still feature the following limitations:

- The Attention mechanism is able to work with only a fixed-length input, divided into segments or chunks in advance.
- The chunking of input leads to a partial loss of context.

3.2 Bidirectional Encoder Representations from Transformers (BERT)

BERT is a bidirectional state-of-the-art language model with a transformer architecture, replacing sequential RNNs, with a faster approach based on the Attention mechanism. The model is also pre-trained on two unsupervised tasks: modeling language masks and predicting the next sentence. This allows utilizing the pre-trained BERT model, fine-tuning it for specific tasks, such as text classification, question-and-answer systems, and many others.

The initial version of BERT published by Google AI Language [18] presents two main models shown in Table 3. The comparison of two main BERT models.. This model is multilingual and supports 104 languages, including Finnish.

Table 3. The comparison of two main BERT models.

Model name	**BERT BASE**	**BERT LARGE**
Number of transformer blocks (L)	12	24
Hidden layer size (H)	768	1024
Attention heads (A)	12	16

Both BERT models are built on the basis of stacks of encoders as shown in Figure 10. Each encoder block represents a more complex model architecture. Unlike directional models, BERT utilizes pre-trained Transformer encoders that allow it to read the entire text at once, rather than a sequence of words. Moreover, BERT has more attention

heads, larger feedforward-networks, than the default configuration in the implementation of the Transformer (6 encoder layers, 512 hidden units, and 8 attention heads).

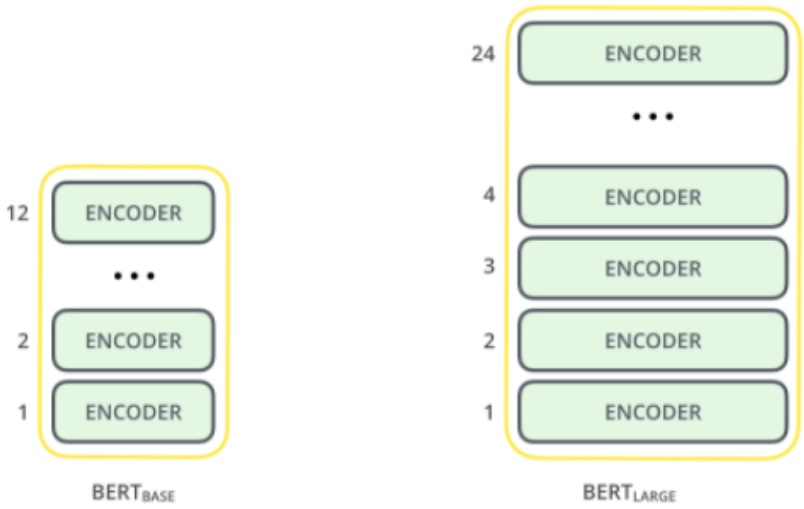

Figure 10. BERT encoder stacks [19].

One of the advantages of BERT is a bidirectional model, which allows it to train in the left and right directions or utilize a context in all layers as shown in Figure 11.

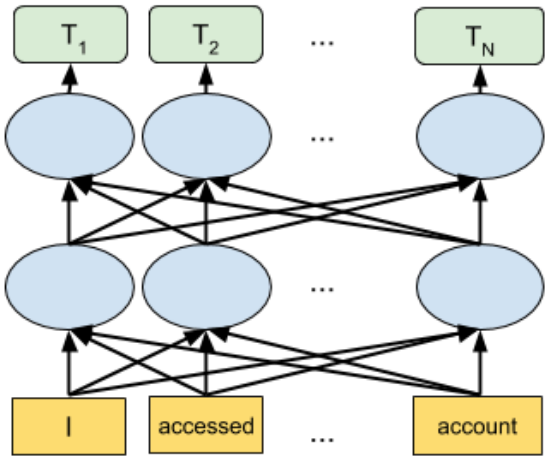

Figure 11. Bidirectional model [19].

Pre-training

BERT uses a specific format for the input to pre-train the model. As an input the model receives the set of tokens. These tokens are the sum of three different embeddings, that are needed to transform textual data into vectors. For this task BERT utilizes the

WordPiece tokenizer [20]. The tokenizer includes vocabulary of the most common words and letters of the alphabet. If some word does not belong to vocabulary, the tokenizer splits the word into pieces until they are found in the vocabulary. The tokenizer utilizes ## (double hash) sign to remember which tokens are pieces of a word. For instance, it divides token “playing” to “play” and “##ing”. The number of tokens in the sequence cannot exceed 512.

To each sentence (sequence) BERT applies the following set of tokens:

- [CLS] : A classification token at the beginning that is usually used in conjunction with the softmax layer for classification tasks. Otherwise, it can be safely ignored.
- [SEP]: A sequence delimiter token which is used at pre-training for next sentence prediction task. This token must be used when sequence pair tasks are required. In case of a single sequence, this token is appended at the end.
- [MASK]: Token utilized to predict masked word based only on its context. Only used for pre-training.

The input for BERT consists of three embedding layers, see Figure 12. The layer of token embeddings contains the vocabulary IDs for each of the tokens. Sentence Embeddings is a numeric class to distinguish between the first and the last part of a sequence. Transformer positional embeddings provide a position of each word in the sequence.

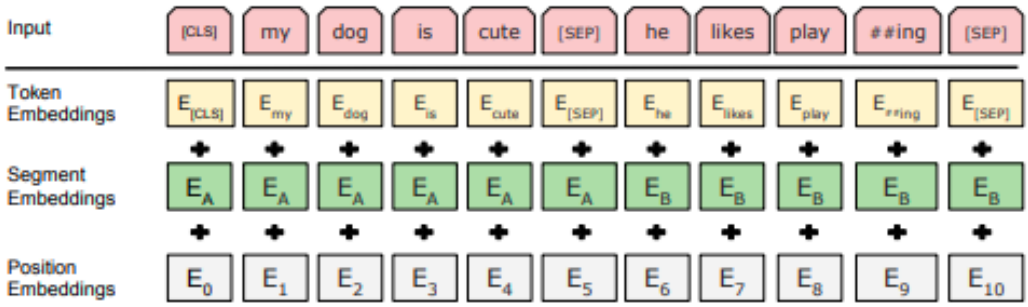

Figure 12. Three embedding layers for BERT input. [18].

The configuration of the BERT model for a specific task usually consists of two steps: pre-training and fine-tuning. The BERT is pre-trained utilizing two unsupervised tasks as Masked Language Modeling (MLM) and Next Sentence Prediction (NSP).

Masked Language Modelling (MLM)

There is the restriction of unidirectionality in the basic form of the Transformer. Therefore, to eliminate this issue the BERT is pre-trained with an MLM task, that enables the Transformer to unite left and right context. In this task, 15 % of words [15] are randomly replaced with a special token in each input sequence. Among these tokens in 80% of times words are swapped with a [MASK], 10% – with random words [22], and the rest are left unchanged as shown in Figure 13. The main goal of this task is to predict the vocabulary IDs of masked words, based on its context.

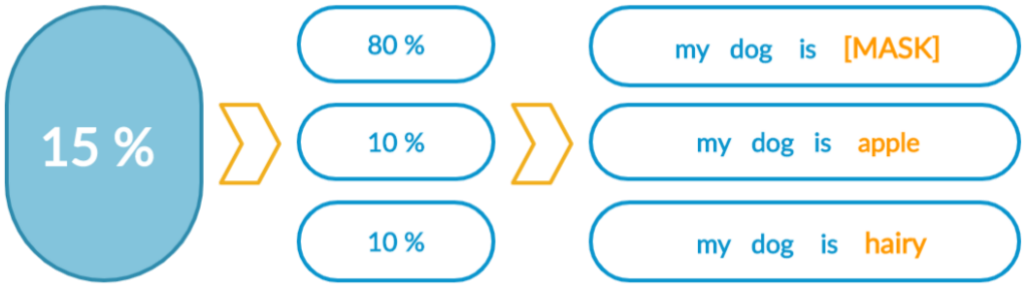

Figure 13. Masked Language Modelling task.

The Softmax activation function [23] generates the output. This function gives the probability of each word in the sentence (sequence).

Next Sentence Prediction (NSP)

BERT utilizes an NSP task to pre-train text-pair representation. The model receives two sentences and aims to predict whether the second sentence in the pair is the next sentence. For 50 % of pairs [21], the second sentence is the next to the first one. In the other 50 % of pairs, the second sentence is replaced by a random sentence from the text. A label for the first case is “IsNext” and “NotNext” is for the second situation [22]. The example of this task is presented in Figure 14.

Moreover, BERT utilize Transformer to make prediction for the next sentence. After that outputs from tokens are provided to the classification layer. The next sentence is predicted applying the probability of every word by means of Softmax activation function [23].

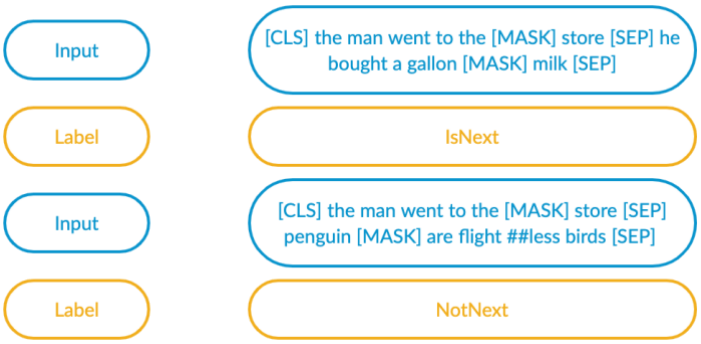

Figure 14. Next Sentence Prediction task.

Fine-tuning

BERT can be applied for a wide range of language tasks such as text classification, question answering task, named entity recognition and others. The fine-tuning of BERT for a specific task usually involves adding a one layer to the core model as shown in Figure 15. For instance, in the case of a classification task, an additional layer is added on the top of the Transformer output. In the fine-tuning process, BERT layers remain unchanged, while the weights of the extra layer are trained.

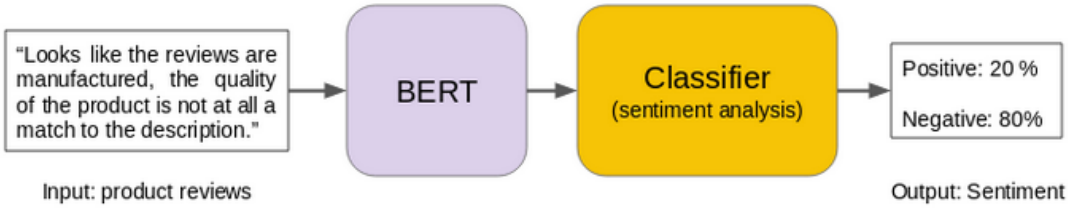

Figure 15. Fine-tuning BERT [23].

3.3 Named Entity Recognition (NER) with FinBERT

One of common NLP problems and underlying subtasks of information extraction is Named Entity Recognition (NER) [26]. There are many studies on this topic, but most of them are focused on English language. One of the last works related to NER in Finnish was published by Ruokolainen at el [27] and based on BiLSTM-CNN-CRF model [28]. This study applies a BERT model to solve a NER task for extracting information from an unstructured Finnish text.

The main purpose of NER task is to recognize different types of entities from the input corpuses of texts. Entities can include a person's name, organization, date, soft and hard

skills etc. NER task belongs to a supervised machine learning, which requires labeled training dataset. For that purpose, words in a sentence are labeled with a predefined list of tags, such as:

- [O] - no meaning,
- [B-PER]/[C-PER] - person name,
- [B-ORG]/[C-ORG] - organization name.

A trained NER model labels each word of a given sentence. For example, “Leonardo[B-PER] DiCaprio[C-PER] will[O] launch[O] his[O] new[O] TV-show[O] in[O] Netflix[B-ORG] Shows[C-ORG].[O]”

BERT is state-of-the-art NLP method, which enables to solve information extraction tasks such as NER. In this case, an additional layer is attached on top of the BERT model output. This extra layer is a regular densely connected neural network layer. The model training is executed only for the extra layer and parameters of BERT remain unchanged.

Moreover, there is Google’s open source BERT model for Finnish language (FinBERT) developed by Turku University. This model outperforms the previously released version of multilingual BERT [25]. The comparison of the FinBERT model with previous ones for the Document classification problem is presented in Figure 16. The figure demonstrates the growth of models’ accuracies as a function of the training dataset size.

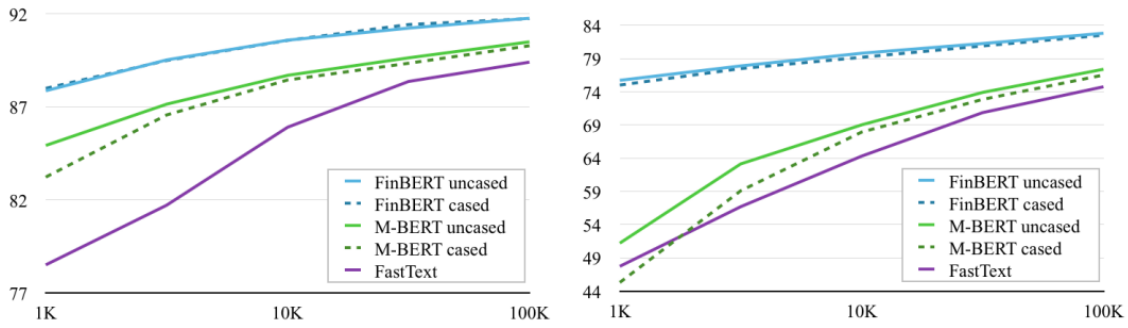

Figure 16. Text classification accuracy with different training datasets: Yle News (left diagram) and Ylilauta online discussion (right diagram).

In addition, Figure 16 presents cased and uncased types of FinBERT models. The former one is trained on a regular text, i.e. capital and lowercase letters, and the later one is trained on a text, which was lower cased. Usually, cased models are utilized when the letter casing is useful for the task. As it turned out document classification problem was

not very sensitive to either of the FinBERT model types. In opposite to that, NER task is more dependent on letter cases, since the extracted terms often contain capital letters. The comparison of the results of BERT models for the NER problem is presented in the Table 4. The FinBERT cased model outperformed other models in the NER task.

Table 4. NER results [25].

Model	Accuracy
FinBERT cased	92.40%
FinBERT uncased	91.50%
Multilingual BERT cased	90.29 %
Multilingual M-BERT uncased	89.06%

The advantage of FinBERT models in the Finnish language domain is mainly caused by the larger amount of training text corpuses obtained from Yle. Cased and uncase FinBERT models contains around 50,000 words. It enables the model to have a rich understanding of a Finnish text. From the examples shown in Table 5. Examples of tokenization with different vocabularies [25]. [25], it can be seen that FinBERT is superior to multilingual BERT model in splitting a sentence into tokens.

Table 5. Examples of tokenization with different vocabularies [25].

Vocabulary	**Example**
FinBERT cased	Suomessa vaihtuu kesän aikana sekä pääministeri että valtiovarain ##ministeri.
FinBERT uncased	suomessa vaihtuu kesän aikana sekä pääministeri että valtiovarain ##ministeri.
Multilingual BERT cased	Suomessa vai ##htuu kesä ##n aikana sekä p ##ää ##minister ##i että valt ##io ##vara ##in ##minister ##i.
Multilingual BERT uncased	suomessa vai ##htuu kesän aikana sekä pää ##minister ##i että valt ##io ##vara ##in ##minister ##i.

In addition to that, the FinBERT cased model demonstrates superior results for all downstream tasks. Therefore, the FinBERT model with cased vocabulary is utilized in this research.

4 Implementation Details

This chapter presents a solution for skills extraction from an unstructured text based on approaches and methods described in the theoretical background and current state of analysis sections. The narration goes through an implementation pipeline, namely, data preprocessing, model architecture, performance evaluation and other aspects of machine learning cycle.

4.1 Data Pre-processing

This section describes the process of how raw dataset is prepared for the purpose of a data extraction task. The process includes data cleaning and filtering, which removes erroneous or corrupted data. Then the section discusses manual annotation process and annotation guidelines.

4.1.1 Cleaning and Filtering Data

Oikotie Työpaikat provided a dataset of job advertisements published on tyopaikat.oikotie.fi since March of 2014 till November of 2018. The dataset contains about 300 000 job advertisements in a tabular format. The example of raw data is presented in Figure 17. Each job advertisement is defined by a unique identifier – “ID”, job name – “NAME”, job description – “JOBDESCRIPTION” and creation date – “CREATEDATE_ORA”. Job descriptions are mostly in a single language and the majority of them is in Finnish. In addition, there is an amount of job descriptions in two languages.

	ID	NAME	JOBDESCRIPTION	CREATEDDATE_ORA
0	757390	Business Process Expert, Wood purchasing and f...	Group Services\r \r Metsä Group’s Gr...	03.01.2014
1	757430	Levyseppä-hitsaaja	Haemme asiakkaallemme Salon seudulle \nlevysep...	03.01.2014
2	757030	Järjestelmäasiantuntija / System Specialist	<p>Job ID: 157518 <stro...	02.01.2014
3	756970	Kirjanpitäjä	Haemme asiakasyrityksellemme tilinpäätöstaitoi...	02.01.2014
4	757431	Levyseppä-hitsaaja	Haemme asiakkaallemme Salon seudulle voimassa ...	03.01.2014
...	...	...	...	...
293369	1200997	Osa-aikaisia porrassiivoojia Helsinkiin	Etsitkö itsenäistä ja liikkuvaa työtä? Meillä ...	09.11.2018
293370	1205375	Toimistoassistentti, Työttömyysvakuutusrahasto...	Etsimme asiakkaallemme Työttömyysvakuutusrahas...	24.11.2018
293371	1206278	Logistiikkatyöntekijä	<p>Etsimme asiakkaallemme Turkuun logistiikkat...	27.11.2018
293372	1206279	DNA Palvelumyyjä Kuopio (7.1)	<p>DNA:n puhelinpalvelukeskuksissa työskentele...	27.11.2018
293373	1206280	Huoltoasentaja vakituiseen työsuhteeseen	<p>Barona Kunnossapito Oy hakee Helsinkiin huo...	27.11.2018

293371 rows × 4 columns

Figure 17. Example of row data.

Job advertisements are imported and published to Oikotie Työpaikat through different channels. This brings on one hand flexibility for customers, but on the other hand an issues of different data formats. Some of the job descriptions are in plain text, while the others contain various special symbols or given in an html format. It is also worth mentioning that the fraction of data is a pour noise, because some descriptions are suspiciously short and contain nonsense text. Therefore, the data preprocessing and cleaning are essentially important.

The first step is to remove html tags and special symbols from textual data. Jupyter Notebook and Python libraries were utilized to build a data cleaning process. Python BeautifulSoup library is a common way to convert an html text into a human readable text. Regular expressions allowed to add dots to the ends of sentences or remove special symbols, which constitute a noticeable part of noise in textual data. Such symbols are soft hyphens, multiple line breaks and others. The example of cleaned data is presented in the Figure 18.

```
id,title,description,desc_langs
757430,Levyseppä-hitsaaja,"Haemme asiakkaallemme Salon seudulle levyseppä-hitsaajaa ohutlevytuotteiden hitsaukseen. Olet koulut
ukseltasi levyseppä-hitsaaja sekä omaat jo jonkin verran työkokemusta erityisesti rosteri ohutlevytuotteiden hitsauksesta. Työ
on päivätyötä, mutta hakijalla pitää olla myös 2-vuorovalmius. Työpaikan sijainnin vuoksi tarvitset oman auton työmatkoja varte
n. Työ alkaa tammikuussa ja tehtävässä on mahdollisuus pidempään työsuhteeseen.",fi
756970,Kirjanpitäjä,"Haemme asiakasyrityksellemme tilinpäätöstaitoista kirjanpitäjää, toimistonhoitajaa. Tehtävässä hoidat itse
näisesti kirjanpidon ja teet tilinpäätöksen sekä vastaat kk-tulosraportista sisältäen tuloslaskelman ja saldojen täsmäytykset.
Hoidat myös veroilmoituksen ja intrastat ilmoitukset tuonnista. Toimiston hoitajana tehtäviisi kuuluu myös laskutus, maksut ja
tilitysten tekeminen sekä monipuolisesti erilaisia toimiston rutiinitehtäviä ja asiakaspalvelua. Sinulla on kaupallinen koulutu
s ja vankka ammattitaito kirjanpitotehtävistä sekä taloushallinnon muista tehtävistä. Arvostamme huolellisuutta ja tarkkuutta s
ekä oma-aloitteista ja itsenäistä työskentelytapaa. Työ alkaa sopimuksen mukaan, sopivan henkilön löydyttyä Tarvitset työmatkoj
a varten oman auton. Tehtävä on aluksi vuokratyösuhde, mutta tehtävässä on mahdollisuus vakituiseen työsuhteeseen. Lisätietoja
tehtävästä antaa VMP Groupissa HR-koordinaattori Seija Rahkonen, puh 02-728 1450(ark klo 9-15). Lähetä hakemuksesi ja ansioluet
telosi sähköpostilla 10.01.2014.mennessä osoitteeseen salo@vmp.fi. Merkitse otsikoksi ""Kirjanpitäjä"". Aloitamme haastattelut
jo hakuajan kuluessa.",fi
```

Figure 18. Example of cleaned data.

After cleaning the next step of preprocessing is data filtering. First of all, empty or suspiciously short JDs are ruled out. Specifically, if JDs text length is less than 100 characters. Secondly, there is a number of JDs present in two languages, Finnish/Swedish or Finnish/English. For the sake of simplicity and the goal to develop a model for Finnish language, JDs posted in multiple languages or any other language except Finnish were excluded from the consideration.

After cleaning and filtering, there are 225 865 job descriptions left in the dataset. This amount is more than enough to train a model to extract skills.

Finally, the JDs were saved as separate files. This was a requirement for a word labeling tool. In addition, the developed neural network model processed each JD separately.

4.1.2 Data Labelling and Skill Definition

For the NER task, each word in the text must be marked with a label that tells the model the specific value of the token. This approach allows a model to analyze each token in a sequence and extract necessary information. In our case, extract skills from JDs.

In this research, skills required in JDs were manually labeled to create a training dataset. For that purpose, 100 JDs were randomly selected from the cleaned and filtered dataset. Then, these JDs were labeled utilizing TagTog [24], which is a web-based tool for NLP text annotation. It is worth noting, that a job skill is often not a single word, but rather a group of words or a phrase. That is why two different labels were chosen to tag skills in a text. The first label – B-skill – indicates the first word of a skill name. If the skill name is a single word, then B-skill label is the only required annotation. On the other hand, if a skill name is a phrase, then the first word is labeled as B-skill and the rest of the skill name words are marked with C-skill labels. Figure 19 presents the example of labeled JD, where:

- [B-skill] - beginning skill.
- [C-skill] - continuation skill.

It turned out that the manual data labelling is quite a challenging task. There are multiple reasons for that. Firstly, it was often hard to distinguish skills required for a job from job daily tasks or even a job title. Secondly, JD is rather a free style text. Thirdly, it was difficult to preserve the same labelling logic over time. Therefore, a team of experts in this field would improve the quality of the annotation process.

What about the “skill” term itself in this domain of research? Skill is commonly defined as an ability to do a work or action. Usually, skills are directly mentioned in JDs. Then such skills are called explicit. However, it is not always the case. Often enough, there is the description of a required ability for a task without the factual skill name. Such skills are called implicit, and the extraction of implicit skills is a separate field of study. The current thesis considers the extraction of explicit skills.

Based on the knowledge obtained in Section 2, the ML model should be able to extract Soft and Hard skills. Soft skills include communication skills, collaborative mindset, curiosity, and others. Hard skills are more related to a specific work industry, for example, Python programming, Keras, TensorFlow, PyTotch, Deep learning etc. In addition, the

skill term is extended with various certificates, passports and diplomas, own car and different important tools that a job description requires.


S-Pankin tavoitteena on tuottaa Suomen paras digitaalinen pankkipalvelukokemus. Haluamme tarjota asiakkaillemme ylivertaisen helppoja ja hyödyllisiä tapoja hoitaa arjen raha-asioita. Etsimme S-Pankkiin ihmisiä, jotka haluavat haastaa alan totuttuja käytäntöjä ja hyödyntää digitalisaation tarjoamia mahdollisuuksia. Haemme nyt Digitaalisen datan analyytikkoa tuloshakuiseen Digitaalisen myynnin porukkaan. Tehtävänäsi on tuottaa näkemyksiä sekä tietoa digitaalisten palveluiden asiakaskäyttäytymisestä myynnin, markkinoinnin, asiakkuusviestinnän ja palvelukehityksen tarpeisiin. Pyrit löytämään uusia mahdollisuuksia S-Pankille analysoimalla digitaalista dataa ja parannat havainnoillasi asiakaskokemusta ja myyntiä digitaalisissa kanavissa. Tärkeää on, että osaat löytää oikeat tiedonjyvät datassa ja osaat tuottaa niistä konkreettisia toimenpide-ehdotuksia. Toimit tiiviisti digitaalisen liiketoiminnan ytimessä. Vastuullesi kuuluu digitaalisen analytiikan yhdistäminen kantadataan ja yhdistetystä tiedosta johtopäätöksien tuottaminen digitaaliselle liiketoiminnalle. digitaalisten palveluiden asiakaskäyttäytymisen ja asiakaspolkujen ymmärryksen ja näkemyksen tuominen digitaalisen liiketoiminnan tueksi markkinointidatan mallinnus ja analysointi sekä attribuutiomallinnus eri digitaalisista kanavista, kuten display, uutiskirjeet, haku, offline CRM-dialogien toimivuuden mittaus ja analysointi hyödyntämällä useita datan lähteitä, kohderyhmäpoimintojen suunnittelu ja toteutus raporttien määrittely ja tuottaminen digitaalisen liiketoiminnan tarpeisiin osallistuminen päätöksentekoon tarjoamalla näkemystä digitaalisten palveluiden toiminnasta ennustavan analytiikan ja koneoppimisen mallien kehitystyöhön osallistuminen. Tehtävässä menestyminen edellyttää analyyttisyyttä, ymmärrystä numeroiden päälle tilastotieteellistä, matemaattista taustaa ymmärrystä tietorakenteista, tietokannoista ja tilastolliseen analyysiin käytettävistä työkaluista sekä ohjelmointiosaamista, kuten SAS, Matlab, Python, C, R, BigQuery ymmärrystä digitaalisesta datasta ja asiakastiedon hyödyntämisestä monimutkaisten ongelmien hahmotus- ja ratkaisukykyä sekä kykyä tuottaa itsenäisesti käyttökelpoisia analyyseja ja raportteja kokemusta visualisointi-, raportointityökaluista (esimerkiksi SAS Visual Analytics, Qlik, Tableau, Cognos). Aiempaa työkokemusta sinulle on kertynyt analyytikkona, tilastotieteilijänä tai vastaavissa tehtävissä, joissa digitaalisten palveluiden kehittäminen ja datan kanssa työskentely ovat olleet arkipäivää. Tässä tehtävässä menestyy todennäköisesti parhaiten tekijä, joka on varustettu ennakkoluulottomalla asenteella, ei kaihda haasteita ja jolle myös rutiininomaiset työtehtävät ovat mielekkäitä. Tarjoamme haasteellisen ja monipuolisen tehtäväkentän digitalisoituvassa pankissa hyvät kehittymismahdollisuudet vahvasti toimialaa uudistavassa organisaatiossa innostavan työyhteisön S-ryhmän kattavat henkilöstöedut. Lisätietoa tehtävästä antavat Esa Peltonen (esa.peltonen@s-pankki.fi, 040-5600670), digitaalisen analytiikan päällikkö, ja Jarkko Kantoluoto, digitaalisen myynnin johtaja (050-3884081, jarkko.kantoluoto@s-pankki.fi). Meidät tavoitat parhaiten arkena klo 8 9 sekä klo 1617. Jos haluat olla mukana rakentamassa Suomen parasta digitaalista pankkipalvelukokemusta, lähetä hakemuksesi palkkatoivomuksineen viimeistään 25.9.2016 oheisella hakulomakkeella. Huomioimme ainoastaan hakulomakkeella saapuneet hakemukset.

Figure 19. Example of labeled data.

4.1.3 Parsing Annotation Files and Preparing Input for BERT

After the completion of the text annotation phase, skill labels are available in the form of a json document. The example of this data is presented in Figure 20. As it may be seen from the picture, json files contain lots of excessive and irrelevant fields for the NER task. Only ‘classId’ and ‘text’ fields were extracted and converted to a csv format.

The example of converted csv file is presented in Figure 21. These files contain the following columns:

- jid – Job description ID is required to identify the corpus of text from which entities are extracted.

- sentensce_id – Sentence ID is used to determine the sequence in the text.
- word – This is a column containing tokens that are assigned to labels.
- label – Tags that are assigned to each token.

Tokens that are not related to skills are marked with the [O] tag. This tag also labels punctuation marks, such as a dot, comma, dash etc.


```
▾ root: {} 6 keys
  ▾ annotatable: {} 1 key
    ▾ parts: [] 1 item
        0: "s1v1"
    anncomplete: true
    sources: [] 0 items
    metas: {} 0 keys
  ▾ entities: [] 44 items
    ▾ 0: {} 7 keys
        classId: "e_4"
        part: "s1v1"
      ▾ offsets: [] 1 item
        ▾ 0: {} 2 keys
            start: 29
            text: "tilinpäätöstaitoista"
        coordinates: [] 0 items
      ▾ confidence: {} 3 keys
          state: "pre-added"
        ▸ who: [] 1 item
          prob: 1
        fields: {} 0 keys
        normalizations: {} 0 keys
    ▸ 1: {} 7 keys
    ▸ 2: {} 7 keys
```


Figure 20. Example of json document.

	jid	sentence_id	word	label
1	756970	sentence_1	Haemme	O
2	756970	sentence_1	asiakasyrityksellemme	O
3	756970	sentence_1	tilinpäätöstaitoista	O
4	756970	sentence_1	anpitäjää/toimistonhoitajaa	O
5	756970	sentence_1	.	O
6	756970	sentence_2	Tehtävässä	O
7	756970	sentence_2	hoidat	B-skill
8	756970	sentence_2	itsenäisesti	C-skill
9	756970	sentence_2	kirjanpidon	C-skill
10	756970	sentence_2	ja	O
11	756970	sentence_2	teet	B-skill
12	756970	sentence_2	tilinpäätöksen	C-skill
13	756970	sentence_2	sekä	O
14	756970	sentence_2	vastaat	B-skill
15	756970	sentence_2	kk-tulosraportista	C-skill
16	756970	sentence_2	sisältäen	O
17	756970	sentence_2	tuloslaskelman	O
18	756970	sentence_2	ja	O
19	756970	sentence_2	saldojen	O
20	756970	sentence_2	täsmäytykset	O
21	756970	sentence_2	.	O

Figure 21. Example of output file converted to CSV.

Another important point in the preparation of the input data is the fact that BERT is capable of processing an input sequence of no more than 512 tokens. It means that a sentence in a JD cannot be longer than 512 words. Figure 22 shows the distribution of the number of words in JD sentences. As illustrated on the diagram above, the maximum number of words in a sentence does not exceed 40 words, which definitely complies the limitations of the BERT model. Furthermore, there is a peak at the beginning of the diagram, which indicates the presence of sentences of 1-5 words. In most cases, such short sentences are headings.

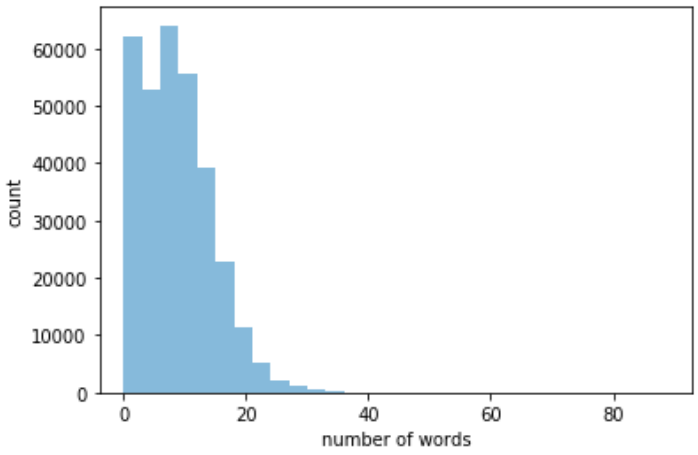

Figure 22. Number of words in a sentence.

4.2 Frameworks and Libraries

As it was presented in Section 3, a version of Google's BERT model for Finnish language demonstrates outstanding results. Therefore, this research is based on FinBERT model. However, the BERT model does not directly support NER. Therefore, this research utilizes NER approach for FinBER described by Devlin et al. [26] and implemented using Keras.

Keras is a high-level API (Application Programming Interface) that can use TensorFlow functions. Keras was designed with convenience and modularity as guidelines. As a practical matter, Keras makes it as easy as possible to create many powerful yet complex TensorFlow functions, and is configured to work with Python without any major changes or settings. This functional API suggests a fast way to create prototypes of modern deep learning models. Moreover, Keras was used to create and train the BERT model and is therefore an important tool that is utilized in this thesis.

TensorFlow is an open source library built for Python by the Google Brain team. TensorFlow compiles many different algorithms and models, allowing the user to implement deep neural networks for use in tasks such as image recognition, classification, and natural language processing.

4.3 Model Architecture and Parameters

The main building block of the machine learning model developed in this study is recently introduced FinBERT with cased vocabulary. Similar to the original version of BERT, the Finnish language model consists of a multi-layer transformer that receives a sequence of tokens as input. The output of FinBERT is a sequence of context-based embeddings of these input tokens. An additional layer for the NER task is a Dense layer that is utilized for fine-tuning developed model. The additional layer produces a probability distribution over output labels as shown in Figure 23.

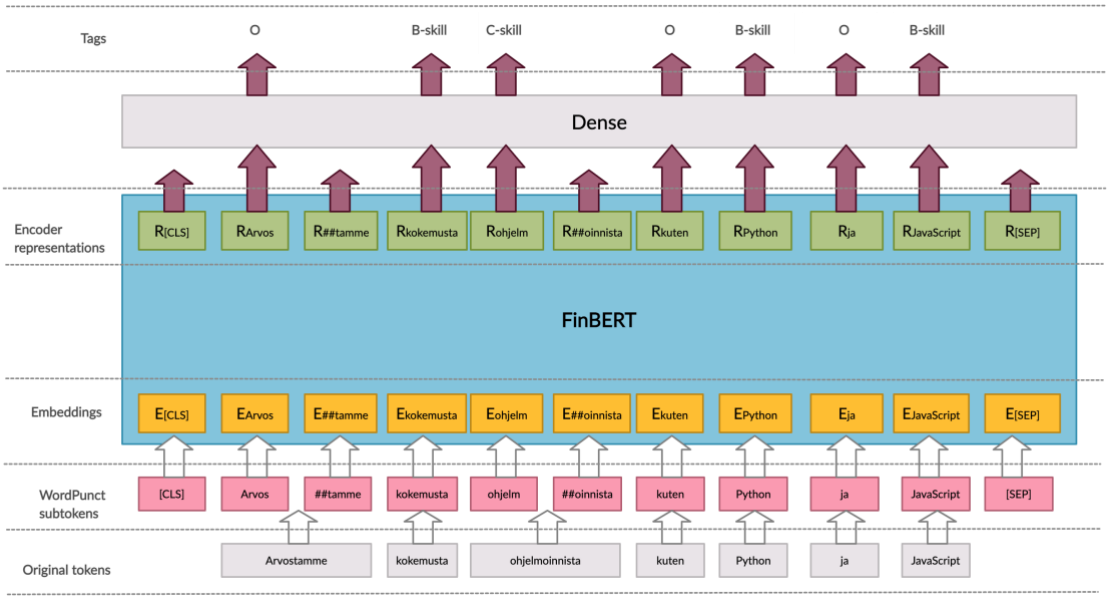

Figure 23. Model architecture.

The NN model was trained on 85 labeled job descriptions. The training process took three hours and was performed on MacBook Pro with 2,8 GHz of Processor Speed and 16 GB of RAM memory. The maximum length of the input sequences was 128 tokens. The developed NN model includes the following set of parameters:

- Batch size: 8
- Top layer learning rate: 5e-5

- Optimizer: AdamOptimizer

The trained model was evaluated based on traditional performance evaluation metrics. Finally, the whole unlabeled dataset containing roughly 200 000 JDs was fed to the NER model. The inferences are presented in the form of cloud of words and job skills demand over time.

4.4 Model Training and Validation

The labeled dataset of 85 job descriptions was randomly split into training and validation sets: 15% of data constitutes validation dataset, the rest is training.

In order to avoid overfitting and at the same time enable good generalization level for the model, learning curves [29] were built and presented in Figure 24 and Figure 25. Learning curves calculated on the metric by which the parameters of the model are being optimized. In our case this metric is a loss and the parameter is the number of epochs. The exploration of the loss was conducted for fifteen epochs.

As Figure 24 and Figure 25 demonstrate the training loss decreases with the increase of the number of epochs. This is a good sign which tells that the model actually trains. However, while training the model starts to memories data samples. This is called overfitting and leads to a significant decrease in model prediction quality on unseen data. In order to avoid overfitting, the model is supplied with a validation dataset.

The validation loss enables to choose the optimum number of epochs for model training. On the same Figure 24, the validation loss gradually goes down and starts to increase after reaching its minimum. While the loss decreases the model still learns hidden patterns in the text. After reaching the minimum the overfitting starts. Therefore, the optimum number of epoch is seven.

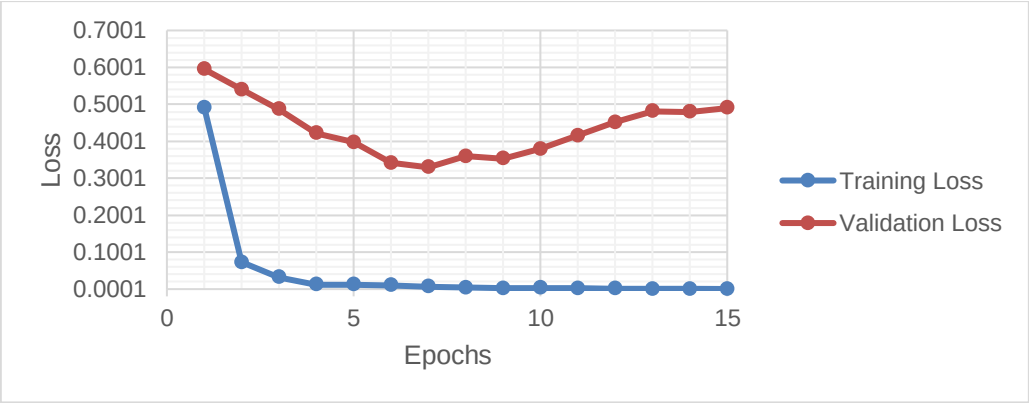

Figure 24. Train and Validation Learning Curves Showing an Overfitting Model.

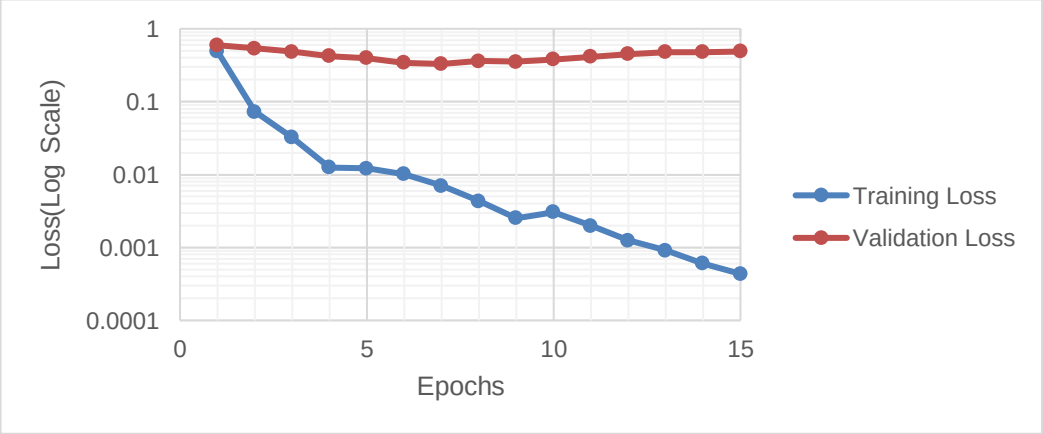

Figure 25. Train and Validation Learning Curves (Log Scale).

As Figure 24 and Figure 25 demonstrate the training and validation curves are located far from each other. Moreover, training curve is close to x axis. Both of these observations emphasize that the model was trained with too soft regularization. Regularization allows a model to avoid overfitting and achieve higher performance. However, the choice of optimal regularization methods and its parameters is a matter of a follow-up research.

4.5 Model Evaluation

The validation dataset was utilized during model training to recognize and avoid the model overfitting. Furthermore, an additional test dataset was deliberately postponed to evaluate the model performance. This test dataset contains 15 JDs, which is about 1541 words. These job descriptions are unfamiliar to the model and are being applied for the first time.

Named Entity Recognition is a type of a classification task. For this kind of task, the main approach for a model performance evaluation is a confusion matrix. The confusion matrix is computed on a set of test data for which true values are known. In the case of binary classification, the confusion matrix is a table with four different combinations of predicted and actual values [30].

On the test dataset, the model recognized 134 phrases as skills. Among these phrases, 91 were correctly identified by the model (TP), 43 – misclassified as skills (FP), and missed 28 phrases (FN). These results are presented in Table 6.

Table 6. Confusion Matrix (Phrase) for the developed model.

		Predicted Value		
		Negative	Positive	
Actual Value	Negative	X (TN)	43 (FP)	X
	Positive	28 (FN)	91 (TP)	119
		X	134	

As it may be noticed from the Table 6, the value of true negatives is empty. This is due to the fact that the concept of a phrase was not introduced in this study and our model was not trained to count the number of phrases in each JD.

Additionally, there is a list of general metrics that are commonly used to evaluate model performance. These metrics are basically calculated from a confusion matrix and include the accuracy, precision, recall and F1-score [30].

$$Accuracy = \frac{TP + TN}{Total}$$

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

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

$$F1\_score = \frac{2 * Recall * Precision}{Recall + Precision}$$

Based on these metrics, it is possible to conclude how well the developed model is able to recognize phrases as skills in unstructured text. The calculated metrics are presented in Table 7.

Table 7. Evaluation metrics (Phrase) for the developed model on test dataset.

Accuracy	Precision	Recall	F1-score
93,64	67,91	76,47	71,94

Moreover, it may seem that values are not quite high. However, it is worth noting that labeling data as skills is a complex and painstaking process that requires a lot of time. According to Section 2, a group of linguistic experts is usually involved in such kind of annotation processes. In the current research, the data labeling was performed by one person with no linguistic education. Therefore, the dataset used in this work theoretically may contain labeling mistakes and ambiguities.

Furthermore, it was decided to do model evaluation on the level of individual words rather than phrases. This is a legit idea, because finding the edges of skill phrases is a quite cumbersome process and depends a lot on the qualification of a person preparing training and testing datasets. As a consequence, the model learned training data might not be able to fully capture deviated patterns in test data. The confusion matrix representing the quality of predictions on the level of words is show in Table 8. The model predicted labels for 1541 words in the testing dataset and 210 out of 238 skill words were correctly assigned to the positive class. Key performance metrics are listed in Table 9.

Table 8. Confusion Matrix (Words) for the developed model.

		Predicted Value		
		Negative	Positive	
Actual Value	Negative	1245 (TN)	58 (FP)	1303
	Positive	28 (FN)	210 (TP)	238
		1273	268	

Table 9. Evaluation metrics (Words) for the developed model on test dataset.

Accuracy	Precision	Recall	F1-score
94,42	78,35	88,24	83,00

As it was expected the model performance on the level of words is higher than on the level of phrases. For instance, Precision grew by over 10% and reached 78%. Other metrics showed higher values as well.

The extraction of individual words makes a lot of sense, since skill phrases explaining the same skill cannot be easily matched for further analysis. This happens because of even a slight difference in the text or an extra descriptive word.

4.6 Results

This section presents examples of analytical and statistical results that can be performed based on the extracted skills from JDs. All analytical results are based on data obtained from JDs published on tyopaikat.oikotie.fi since March of 2014 till November of 2018. As discussed in Section 4.1.2 the developed model allows to extract two types of skills namely soft and hard skills. The model was also trained to fetch entities such as diplomas, certificates, and other necessary documents confirming qualifications.

As a post-processing of the output data, the extracted skills represented by a single word were converted to dictionary form. Synonymous skills were combined together and their results were summarized. For example, words asiakaspalveluhenkinen, asiakaspalveluasenne, asiakaslähtöinen, asiakaspalveluhenkisyys and others are synonyms and were combined into asiakaspalvelulähtöinen (customer service oriented).

4.6.1 Cloud of Words

First of all, it was decided to build a cloud of words for a few professions as the example of possible service built on the extracted skills. A cloud of words is a combination of tags that together have the shape of a cloud. The size of each word in the cloud is different and usually depends on its importance and how often a word appears in the text. In our case, the most popular skills have a larger size, and the less popular ones have a smaller size, respectively. Siivooja (Cleaner) is a profession for which the most JDs were

published on Oikotie Työpaikat in the given period of time. The example of cloud of words for Cleaner is shown in Figure 26.

Figure 26. Cloud of words for the profession Siivooja (Cleaner).

As shown in Figure 26, the most popular skill is suomenkieli (finnish language), as well as reipas (brisk), itsenäinen (independent) etc. Besides, according to the results, it is important for Cleaner to have a driving license and an own car. The less popular skills are luotettava (reliable), oma-aloitteisuus (self-initiative), fyysinen (physical) and laitoshuoltaja (plant maintainer).

For the better clarity, the distribution of skills by popularity for the Cleaner occupation is presented in Figure 27. The count represents the number of JDs where a skill appears. Since Cleaner is usually a profession that does not require high qualifications, almost all of described skills are soft skills. Only such skills as työturvallisuuskortti (Occupational Safety Card) and ajokortti (driving license) can be attributed to hard skills.

Results also demonstrate that the main skill is kokemus (work experience) for Cleaner and other professions. This skill is present in almost all JDs and will be excluded from statistics in the future to make it suitable for analysis and viewing.

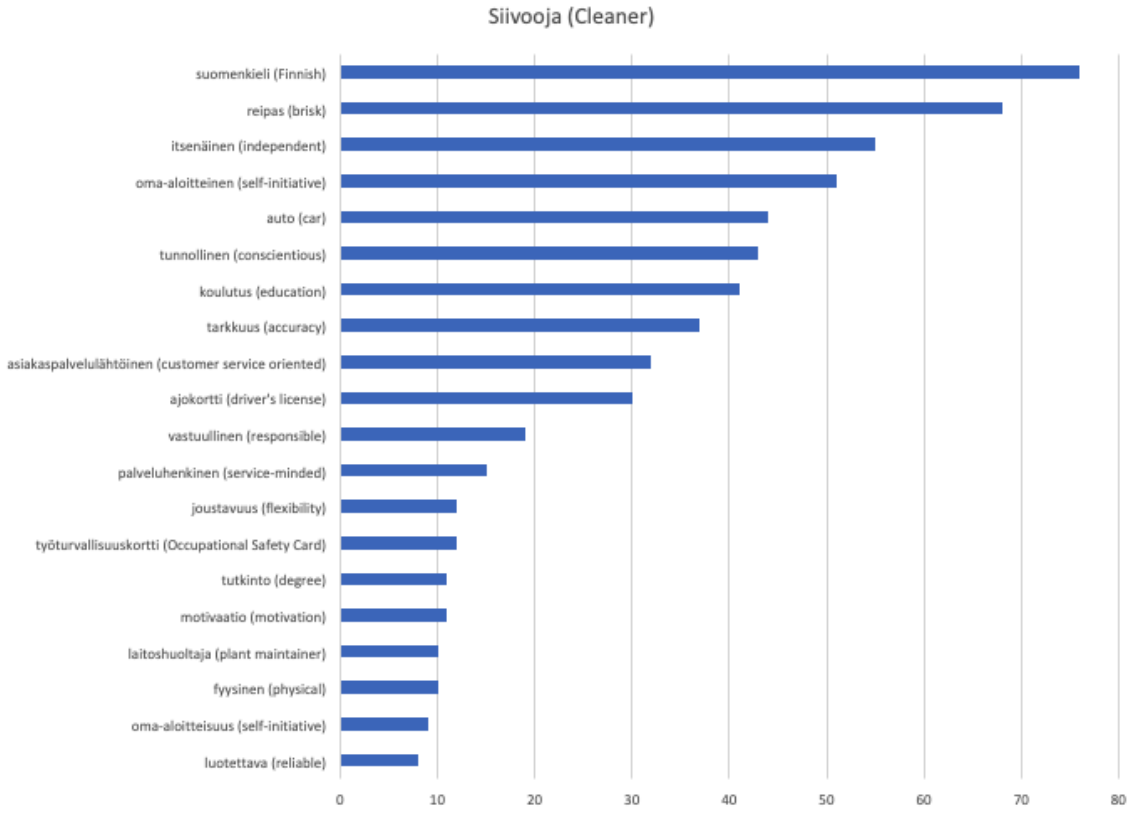

Figure 27. The most popular skills for the profession Siivooja (Cleaner).

Similar to the Cleaner profession, a cloud of words and the distribution of the most popular skills were built for Ohjelmistokehittäjä (Software Developer). These analytical results are presented in Figure 28 and Figure 29. Unlike the previous example, a Software Developer is a highly qualified profession that requires hard skills. In this case, skills include programming languages such as C, JavaScript, SQL, Java, CSS, HTML etc. Furthermore, it is important to know both suomenkieli (Finnish) and englantinkieli (English). According to the analysis, soft skills that are important for a Software Developer include itsenäinen (independent) and oma-aloitteinen (self-initiative).

A similar cloud of words can be easily drawn for any profession or job posting. This is a powerful tool that can merely improve user experience. For example, the user does not need to read a long job description, all the necessary skills will be visible in the cloud of words.

Figure 28. Cloud of words for the profession Ohjelmistokehittäjä (Software Developer).

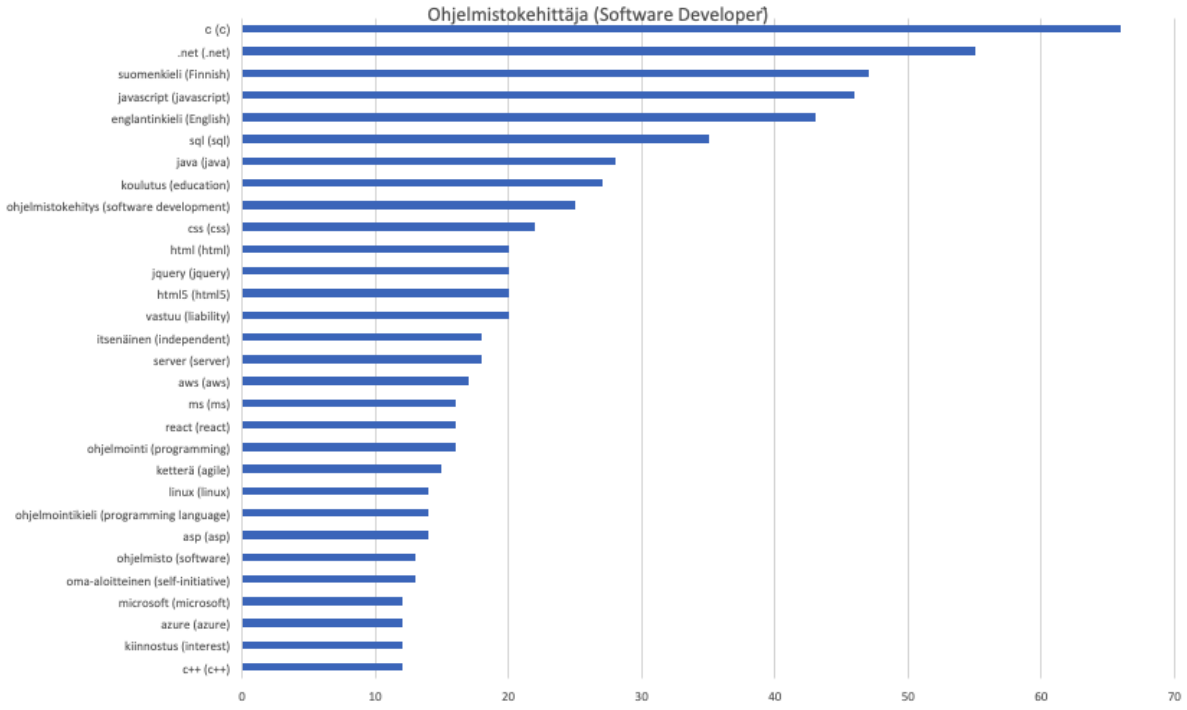

Figure 29. The most popular skills for the profession Ohjelmistokehittäjä (Software Developer).

4.6.2 Skill Demand

The next step was to evaluate skills demand for soft and hard skills separately. In addition, it was also decided that it is useful for Oikotie Työpaikat users to evaluate demand for the most commonly required languages, as well as education and certificates. These diagrams are presented at the end of this section.

Soft skill demand

Soft skills characterize non-specialized qualities of a person. However, these skills are quite important for a career and often play a key role for successful participation in the work process and high productivity. Since soft skills are not related to a specific professional field, this type of skills is the most popular and represented in almost every JD. The number of soft skills is quite high, as a consequence it was decided to consider their statistics separately. The diagram of twenty most popular soft skills is presented in Figure 30.

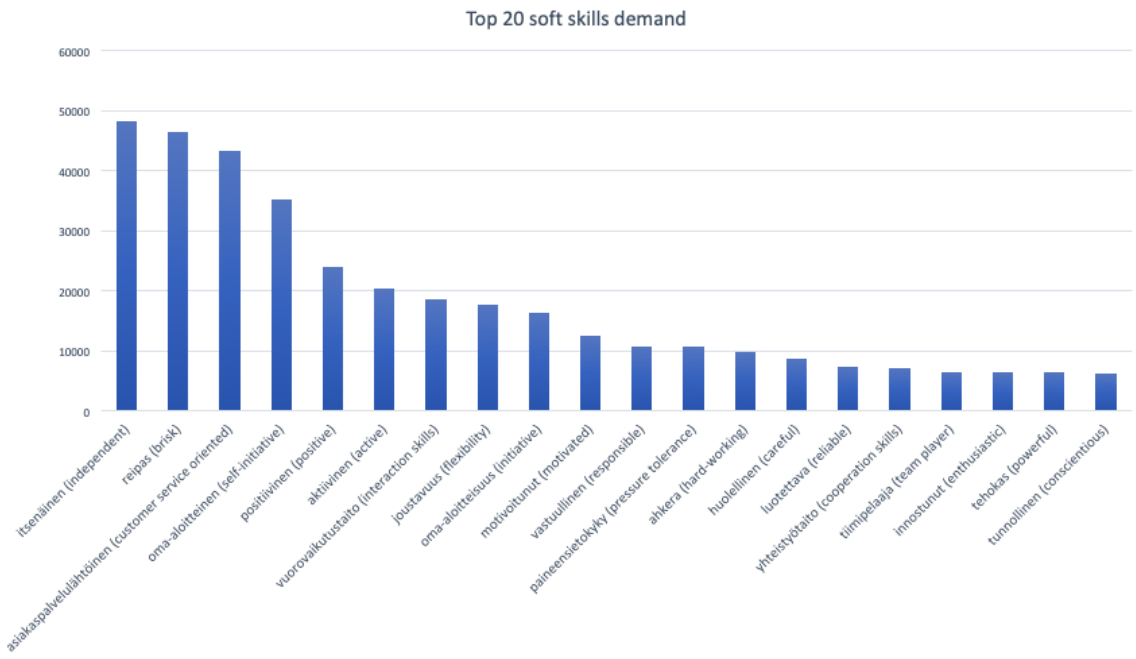

Figure 30. Top 20 the most popular soft skills.

As demonstrated in the figure above, the most demanded soft skills are itsenäinen (independent), reipas (brisk), asiakaspalvelulähtöinen (customer service oriented), oma-aloitteinen (self-initiative), positiivinen (positive) etc.

Hard-skill demand

Hard skills are professional competencies that need to be learned and can be measured. This type of skills usually characterizes a specific professional area. In the example of Software Developer profession, in the most case hard skills are programming languages. Therefore, it was decided to look at how the demand of the most popular programming languages changed over time. Figure 31 presents the diagram of common programming languages, where the x-axis is the number of JDs that mention this language, and the y-axis is the date (YYYY.MM) when job postings were published on the Oikotie web-site.

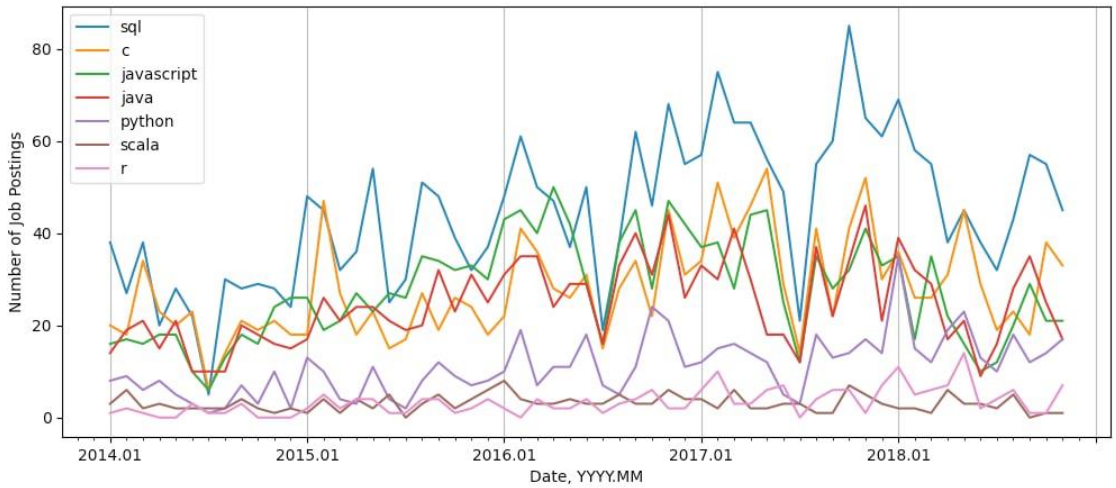

Figure 31. The most popular programming languages.

As demonstrated in the figure above, the local minimums of programming languages demands are in Julies for the considered time interval of more than four years. Apparently, this is due to the fact that midsummer is a vacation time in Finland and the demand for any labor force is falling. The maximum demand for programming languages is observed in the beginning and end of the year. According to the diagram, the most popular programming languages are SQL, C, Java and JavaScript. In general, there is an increase in demand for all programming languages.

Languages

Often enough one of the important requirements for candidates in JDs is knowledge of local and foreign languages. Language skills demand is presented in Figure 32. As shown in the pie chart below, English and Finnish occupy the first position with a

difference of only 6 %. The third position of the rating is held by the Swedish language and is 12 %. The remaining 2% is made up of other languages.

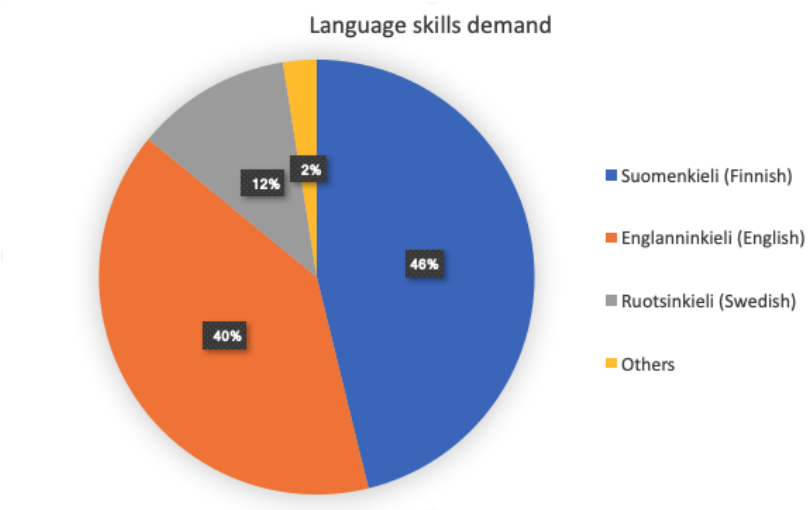

Figure 32. Language skills demand. Other languages include Venäjänkieli (Russian), Saksankieli (German), Vironkieli (Estonian), Ranskankieli (French), Espanjankieli (Spanish), Italiankieli (Italian).

According to the Figure 33, the language demand on the Finnish job market is growing during the considered time interval of over four years. However, the most increase of the demand is experienced by Finnish and English languages. The seasonal minimums and maximums are also well observed on the figure.

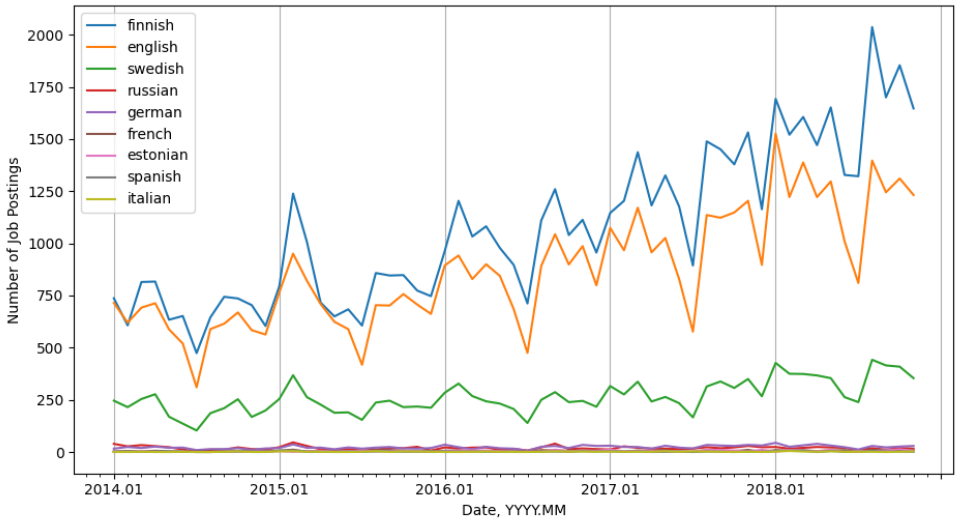

Figure 33. Language skills demand over time.

Education and certificates

Finally, the demand for education and certificates was constructed separately and presented in Figure 34.

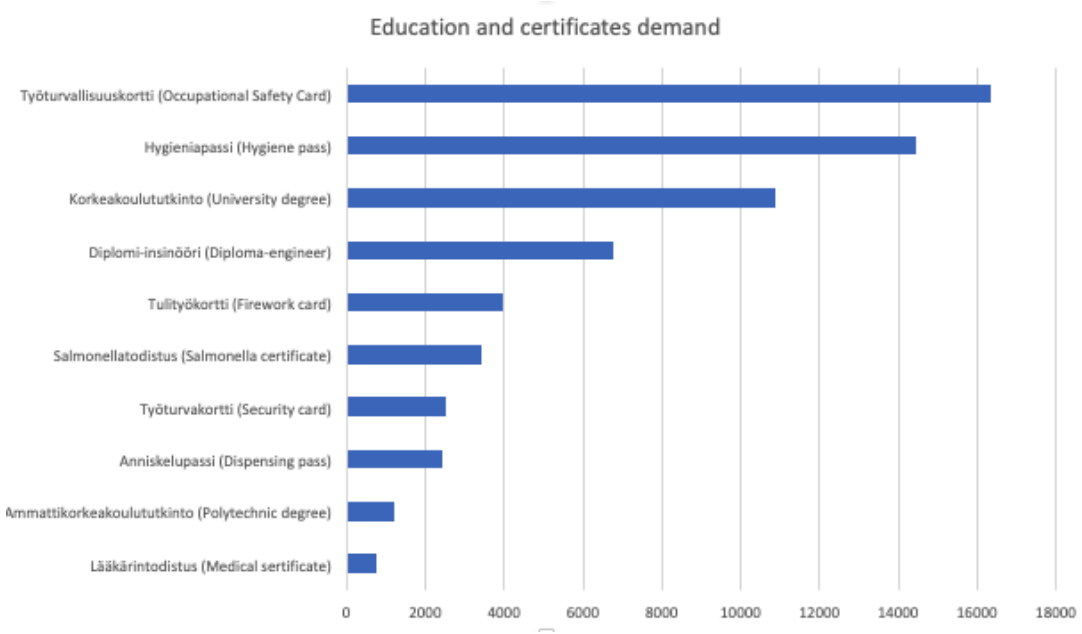

Figure 34. Education and certificates demand.

The most required certificates are Työturvallisuuskortti (Occupational Safety Card) and Hygieniapassi (Hygiene pass). In addition, Korkeakoulututkinto (University degree) is important for many recruiters.

These sections presented few examples of applying the developed skill extraction framework. The results exhibit that this framework has a chance to pave a fundament for a lot of useful applications and improve the Oikotie Työpaikat job market service.

5 Conclusion

The objective of this master thesis was to develop a skills extraction framework for job advertisements for job service Oikotie Työpaikat. Further, based on this framework Oikotie can easily build a recommendation system for better user experience, evaluate job market demand or create other useful services.

5.1 Summary

The first step of this master thesis was to explore a variety of currently used skills extraction systems from a free written text such as job descriptions. Apparently, the diversity of skills extraction systems is huge, namely, from diving into technologies based on a skills taxonomy to developing a machine learning model. According to the research carried out in Section 2, systems based on the skills taxonomy combined with the graph-based approach have obvious disadvantages. These include the time-consuming building of a skills library that requires the involvement of a team of experts or users, the difficulty in updating the skills dictionaries, and other obstacles. Modern machine learning technologies have clear advantages and allow to achieve impressive results. However, ML approaches mentioned in Section 2 do not utilize the state-of-the-art NLP algorithms and might be significantly improved.

The main goal of Section 3 was to study different NLP techniques that take into account the context of words. The investigation was started with general seq2seq models such as RNN and LSTM that demonstrated obvious shortcomings in the modeling of long and short-range dependencies. Moreover, the sequential nature of RNN and LSTM model architectures prohibit parallel processing of words, which is important for handling large corpuses of texts. A newer model called Transformer in combination with the Self-attention mechanism eliminates the disadvantages mentioned above. However, there is another state-of-the-art algorithm that surpasses the previous approaches and its name is BERT. Unlike directional models, BERT utilizes pre-trained Transformer encoders that allow it to read the entire text at once, rather than a sequence of words. The fine-tuning of BERT for a specific task usually involves adding just one layer to the core model. Due to this fact, the creation of a skills extraction system is reduced to configuring only one layer of the neural network, which requires a relatively small amount of training data. Since the extraction system must be able to process words in Finnish, it was decided to leverage Google's open-source BERT model for the Finnish language (FinBERT)

developed by Turku University. This version of BERT outperformed the previous multilingual model in a wide range of tasks, especially in classification problems. One of the classification subtasks is NER, which can be easily applied to extract entities such as skills from unstructured texts, and it is utilized in this study.

A detailed process of implementing the model for extracting skills was presented in Section 4. The dataset provided by Oikotie Työpaikat contains about 300 000 job advertisements. The raw data were cleaned and pre-processed as input for BERT model. Moreover, 100 JDs were randomly selected from the pre-processed data. This dataset was labeled utilizing the web-based tool for NLP text annotation called TagTog. In order to facilitate data labeling and eliminate inaccuracies, the definition of the skill concept was described in details in Section 4. However, during the annotation process, there were a huge amount of edge cases that required involving linguistic experts and further improvement of the skill concept. The architecture of the developed model contains the main block based on FinBERT and the additional layer was chosen as a simple Dense layer with a softmax activation function. This Dense layer was fine-tuned for the NER task. The whole model was trained on 85 labeled job descriptions, that were randomly split into training and validation datasets. According to the learning curves, the sufficient number of epochs was 7, otherwise, the model overfitting starts. Furthermore, for evaluating model performance the additional test dataset was deliberately postponed and contained 15 JDs, which was about 1541 words. For classification tasks such as NER, the main approach for model performance evaluation is a confusion matrix and its based different evaluation metrics as accuracy, precision, recall, and F1-score. These metrics were calculated firstly for skill phrases and then for every single word.

5.2 Model Performance and Evaluation

In the case of phrases, model evaluation metrics were not quite high. This is due to the fact that the dataset used in this thesis theoretically may contain labeling mistakes and ambiguities for above mentioned reasons. Furthermore, it was decided to do a model evaluation on the level of individual words rather than phrases. This is a legit idea because finding the edges of skill phrases is a quite cumbersome process and depends a lot on the qualification of a person preparing training and testing datasets. As a consequence, the model learned training data might not be able to fully capture deviated patterns in test data. In the case of single words, the model performance is higher than on the level of phrase. For instance, Precision grew by over 10% and reached 78%.

Other metrics showed higher values as well. The extraction of individual words makes a lot of sense since skill phrases explaining the same skill cannot be easily matched for further analysis. This happens because even a slight difference in the text or an extra descriptive word leads to the mismatch of skill phrases.

5.3 Results Highlights

The developed skill extraction framework achieves noticeable results. Extracted skill phrases include soft skills, hard skill and different qualification certificates. Although, a noise is present in the model output, there are ways for improvements. However, already the current model enables to make valuable inferences and draw descriptive statistics in the job market domain.

The skill extraction framework may become a fundamental tool for a number of potential services oriented for actual customers and businesses. Such services are limited by only one's imagination or creativity. For example, there could be a service to facilitate CV writing or pointing an applicant to missing skills for a better look on a job market. This feature may lead to a beneficial integration with universities or online learning platforms. Also, HR departments of big companies or governmental organizations might need a skill demand map over time to better plan their strategies in industry.

5.4 Future Work

First of all, it is planned to improve the architecture of the model by replacing a simple Dense layer with a more advanced one. For instance, the Conditional Random Field (CRF) layer is used for NER in the research proposed by Arkhipov et al [31]. In addition, the optimal regularization method allows a model to avoid overfitting and achieve higher performance.

Secondly, the developed model can be applied not only to extracting skills from JDs, but also from CVs. In this case, new training data should be prepared and labeled based on users' CVs. After training the model, it will be possible to normalize the obtained output data and build a recommendation system for users similar to the paper Gugnani, A [9].

References

1 Cedefop (2015). Skill shortages and gaps in European enterprises: striking a balance between vocational education and training and the labor market. Luxembourg: Publications Office. No 102.

2 Bastian, M.; Hayes, M.; Vaughan, W.; Shah, S.; Skomoroch, P.; Kim, H.; Uryasev, S.; and Lloyd, C. 2014. LinkedIn Skills: Large-Scale Topic Extraction and Inference. In Proceedings of the 8th ACM Conference on Recommender Systems, 1–8. New York: Association for Computing Machinery.

3 Kivimäki, I.; Panchenko, A.; Dessy, A.; Verdegem, D.; Francq, P.; Fairon, C.; Bersi- ni, H.; and Saerens, M. 2013. A Graph-Based Approach to Skill Extraction from Text. Paper presented at the Graph-Based Methods for Natural Language Processing workshop, 18 October, Seattle, WA.

4 ESCO: European Skills, Competences, Qualifications and Occupations https://ec.europa.eu/esco/ . Accessed 17 Dec 2015. Madely du Preez

5 Madely du Preez. Taxonomies, folksonomies, ontologies: what are they and how do they support information retrieval? The Indexer The International Journal of Indexing 33. March 2015.

6 Wang, Z.; Li, S.; Shi, H.; and Zhou, G. 2014. Skill Inference with Personal and Skill Connections. In Proceedings of the 25th International Conference on Computational Linguistics (COLING), 520–529. Stroudsberg, PA: Association for Computational Linguistics.

7 Phuong, H., Mahoney, T., Javed, F., McNair, M., 2018. Large-Scale Occupational Skills Normalization for Online Recruitment. AI MAGAZINE. Association for the Advancement of Artificial Intelligence. ISSN 0738-4602.

8 Sharma, N. Job Skills extraction with LSTM and Word Embeddings. University of Technology Sydney - UTS, Sydney, Australia. Sept 2019.

9 Gugnani, A. Implicit Skills Extraction Using Document Embedding and Its Use in Job Recommendation. Conference: AAAI - Innovative Applications of Artificial Intelligence (IAAI). February 2020.

10 Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems, 3111–3119.

11 Gugnani, A.; Kasireddy, V. K. R.; and Ponnalagu, K. 2018. Generating unified candidate skill graph for career path recommen- dation. In 2018 IEEE International Conference on Data Mining Workshops (ICDMW), 328–333. IEEE.

12 MediaWiki API help. https://en.wikipedia.org/w/api.php Accessed 15 Oct 2020.

13 Watson Natural Language Understanding. https://www.ibm.com/cloud/watson-natural-language-understanding. Accessed 15 Oct 2020.

14 Alex Graves. Sequence Transduction with Recurrent Neural Networks. Department of Computer Science, University of Toronto, Canada. 14 Nov 2012. https://arxiv.org/abs/1211.3711. Accessed 22 Oct 2020.

15 Guliano Giacaglia. How Transformers Work. The Neural Network used by Open AI and DeepMind. 11 March 2019. https://towardsdatascience.com/transformers-141e32e69591. Accessed 22 Oct 2020.

16 Bengio Y., Simard P., Frasconi P., Learning Long-Term Dependencies with Gradient Descent is Difficult. 2 March 1994. http://ai.dinfo.unifi.it/paolo//ps/tnn-94-gradient.pdf. Accessed 23 Oct 2020.

17 Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. 12 Jun 2017. https://arxiv.org/abs/1706.03762. Accessed 23 Oct 2020.

18 Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Google AI Language. 24 May 2019. https://arxiv.org/pdf/1810.04805.pdf. Accessed 28 Oct 2020.

19 Jay Alammar, The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning) https://jalammar.github.io/illustrated-bert/ . Accessed 29 Oct 2020.

20 Y. Wu, M. Schuster, Z. Chen, Q. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Ł. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes and J. Dean Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation, 2016.

21 Devlin, J. und Chang, M.-W. (2018) Google AI Blog: Open Sourcing BERT: State-of-the-Art Pre-training for Natural Language Processing, Google AI Blog. Available here: https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html. Accessed: 2 Nov 2020.

22 Jacob Devlin, Ming-Wei Chang, Open Sourcing BERT: State-of-the-Art Pre-training for Natural Language Processing, Google AI Language, 2 Nov 2018, https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html. Accessed: 3 Nov 2020.

23 Dario Radecic, Softmax Activation Function Explained, 18 Jun 2020. https://towardsdatascience.com/softmax-activation-function-explained-a7e1bc3ad60. Accessed: 3 Nov 2020.

24 Tagtog application for labeling entities in the text. https://www.tagtog.net/ Accessed: 4 Nov 2020.

25 Antti Virtanen, Jenna Kanerva, Rami Ilo, Jouni Luoma, Juhani Luotolahti, Tapio Salakoski, Filip Ginter, Sampo Pyysalo. Multilingual is not enough: BERT for Finnish. Turku NLP group, University of Turku. 15 Dec 2019. https://arxiv.org/abs/1912.07076. Accessed: 6 Nov 2020.

26 Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. https://arXiv preprint arXiv:1810.04805. Accessed: 9 Nov 2020

27 Ruokolainen, T., Kauppinen, P., Silfverberg, M., and Lindén, K. (2019). A Finnish news corpus for named entity recognition. Language Resources and Evaluation, pages 1–26.

28 Meizhi Ju, Makoto Miwa, and Sophia Ananiadou. A neural layered model fornested named entity recognition. In Proceedings of The Sixteenth Annual Conference of the North American Chapter of the Association for ComputationalLinguistics: Human Language Technologies (NAACL HLT 2018), pages 1446–1459, 2018 https://www.researchgate.net/publication/334847516_A_Finnish_news_corpus_for_named_entity_recognition. Accessed: 9 Nov 2020.

29 Michel JoseAnzanello, Flavio SansonFogliatto. Learning curve models and applications: Literature review and research directions. International Journal of Industrial Ergonomics Volume 41, Issue 5, September 2011, Pages 573-583. https://www.sciencedirect.com/science/article/abs/pii/S016981411100062X. Accessed: 13 Nov 2020.

30 Sarang Narkhede, Understanding Confusion Matrix. May 2018, Towards Data Science. https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62. Accessed: 14 Nov 2020.

31 Mikhail Arkhipov, Maria Trofimova, Yuri Kuratov, Alexey Sorokin. Tuning Multilingual Transformers for Named Entity Recognition on Slavic Languages. *Neural Networks and Deep Learning Laboratory, Moscow Institute of Physics and Technology. Faculty of Mathematics and Mechanics, Moscow State University. https://pdfs.semanticscholar.org/701b/49596ab9471fa2be9c58ea625dd7b7471b15.pdf. Accessed: 18 Nov 2020.