UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Meelis Perli

Representation Learning on Free Text Medical Data

Master's thesis (30 ECTS)

Supervisors: Raivo Kolde, PhD
Sven Laur, PhD

Tartu 2021

Representation Learning on Free Text Medical Data

Abstract:

Over 99% of the clinical records in Estonia are digitized. This is a great resource for clinical research, however, much of this data cannot be easily used, because a lot of information is in the free text format. In recent years deep learning models have revolutionized the natural language processing field, enabling faster and more accurate ways to perform various tasks including named entity recognition and text classifications. To facilitate the use of such methods on Estonian medical records, this thesis explores the methods for pre-training the BERT models on the notes from "Digilugu". Three BERT models were pre-trained on these notes. Two of the models were pre-trained from scratch. One on only the clinical notes, the other also used the texts from the Estonian National Corpus 2017. The third model is an optimized version of the EstBERT, which is a previously pre-trained model. To show the utility of such models and compare the performance, all four models were fine-tuned and evaluated on three classification and one named entity recognition downstream tasks. The best performance was achieved with the model trained only on notes. The transfer learning approach used to optimize the EstBERT model on the clinical notes improved the pre-training speed and performance, but still had slightly worse performance than the best model pre-trained in this thesis.

Keywords:

Artificial intelligence, transfer-learning, natural language processing, deep learning

CERCS: P176, Artificial intelligence

Esindusõpe Vabatekstilistel Meditsiinilistel Andmetel

Lühikokkuvõte:

Eestis üle 99% kliinilistest andmetest on digiteeritud. See on suurepärane ressurss kliiniliste uuringute jaoks, kuid neid andmeid ei ole lihtne kasutada, sest suur osa andmetest on vaba teksti kujul. Viimastel aastatel on süvaõppe mudelid revolutsioneerinud loomuliku keele töötlemise valdkonna. Uued mudelid on kiiremad ja saavad parimaid tulemusi erinevates ülesannetes, sealhulgas nimetatud üksuste tuvastamises ning teksti klassifitseerimises. Eesti meditsiiniandmete kasutamise hõlbsustamiseks, uuritakse selles lõputöös meetodeid Digiloost pärinevate kliiniliste kokkuvõtete peal BERT mudelite eel-treenimiseks. Kolm mudelit eel-treeniti nullist. Esimene ainult kliiniliste kokkuvõtete peal. Teise eel-treenimiseks kasutati lisaks kokkuvõtetele ka tekste Eesti rahvuskorpusest 2017. Kolmas mudel on optimiseeritud versioon EstBERT mudelist, mis on varasemalt eeltreenitud mudel. Nende mudelite kasulikkuse näitamiseks ja võrdlemiseks peenhäälestati ja hinnati kõiki neli mudelit kolmel teksti klassifitseerimise ja ühel nimega üksuse tuvastamise allavoolu ülesannetel. Parima tulemuse saavutas mudel, mida treeniti ainult kliinilistel kokkuvõtetel. Esindusõppe meetod, millega EstBERTi mudelit optimeeriti kliiniliste kokkuvõtete peal, kiirendas eel-treenimise kiirust ja parandas tulemusi, kuid ikkagi jäi alla parimale mudelile, mis siin lõputöös treeniti.

Võtmesõnad:

Tehisintellekt, siirdeõpe, loomuliku keele töötlus, süvaõpe

CERCS: P176, Tehisintellekt

Contents

1. Introduction

In Estonia, 99% of the health data is digitized. Every patient has their own records, which they can access via the Electronic Health Record (EHR)[1] system. For example, it contains their time-critical data, blood type, allergies, dental information, previous treatments, etc. This system allows the healthcare workers to easily access patient's records, even read test results as they are entered. The system also compiles data for statistics and machine learning (ML) [1].

The data in the EHR system falls into two major categories: structured and unstructured. Structured data is organized and machine readable, making it easier to be used by machine learning (ML) techniques. For example, a person's blood type, age, gender is structured.

However, there are a lot of clinical notes in the EHR system which are unstructured. Making them more difficult to use. Clinical notes can contain all sorts of information about the patient and with the right methods, the notes can be used separately or even used to enrich structured data [2], [3].

The main goal of this thesis is to train a Bidirectional Encoder Representations from Transformers (BERT) model on these notes. The model is trained in two phases. First the model learns the general language of the notes in the process called pre-training. After that, the model can be fine-tuned to accomplish a more specific downstream task. Such as named entity recognition (NER) and text classification. Thus, making it possible to use these clinical notes more easily. There were multiple reasons to use BERT model in particular.

First, this model was the state-of-the-art natural language model at the time of writing this thesis, beating other models like ELMO in performance and training speed [4]–[8].

Secondly, once it has been pre-trained, it can be fine-tuned relatively cheaply (compared to pre-training) to accomplish a more specific task [9]. For example,[2] it can be used for named entity recognition (NER) [5], readmission prediction [4], text classification, extractive question answering and text summation. NER and text classification are used in this thesis to validate the models.

[1] https://www.digilugu.ee/login
[2] https://huggingface.co/transformers/task_summary.html

Thirdly the model does not use recurrent layers, but instead it only relies on multi-head self-attention mechanisms, which also allow parallelization when training. Making the training faster than other commonly used methods, that use recurrent or LSTM layers [10].

Thus, a BERT model that is pre-trained on Estonian medical notes from “Digilugu” is the main result of this thesis. As the pre-training is expensive, the secondary goal was to explore various ways to perform transfer learning, by utilizing existing Estonian language models in the pre-training process. A BERT model pre-trained on Estonian Corpus called EstBERT has been recently published [6]. Here, several options for utilizing EstBERT, with and without medical not data, were implemented and evaluated.

This thesis begins by giving an overview of what has been done with BERT models before and the goals of this thesis are formulated as hypotheses (chapter 2). Then the BERT model is explained and how to pre-train and fine-tune it (chapter 3). After that the data and the taggers used to clean the notes are described (chapter 4). Next the methods used to pre-train and fine-tune the models are described (chapter 5). After that, the results that the fine-tuned models achieved, are presented (chapter 6). Finally, the future work and potential improvements are discussed (chapter 7) and the conclusions are laid out (chapter 8).

2. Related Work

There are many ways to represent words. For example, Word2Vec uses a neural network model to learn a high dimensional vector representation of words such that similar words tend to be close [12]. FastText associates a vector representation to each character *n*-gram and then sums these vectors to represent words [13]. ELMo models use a bidirectional language model to assign words a deep contextualized representation [14].

BERT models revolutionized the way the words are represented in NLP. Quite a few BERT models have already been trained on clinical texts. Here is an overview of some of the models and how the ones in this thesis will differ.

Alsentzar et al. trained a publicly available BERT model on clinical notes. They only used clinical notes from an intensive care unit of a single institution (Medical Information Mart for Intensive Care III (MIMIC-III) dataset) [7], [15].

Gu et al. put it to a test whether it is better to only use domain specific texts or to mix the domains. They found that it is considerably better to train a model from scratch on only the domain specific texts. But they used the same vocabulary sizes (30K) for their BERT models, which might be an issue due the possibility of out-of-vocabulary words [8].

Kim and Lee trained BERT for clinical entity recognition from diagnosis texts. They showed that the BERT model is better than the other state-of-the-art models for clinical entity recognition. Using a question set they also attempted domain adaptation, such that the answer set was the source domain and question set was the target domain. They found it to be worse than the standard learning [5].

Huang et al. trained a ClinicalBERT model to predict patient readmission. ClinicalBERT is a BERT model that is both pre-trained and fine-tuned on the MIMIC III. They showed that ClinicalBERT outperforms a BERT model that is pre-trained on standard language corpora but fine-tuned on clinical notes [4].

The EstBERT model was trained by Tanvir et al. It is a BERT model, that has been pre-trained on the Estonian National Corpus 2017 [11]. The EstBERT model was the only Estonian language specific BERT model at the time of making this thesis [6].

In this thesis, three models were trained on clinical notes from health care institutions across Estonia. The first one was trained purely on the notes and from scratch. The second model

was also trained on the Estonian National Corpus 2017. The third model is an optimized version of the EstBERT model. These models were created to test the two hypotheses:

1) Increasing training set size by adding text from other domains improves the performance of the resulting models.
2) The existing EstBERT model already captures enough knowledge about language structure that slight further optimization with medical texts is sufficient.

Gu et al. already tested if it is better to use domain specific texts or to mix the domains, but here the vocabulary size was adjusted to make room for words from both texts [8]. The idea is that BERT models can only use sequences with up to a certain length. So, when there are a lot of out-of-vocabulary words, the sentences must be truncated, which reduces the amount of information and hinders the performance.

The second hypothesis comes from the search of a better way to pre-train the BERT models. The idea is to test if an already pre-trained BERT model can be efficiently adapted to perform at least on par with the models that were trained from scratch.

3. Technical Background

BERT or a Bidirectional Encoder Representations from Transformers is a powerful neural language model. The model's core is called the Transformer (see Figure 1).

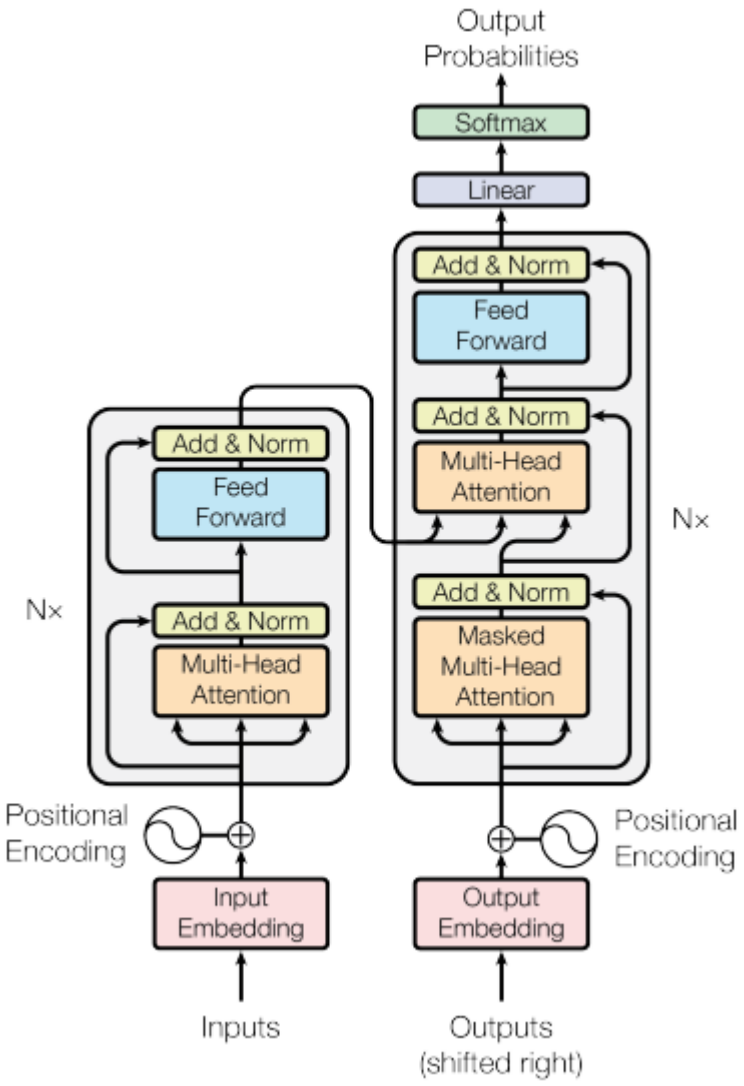

Figure 1. The Transformer - model architecture [17].

The Transformer itself consists of encoder and decoder stacks. Both stacks have the same number of identical layers of:

- The encoder consists of a multi-head self-attention mechanism and a position-wise fully connected feed-forward network.
- The decoder also has the components that the encoder has, but also has a multi-head self-attention mechanism for the output of the encoder stack.

Attention mechanisms conceptually works like a person's cognitive ability to focus some things and ignore others [16]. For example, in a translation task, the attention mechanism helps to map the input words to the output words (see Figure 2). Here both the inputs and outputs are used to learn which words should be attended to.

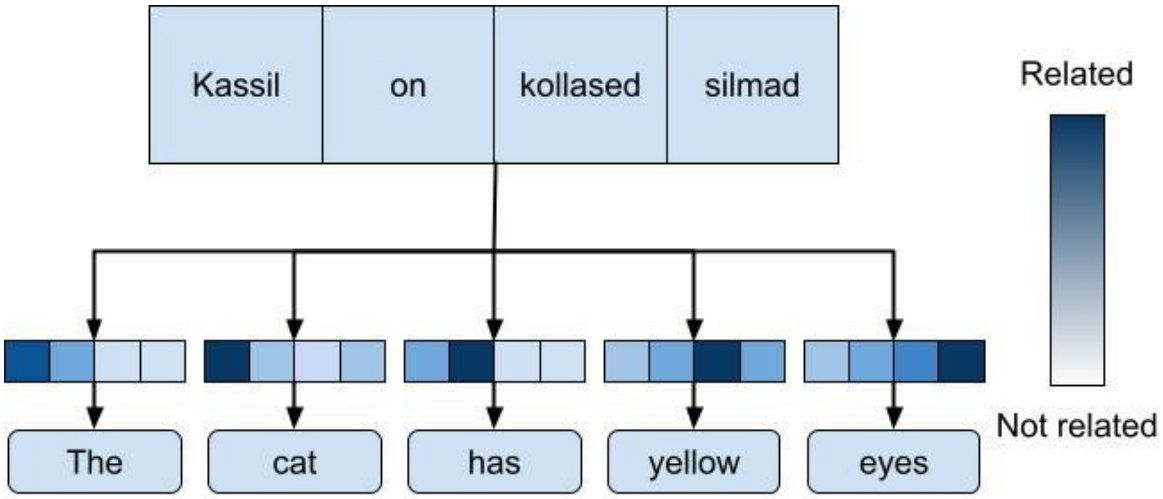

Figure 2. Attention mechanism in a translation task. The input words are multiplied by the corresponding weights of the outputs to focus more on the related words.

However, the Transformer uses self-attention. The difference is that instead of describing how related each word in the input sequence is to each word in the output sequence. The idea of a self-attention mechanism is to find how related each word in the input sequence is to every other word in the input sequence. Since the model works on continuous vectors instead of words. It could be rephrased that each self-attention mechanism learns how related the input vectors are to each other. The "Multi head" means that the process of training the self-attention mechanisms can be done in parallel [10].

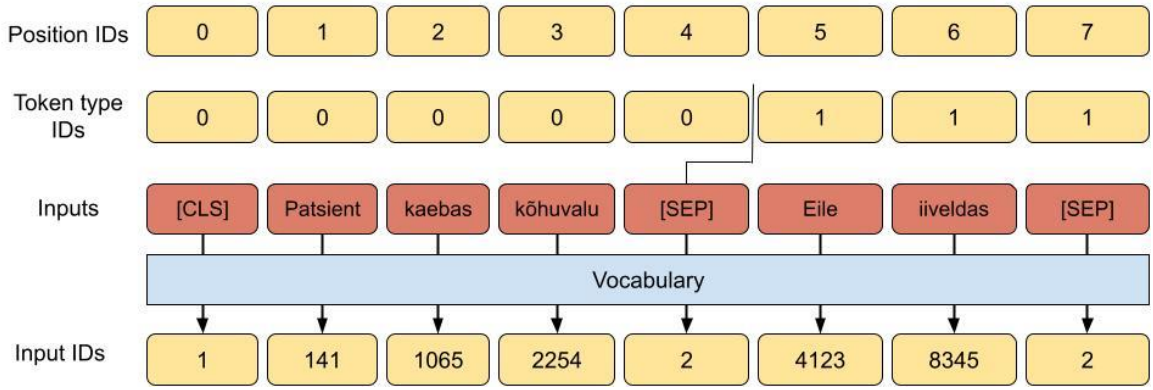

Figure 3. The BERT model inputs.

While other prominent models like ELMo use left-to-right or right-to-left methods to learn the token representations [14]. BERT can learn the token representations bidirectionally [9].

It means that the model processes the entire sequence of tokens at the same time and learns the representations that way instead of going through the tokens n-gram at a time [14].

The BERT model requires for the sentences to be turned into sequences of input ids, token type ids and position ids (see Figure 3). The input ids can be created by using a Vocabulary. The vocabulary is basically a hashmap where the keys are the tokens and row numbers are the input ids. The token type ids can be assigned by using the *[SEP]* tokens. The position ids are just the positions of the tokens. All these ids are mapped to their corresponding embeddings in the model and for each token the embeddings are summed together [9].

BERT model itself is quite large. The models trained in this thesis are identical to $\text{BERT}_{\text{base}}$ that has 12 layers, 764 hidden units and 110M trainable parameters. The next sections cover how to train a BERT model and what kind of inputs and outputs are required for the pre-training tasks.

3.1 BERT Pre-training

BERT models are usually pre-trained on 2 different unsupervised tasks. Masked language modeling (MLM) and next sentence prediction (NSP) [9]. Both tasks require additional parameters. However, the two following parameters are required for all tasks:

- *input_id*: Numerical representations of tokens, that are used to link the tokens with their corresponding embeddings.
- *attention_mask*: Shows which tokens should be attended to.

The objective in MLM is to fill in the blanks. Usually, 15% of the tokens in a sentence are masked. Of these masked tokens, 80% are replaced with the mask token *[MASK]*. 10% original is kept and 10% of the time replaced with a random token. To be able to check if the model predicted correctly, only one additional parameter is required:

- *labels*: The correct input_ids.

In NSP, the objective is to predict if 2 sentences are consecutive. In training, 2 sentences are given to the model and 50% of the time, the sentences are consecutive and in the other times a random sentence is selected as the second sentence. Here two additional parameters are required:

- *token_type_ids:* Shows which tokens belong to the first or the second sentence.
- *next_sentence_label*: Indicates if the sentences are consecutive or not.

This is all that is required to pre-train BERT using these two tasks. Note that during pre-training, all the model's parameters are modified. Once the model has been pre-trained, it can be fine-tuned. The next section describes what it is and how it is done.

3.2 Fine Tuning

Fine-tuning is the adaption of a pre-trained model to solve a new task. This is achieved by first changing the head of the model. The new head is added on top of the pooling output layer of the model. For example, in case of the *SequenceClassification* head, a linear (dense) layer is added.

The model heads used for MLM and NSP are not suitable for other tasks, thus a suitable head must be attached. The new head is initialized randomly because it has not been trained yet. Secondly the model with the new head requires further training on another dataset. Usually, labelled dataset since most of the tasks that the model is used for are supervised.

During the fine-tuning process, all the model's parameters are tuned. The process itself requires much less time and data than pretraining.

The Transformers[3] Python library provides 6 different model heads for fine-tuning:

1) MaskedLM – Used for the MLM task.
2) NextSentencePrediction – Can be used to predict if the two sentences are consecutive.
3) SequenceClassification – Mostly used for sequence classification but can also be used for regression tasks if the number of classes is set to 1.
4) MultipleChoice – Instead of classifying, this model head is designed to choose the answer from varying options based on a sequence.
5) TokenClassification – Similar to sequence classification, but instead of classifying the sequences, each individual token is classified.
6) QuestionAnswering – It can be used to answer extractive questions based on a text. For example, in case of text "That apple tree is 3 years old." A question like "How old is the apple tree?" can be asked.

These tasks are not all that the model can do. For example, the model can also be used to generate[4] and summarize[5] text.

[3] https://huggingface.co/transformers/
[4] https://colab.research.google.com/github/huggingface/blog/blob/master/notebooks/02_how_to_generate.ipynb
[5] https://github.com/huggingface/notebooks/blob/master/examples/summarization.ipynb

4. Data

Two datasets were used to train the BERT models. The epicrises dataset from the e-Health Records[6] and Estonian National Corpus 2017[7] [11]. The epicrises contain both structured and unstructured data about the patient and their visit. The unstructured data, that is used in this thesis is called clinical notes. Structured data was also used. This kind of data includes diagnose codes, the healthcare worker's profession, when the patient visited and such. This kind of info can be used to fine-tune and test the models.

The epicrises are confidential and thus cannot be shared. The link to the unprocessed Estonian National Corpus 2017 can be found at the bottom of this page. Table 1 gives an overview of the statistics of the used datasets.

Table 1. Statistics of the datasets.

	Sentences	**Words**	**Punctuation and numbers**
Sentences before cleaning	94.6M	609M	463M
Sentences after cleaning	91.1M	585M	330M
Cleaned Estonian National Corpus 2017	75.7M	945M	208M

Tanvir et al. already pre-processed the Estonian National Corpus 2017 to train the EstBERT model [6]. The same pre-processed corpus was given by them to be used in this thesis too.

The next sections first give an idea what the clinical notes are and how these looks like. After that, the parts that need to be modified or removed are discussed. Finally, the mechanisms that are used to detect such parts are described.

4.1 Clinical Notes

Whenever a patient visits a doctor in Estonia, a new clinical note is made or a previous one is updated [1]. Clinical notes can contain all sorts of information about the patient. Like what worries they have complained about, analysis results, what the doctor assigned to them etc.

[6] https://e-estonia.com/solutions/healthcare/e-health-record/
[7] Estonian National Corpus 2017

All this information can be useful, but the notes can be quite difficult to work with, because the notes are unstructured.

The notes can also vary a lot. Some only have a few words in them, not even a header. Others have multiple headers, tables, long lists and anonymised parts. An example of a made-up clinical note can be seen in Figure 4. All the segments encased by a red rectangle should be modified or removed for various reasons.

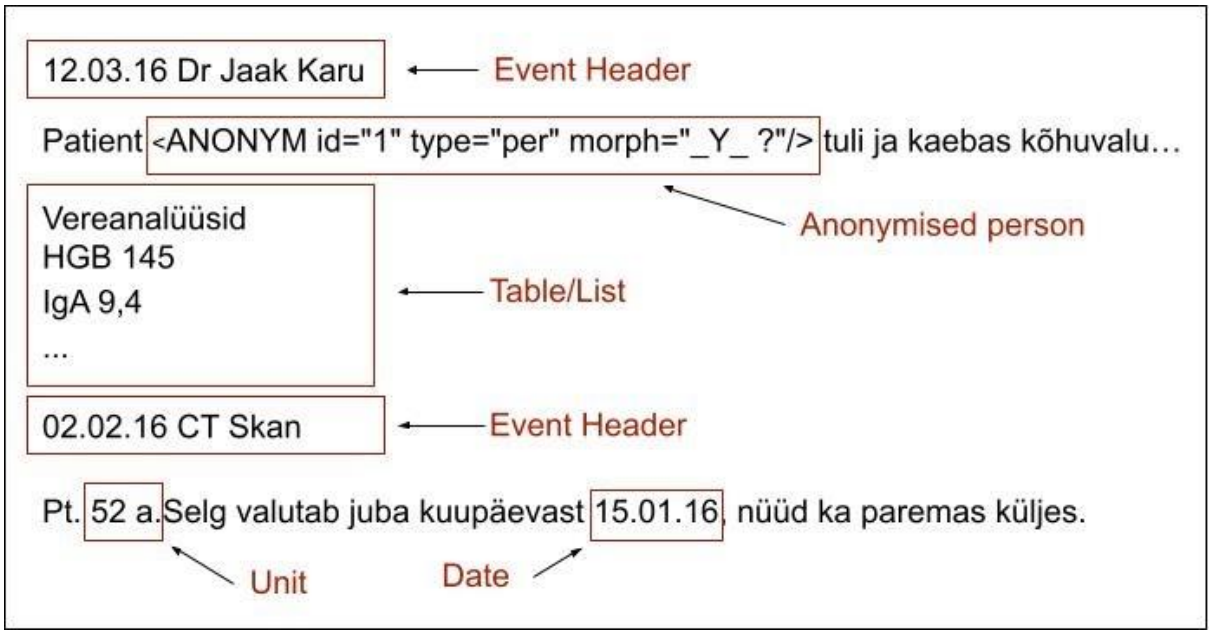

Figure 4. An example of a clinical note

The event headers in most cases are just the date when the note was created. Sometimes it is accompanied by the doctor's name. The event headers should be removed, because these are not really a part of a sentences and do not contain useful information about the patient.

All the notes are anonymized before they can be used for research. This results in the anonymization tags like *<ANONYM id="1" type="per" morph="_Y_?"/>* (see Figure 4). Tags like this one should be replaced because they contain morphological information.

Some of the notes contain tables or lists. These tables were just removed because these have a different structure than the sentences.

There are a lot of different units in the notes. Although it might be difficult for the BERT model to make sense of the numbers [17]. They were kept the same because there are so many of them and removing them would just leave a lot of holes. Also, the units can be useful.

Dates however should be modified because it might be difficult for the model to learn these too [17].

All these segments can be detected with taggers. The next section explains what the taggers are and what kind of taggers were used in this thesis.

4.2 Taggers

Taggers are mechanisms that classify tokens in a sequence of text. There are multiple ways to implement taggers. In rule-based cases, it might be enough just to create a tagger based on regular expressions. Another option is to create a list of relevant tokens or phrases and just tag these. With finite grammar, it is possible to create even more complex taggers [18]. There are other ways to create taggers, but the taggers used to tag the text segments in Figure 4 were made by using these methods and thus the explanation is limited to only these. The next subsections describe each tagger and how they were used in this thesis.

Event Header Tagger

The event header tagger was not made nor modified by the author of this thesis. It was made by a company called STACC. The tagger uses specified grammar to find the headers. This tagger was used to remove the event headers from the texts.

Anonymised Tagger

This tagger was also made by STACC but was modified in this thesis to also extract the morphological information. The tagger uses regular expressions to find the tags. Once found, pattern matching is used to extract the id, type, and the morphological information.

Table 2. POS tags and how they were replaced. * Only the nouns and verbs were replaced with a real word and put into the correct form, because these only convey the morphological information.

POS tag	Description	Replacement
S	Noun	tema*
V	Verb	tegema*
A	Adjective	ADJ
H	Real name	NAME
D	Adverb	ADV
I	Interjection	INJ
Y	Abbreviation	XXX

The id, in the tag, is just a unique identifier. These are unique per note, so it only helps to identify which tag you are dealing with. The type shows the reason why the word was anonymized. Finally, the morphological information about the original word is preserved as a POS tag.

In this thesis these tags were replaced. The replacements are shown in Table 2. In the case of nouns and verbs, the word forms are also added.

Table Tagger

The table tagger was also made by STACC. But it was modified such that it would not require Luigi[8] pipeline to work and could be directly applied on the notes. This tagger consists of multiple smaller taggers that use regular expressions to different find tables. Once a text has been tagged with those, another tagger is used to filter and combine the results. The goal of this tagger is to find table-like structures or lists from the notes.

The tables were removed once they were found. Because they have a different structure than the sentences.

Unit Tagger

The unit tagger was also made by STACC, but was also modified, because it did not seem to tag anything before. The tagger uses finite grammar, which was modified to get it to work. The modified tagger can tag only the following units: weight, height, blood pressure and tag some other units as ‘unit’.

The tagger was not used for cleaning the texts. However, the unit tagger also allowed to semi-automatically tag the units. This was helpful when creating the dataset for the NER task.

Date Tagger

The Date Tagger originates from the EstNLTK[9] Python library. This tagger also uses finite grammar, because it is not possible to distinguish numbers from dates without any context [18]. In this thesis, the found dates are replaced with the token ‘DATE’, because removing the date would just leave a hole in the text.

[8] https://github.com/spotify/luigi
[9] https://github.com/estnltk/estnltk

5. Methods

This chapter describes how the results of this thesis were achieved. The Chapter begins by describing how the clinical notes were processed. In section 5.2 it is described how the three models: PureBERT, MixedBERT and RetrainedBERT were trained. In section 5.3 the three models and an already pretrained model made by Tanvir et al called EstBERT are fine-tuned on three text classification tasks [6]. Finally, in section 5.4 all four models are fine-tuned on a token tagging task.

5.1 Note processing

The process of how to turn the notes into the sequences used for pretraining is illustrated in Figure 5. The process is divided into 3 steps.

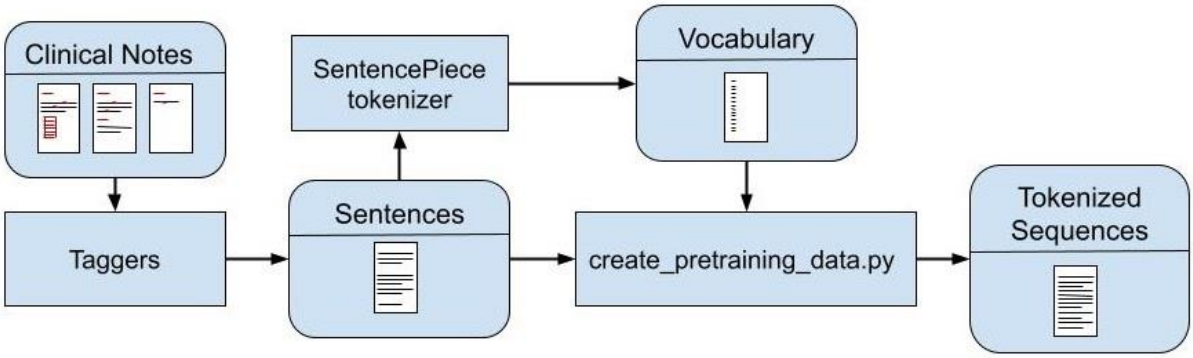

Figure 5. Clinical note processing pipeline.

First the notes are cleaned, and the sentences are extracted. Sentences should be extracted from the notes for multiple reasons. First, the next sentence prediction (NSP) task in pre-training requires consecutive sentences. Secondly, individual sentences are shorter than the entire note. The notes might get truncated in the tokenization process if the model is configured to use shorter sequences. Thirdly, the sentences have a more straightforward structure than a list or a table, which should make it easier for the model to learn the language. The next subchapter covers these more thoroughly.

5.1.1 Note Cleaning

Before the sentences can be extracted, the notes had to be cleaned. The first step was removing the table segments. At least the ones that the tagger found were removed. This was done first so that the other taggers would not break the tables. After that, the event header tagger was used to remove the event headers and split the notes into smaller sections. The event headers sometimes contain a date, so these were removed before using the date

tagger. The other taggers did not need to be used in a particular order. Finally, the date and the anonymised taggers were used to replace text sequences as described in section 4.2.

Sometimes symbols like ".", "-", ":", "|" were still at the beginning of the subsection. These were removed because a sentence should not start with these symbols.

Once the notes had been cleaned, they could be split into sentences. SentenceTokenizer from the EstNLTK library was used to do that. Each sentence was then saved on a separate line. An empty line was added to mark the end of a note. This helps to distinguish which sentences are consecutive, and which are not. This is required by the NSP task.

For example, from the note in Figure 4, two sentences should have been extracted:

1. *"Patient NAME tuli ja kaebas kõhuvalu."*
2. *" Pt. 52 a. Selg valutab juba kuupäevast DATE, nüüd ka paremas küljes."*

The sentences can now be used to create a vocabulary.

5.1.2 Vocabulary and Tokenization

The SentencePiece[10] tokenizer in unigram mode was used to create the vocabularies in this thesis. In the unigram mode the tokenizer splits the less frequent words into sub-word units to accommodate for the out-of-vocabulary words. More frequent words are added intact. Five special tokens were also added to the beginning of the vocabulary:

- [PAD] - Used for padding the too short sentences.
- [UNK] - While there are a lot of words and sub-word units in the vocabulary, it still does not cover all the words, so this token is used instead.
- [CLS] - It means classification. It is put at the beginning of the sequences if the model is used for classification.
- [SEP] - It is a separation token. It is used to separate sentences in tasks, where multiple sentences are used.
- [MASK] - Marks that the token has been masked. It is used in the masked language prediction task.

Once the vocabulary has been constructed, the sentences can be prepared to be used in pre-training. The final step was to use a method called *create_pretraining_data.py*[11] from the

[10] https://github.com/google/sentencepiece
[11] https://github.com/google-research/bert/blob/master/create_pretraining_data.py

BERT python library. It turns all the sentences into the tokenized sequences, which consist of all the variables described in section 3.1. These sequences can then finally be used to pre-train the BERT models, which will be described in the next section.

5.2 Model Pre-Training

Three different BERT models were pre-trained in this thesis. The trained models are named PureBERT, MixedBERT and RetrainedBERT. Naming makes it easier to reference them in this thesis. Figure 6 was created to give a better overview of the pre-training and fine-tuning process.

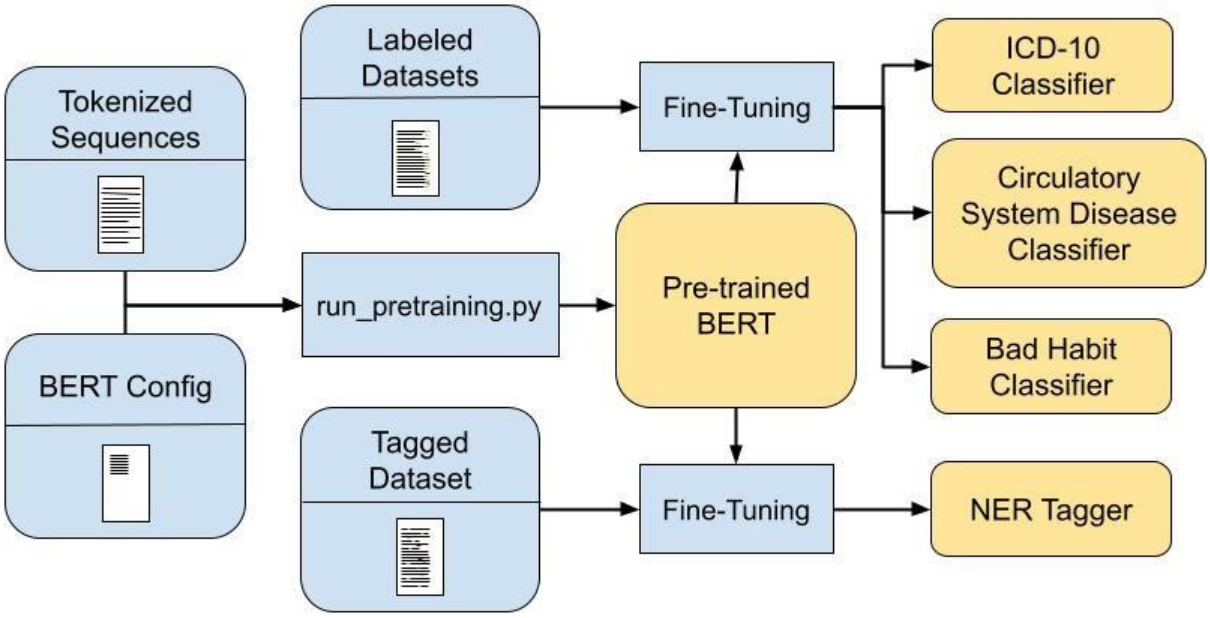

Figure 6. The overview of training and fine-tuning process.

The Pure and Mixed models were trained from scratch. There are only two differences between them. First, the Pure model only uses the sentences from the clinical notes. The Mixed model uses the sentences from both the notes and the Estonian National Corpus 2017. The second difference is the vocabulary size. To accommodate for the new words from the Estonian National Corpus 2017, the mixed model's vocabulary size is 75K, instead of the 50K that the other 2 models use. The RetrainedBERT requires more explanation, thus the subsection 5.2.2 is reserved for it. The next chapter goes over how the pure and mixed models were trained.

5.2.1 Pre-training From Scratch

At this point, the data has been cleaned and prepared for the use in pre-training. However, the model also must be configured. The hyperparameters used to pre-train the pure and mixed models were like as were used to train EstBERT [6]. Table 3 shows the parameters.

Table 3. The parameters used for pre-training. * Vocabulary size is 75K for only the mixed model, 50K in case of the other models.

Param	**Value**	**Param**	**Value**
Train_batch_size	32	Optimizer	Adam
Max_seq_len	128	Learning_rate	1e-4
Max_predictions_per_seq	20	Vocabulary size*	50000 or 75000
Num_train_steps	90000	Num_warmup_steps	9000

Similar hyperparameters were chosen so that it would be better to compare the models, since the RetrainedBERT model was retrained from EstBERT. Also, many other models use similar parameters [4], [9].

However, the models in this thesis were trained a bit differently than EstBERT. EstBERT was trained in 2 stages, first they trained the model for 900k steps on MLM task and then they trained for another 900k steps on the NSP task [6]. Here, the models were also trained 900k steps, but each step consisted of both MLM and NSP tasks, because it is difficult to see if their method increased the performance of the model. Doing both tasks at the same time was also already implemented in the BERT[12] python library, so it made sense not to modify it.

The models were trained on 1 NVIDIA Tesla V100-SXM2-32GB GPU on one node of the High-performance Computing Centre at the University of Tartu . The pre-training took about 83h per model. Fortunately, the models could be trained in parallel. The training time was almost the same amount of time as it took for Tanvir et al to pre-train 1 stage of EstBERT, and with using less GPUs [19].

5.2.2 Transfer Learning

Transfer learning in this thesis means for the model to be trained from a source domain into a target domain. In this thesis, the source domain would be the natural Estonian language or the domain that the EstBERT adapted.

The RetrainedBERT model was created to attempt a more efficient way to train the models. The idea was to use the already pre-trained EstBERT model and adapt it to the clinical domain.

[12] https://github.com/google-research/bert

First, the vocabulary of EstBERT had to be modified, since the domains contain different words in different frequencies. This was done by combining the PureBERT's and EstBERT's vocabularies. All the tokens that were in both vocabularies were kept at the positions that they had in the EstBERT's vocabulary. Because this way their embeddings can be kept the same. All the other tokens were taken from the PureBERT's vocabulary and assigned to random empty positions until the vocabulary was filled.

Secondly, all the 12 layers and embeddings of tokens that were in both vocabularies, were frozen. The 12 layers make up most of the parameters in the model and it would be beneficial if those would not need to be modified. The select embeddings were frozen because the tokens that were present in both vocabularies already had trained embeddings and did not need to be modified.

The *optimization.py*[13] script from the BERT python library was modified to do so. The 12 layers were just removed from the list of trainable variables. The gradients for the embeddings layer had to be multiplied by a vector of ones and zeroes, where one indicates that the embedding should be trained and zero means that the embedding is frozen.

Thirdly, the *run_pretraining.py*[14] script was run on the same cleaned notes that were used to train the pure model. The model completed steps about 30% faster and only required 200K steps to be trained, which took about 13h and 20min to do. The results will be discussed in chapter 6.

Now that all the models have been pre-trained, the models can be fine-tuned. The next section describes how it was done in this thesis.

5.3 Sequence Classification

In this thesis, the models were fine-tuned on three sequence classification tasks. The idea of the sequence classification task is to assign a class to sequences according to a set of classes. The classes can be chosen as required by the user. This is a supervised task, thus labelled data is required to fine-tune the models on this task.

To fine-tune a BERT model on this task. A dataset first must be created such that for each sequence there is a label. The label is just an integer from 0 to n-1 (inclusive), where n is the

[13] https://github.com/google-research/bert/blob/master/optimization.py
[14] https://github.com/google-research/bert/blob/master/run_pretraining.py

number of classes and every integer corresponds to a different class. Once that is done, a pre-trained model with the SequenceClassification[15] model head should be loaded.

The next three subsections describe how the dataset for each individual task was created and how the models were fine-tuned.

5.3.1 ICD-10 Classification

The Digilugu is a database of discharge summary documents that contain both structured and unstructured fields. The structured fields represent information such as time of visit, patient demographics and diagnosis codes. Free text notes contain information about the procedures, medical history and treatment suggestions. Here the idea is to test the models, by predicting structured information on the discharge reports based on the unstructured notes.

One piece of information we can predict is the primary diagnosis of the patient, that is given as the International Classification of Disease version 10 (ICD-10) code. The ICD-10[16] has a hierarchical structure. There are 21 chapters of different diagnoses. Each of the chapters includes more precise diagnoses. For example, the codes *A00-B99* are for certain infectious and parasitic diseases. Where each individual code, for example *A50* and *B7* are more precise diagnoses. Each of these can also have sub-diagnoses. For example, *A50* has 10 sub-diagnose codes from *A50.0* to *A50.9*.

In this task only the 21 chapters of the ICD-10 code are classified. Using all the precise diagnose codes would have increased the number of classes to a too high number, which requires a lot of data and time to learn. Attempting that could yield some results, but it was not attempted in this thesis due to limited time and GPU resources.

Table 4. The parameters used for fine-tuning ICD-10 classifier.

Variable	**Value**	**Variable**	**Value**
Learning_rate	5e-5	Optimizer	Adam
Classes	21	Warmup_steps	2000
Sentences for training	190k	Sentences for testing	63k
Batch_size	32	Epochs	3

[15] https://huggingface.co/transformers/model_doc/bert.html#transformers.BertForSequenceClassification
[16] https://www.icd10data.com/ICD10CM/Codes

The letter and the first number were enough to assign all the diagnose codes to the correct diagnose chapter. These are the labels. So for example *A50.5* and *B7* would be given the same label. However, H32 and H62 were not given the same label, since the first belongs to the Diseases of the eye and adnexa chapter and the other one belongs to the Diseases of the ear and mastoid process chapter.

To tokenize the texts, instead of using the *create_pretraining_data.py* script like before, BertTokenizer[17] from the Transformers Python library was used. It adds the special tokens to the text, tokenizes the texts into *input_ids* and creates the *attention_mask* and the *token_type_ids*.

The parameters that were used to fine-tune all the models on the ICD-10 classification task can be seen in Table 4. Note that only a subset of the data was used, because increasing it would not make the results much better. Fine-tuning a model using these parameters takes about 2h on 1 NVIDIA Tesla V100-SXM2-32GB GPU.

5.3.2 Predicting Diseases of the Circulatory System

The task of predicting the diseases of the circulatory system (DCS) is quite like the ICD-10 classification task. But instead of just predicting the main ICD-10 classes, here only the circulatory system (or the I00-I99) subclasses are predicted.

Since the proportion of the I classes is around 8%, it makes sense to create an oversampled dataset to increase the proportions of the I subclasses. To do that only 2% of the other classes are put together into one class to represent them.

Table 5. The parameters used for training the models on the circulatory system disease prediction task.

Variable	**Value**	**Variable**	**Value**
Learning_rate	5e-5	Optimizer	Adam
Classes	40	Warmup_steps	2000
Sentences for training	751K	Sentences for testing	180K
Batch_size	32	Epochs	3

[17] https://huggingface.co/transformers/model_doc/bert.html#transformers.BertTokenizer

Some of the class I subclasses are quite rare (less than 5 instances out of 426K) so it cannot be expected that the model learns to differentiate these subclasses. To remedy that, all subclasses with less than the median number of instances (49 instances) are put together into another class.

Fine-tuning with the parameters shown in Table 5 took about 5h on the same GPU that was used in the previous task.

5.3.3 Bad Habit Prediction

The goal of this task is to classify sentences into the bad habit classes shown in Table 6. What makes this task different from the previous two is that the data had to be labelled by the author of this thesis. Thus, the amount of data is limited.

Table 6. The labels in the bad habit dataset.

Description	**Does not drink or smoke**	**Drinks in Moderation**	**Drinks too much**	**Has smoked before**	**Still smokes**
Label	0	1	2	3	4
Instances	297	94	95	27	171

To create this dataset, two taggers were used to automatically search for sentences containing words about drinking. Those taggers were not made during this thesis, but like many other taggers used in this thesis, these were also made by STACC. Once such sentences had been found, they were manually read through to fix the labels. For example, the sentence: "*Patsiendil esineb depression, alkoholi ületarbitamine*" would have the label 2. The sentence: "*Ei suitseta 1.5 aastat.*" Would be labelled 3. Some random sentences were also added so that the model would not predict that the patient is smoking or drinking every time. In total 684 sentences were labelled (see Table 6).

Table 7. The parameters used to train the models on the bad habit classification task.

Variable	**Value**	**Variable**		**Value**
Learning_rate	1e-4	Optimizer		Adam
Classes	5	Warmup_steps		100
Sentences	684	k-folds		5
Batch_size	8	Epochs		5

The parameters used to train the models on this task are displayed in Table 7. Note that some of the 0 class elements (25%) were randomly removed from the training set. This was done because during the search for the hyperparameters, the removal of some of the 0 class instances were also tested and removing 25% proved to be the best. During the hyper-parameter search, the best learning rate and the best number of epochs turned out to be 0.0001 and 5, respectively.

During the testing of these models, something interesting emerged. With about the same number of instances in each class, high number of epochs and low learning rate the PureBERT model was observed to be able to extrapolate. Table 8 displays three examples. In all three examples none of the words (except "patsient", which appears a lot) are also in the training dataset with labels other than the 0 label.

Table 8. Examples of extrapolation.

Sentence in Estonian	**Sentence in English**	**Predicted Label**
Põhjustas autoavarii.	Caused a car accident.	Drinks too much
Suri kopsuvähki.	Died of lung cancer.	Smokes
Patsient köhib palju.	The patient coughs a lot.	Smokes

This is an intriguing result, but for a truly useful model more labelled data and experimentation is needed.

5.4 Token classification

The difference between token classification and sequence classification is that in token classification, every token in a sequence is classified instead of the entire sequence. Here also, the user can choose which classes they want the model to predict and must use labelled data for fine-tuning.

Instead of just an integer per sequence as a label for that sequence. Now there must be a list of integers with the same length as the tokenized sequence i.e., there must be a label for every token.

The models were fine-tuned on only one token classification task, because there were not any already tagged and usable clinical notes in Estonian. The data for that one task had to

be made by the author of this thesis and thus the next subsection begins by describing how it was done. Followed by how the models were fine-tuned.

5.4.1 Units and Measurement Recognition

In this thesis, the objective of the NER task is to identify which tokens represent units and measurements. Extracting the units can be useful. For example, if someone wants to do some stats about people's blood pressure, but only has dated texts. They can then use a model which has been fine-tuned to accomplish this task to extract the measurements from the texts automatically.

To fine-tune a model on the NER task requires the tokenized sentences as *input_ids* and the corresponding tags for each token as *labels*. The tags and which unit they represent, can be seen in Table 9.

Table 9. The tags, the number of instances in the dataset and the number of instances after tokenizing.

Label	Non-unit	Other unit	Weight	Height	Blood-pressure	Time	Pulse	Blood sugar
Tag	0	1	2	3	4	5	6	7
Instances	8849	402	228	108	428	190	133	26
EstBERT	15536	488	237	116	608	207	168	34
PureBERT and RetrainedBERT	10403	412	233	109	432	198	134	26
MixedBERT	10407	414	231	110	433	195	134	26

There was no tagged data to fine-tune the models on this task. Thus, the dataset had to be made as a part of this thesis. Here is how it was done:

1. The notes were cleaned and split into the sentences like described in subsection 5.1.1.
2. The sentences were tagged with the unit tagger from section 4.2. The modified version of this tagger separates the numbers and units from the remaining sentence.
3. The numbers, units and text segments created in the previous step were then split into tokens and each token was assigned a tag. If it was a text token, then the tag 0 was assigned. If a number or multiple numbers were associated with a unit, then some other than 0 tag was assigned to the token. Although some of the units were wrongly tagged, using the tagger was still helpful, because the tokens that were

tagged, helped to keep track of the tokens and their tags in the next step. This step resulted in sentence and tag list pairs. In the sentences, each token is separated by a space and the tags are aligned with the tokens. For example, a sentence and the corresponding tag list would be "*RR 150 / 80 mmHg , paremas kõrvas kuulmekilel võõrkeha, 4 4 4 4 4 0 0 0 0 0 0*".

4. The pre-tagged sentences were then fixed manually. In total about 1700 sentences were gone through by the author of this thesis, which took about 3h. Around 800 of the sentences did not contain any units or were duplicates and were deleted to reduce the number of non-unit tags. Some sentences that did not contain any units were kept, to show the model, that every sentence does not contain tokens that need to be tagged. The pre-tagging which was done in the previous step was then fixed manually. In the end both the number and unit symbols had a tag other than 0, but also some related words like the month of a year were tagged.

That was how the NER dataset with 920 tagged sentences was made. Table 9 also shows how many of each token there is. The number of different tags and the size of the dataset can be expanded as required but using 8 different tags and 920 sentences was enough for this thesis.

Now onto the creation of the *input_ids* and the corresponding *labels*. In this thesis, the *input_ids* were created by splitting the sentences at the spaces into a list of tokens and then using the model's tokenizer to create the *input_ids*. However due to out-of-vocabulary words, the tokenizer splits some of the tokens into sub-word units, which makes the tags and *input_ids* unaligned. A way to fix this issue is to use the model's tokenizer to turn the *input_ids* back to the tokens. Doing so, will not put all the tokens back together, but reveal the sub-word units. Then the sub-word units can be used to align the tags to the *input_ids* by first finding what tag the initial token has and then extending the tag to cover the corresponding sub-word unit. Finally, the tags can be used as the *labels* to fine-tune the models.

Table 10. The parameters used for fine-tuning NER token classifier.

Variable	**Value**	**Variable**	**Value**
Learning_rate	5e-5	Optimizer	Adam
Sentences	920	k-folds	5
Batch_size	8	Epochs	5

The parameters used for fine-tuning the models on the NER task can be seen in Table *10*. This time the fine-tuning was even faster. Due to the small dataset size, it only took about 20 seconds to fine-tune a model on 1 NVIDIA Tesla V100-SXM2-32GB GPU.

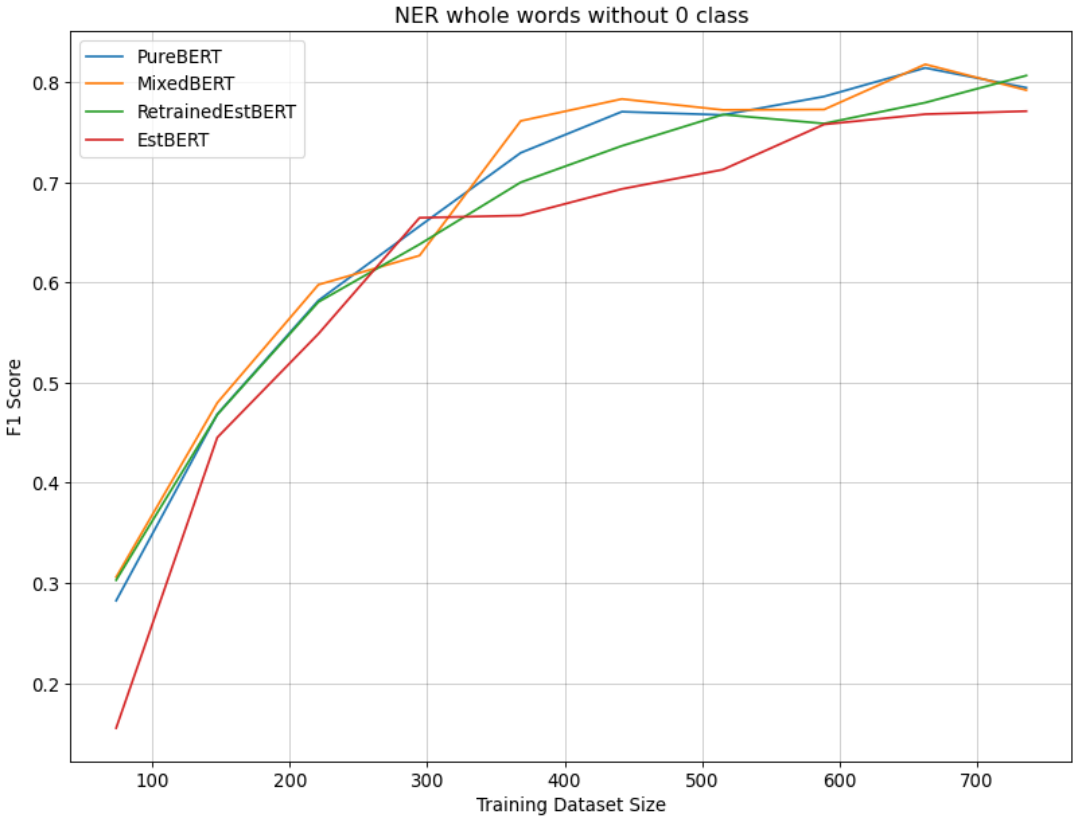

Figure 7. The amount of data vs F1-Score graph

The creation of the dataset for this task raised the usual question. How much data is required? Figure 7 helps to give the answer. Here the cross-validation method and the same parameters as before were used. 187 Random sentences (excluding the ones used for training) were used to validate each time. As can be seen in Figure 7, using an oversampled dataset with around 938 sentences was good enough for this task.

The sub-word units created another issue while validating. When there are a lot of out-of-vocabulary words, the number of sub-word units increases. This can make it unfair to compare the models that use different vocabularies. In the case of EstBERT, the number of non-unit tags increased much more than the other tags (see Figure 9). The non-unit tags were tagged correctly more often than the others, increasing the accuracy.

To make the comparison fairer, the predictions on the sub-word units were put back together into full tokens. The prediction on the token would then be elected from among the predictions on the sub-word units. So, the tag that appears most often in the token's sub-

word units, is set as the prediction for that word. If there are multiple equally frequent tags for a token, then the tag for the token is randomly picked from the equally frequent tags. For example, let's say that the token "blood pressure" got split into the sub-words: "blood", "##pre", "##ssu", "##re", and let's also say that a model predicted the tags: 4, 4, 0, 4 for these sub-words. Then the tag for the token would be elected to be 4.

To make the comparison even better, the non-unit tags were removed. This helps to better show how well the models tag the units and measurements. If non-unit tags were included, then those would improve the metrics of the models but would also hide issues. For example, the model might get 97% accuracy or f1-score with the non-unit tags included, instead of the 80-84% (see Figure 9), but still miss all the relevant tags, because it only predicted the non-unit tags.

6. Results

In this chapter, the results of this thesis are displayed, discussed and conclusions are drawn. More precisely in section 6.1, it is described how the taggers, that were used to clean the notes, were tested. It shows that cleaning the notes with the taggers, can improve the accuracy of the models. Section 6.2 discusses how the models learned in the pre-training phase. In section 6.3 the results of fine-tuning all four models on the four tasks are displayed and discussed. In section 6.4, a statistical test to determine if the models are statistically the same is described. There the best model is selected and renamed EstMedBERT. Finally, in section 6.5, the hypotheses erected in chapter 2 are checked.

6.1 Tagger testing

To test the taggers, 6 BERT models were trained from scratch on a subset of data and evaluated on 4 different tasks: MLM, NSP, ICD-10 classification and NER tagging. All the models had the same parameters for pre-training (see Table 11). The parameters used for fine-tuning the models on the ICD-10 classification and NER task were the same as was used for fine-tuning the four big models.

Table 11. The parameters used to pre-train the tagger testing models.

Param	**Value**	**Param**	**Value**
Batch_size	32	Optimizer	Adam
Max_seq_len	128	Learning_rate	1e-4
Max_predictions_per_seq	20	Vocabulary size	50000
Num_train_steps	200000	Num_warmup_steps	2000

The only difference the models had was the differently cleaned notes. So, in each case the data was cleaned by only using the taggers shown in the Table 12. For all the datasets, the anonymized tagger was also used since the tags had to be removed or replaced anyway. A small version of the final BERT model was also trained to see the effect of using all the taggers. The notes, that the small and the final models use were not cleaned by using the unit tagger, because the units are required for the NER task and it would only remove some of the units.

The accuracies of MLM and NSP were acquired by evaluating the models straight after pre-training, because the models learned the tasks while pre-training. In case of the classification

and NER tasks, the models were first fine-tuned and then validated. These models were fine-tuned and evaluated on these two tasks like the main models were fine-tuned in this thesis.

Table 12. Accuracy of BERT models on differently cleaned notes. * A model that always predicts the most frequent answer is set as the baseline. The values in **bold** show the best result in a task.

Metric: Accuracy	**Baseline***	**Anonymized**	**Event Header + Anon**	**Table + Anon**	**Unit + Anon**	**Date + Anon**	**All except Unit**
MLM	0.088	0.768	0.780	0.789	**0.810**	0.807	0.798
NSP	0.5	0.974	0.969	0.994	0.985	0.981	**0.995**
ICD-10 Classification	0.147	0.656	0.658	0.653	0.653	0.657	**0.696**
NER	0.265	0.637	0.671	0.650	0.668	0.646	**0.752**

The results can be seen in Table 12. As can be seen, the taggers in most cases were much better when compared to the baseline. When comparing with the anonymized tagger, using other taggers with it usually yielded better results. Using all the taggers together however yielded the best results in most tasks.

6.2 Pre-Training Validation

The pre-training of the models was validated by fine-tuning all 19 checkpoints of the three main models on the ICD-10 classification task and the NER tasks. The process was parallelizable so per model it took about 4-16h to fine-tune all the checkpoints of one model (depending on the availability of the GPUs. For comparison, the EstBERT model was also fine-tuned on these tasks, but since there were not any available pre-training checkpoints, only the final version of EstBERT was fine-tuned on these tasks.

Figure 8 displays the F1 scores of the ICD-10 classification task at each checkpoint. Each checkpoint was made after 50K training steps. As can be seen, the three models that were pre-trained in this thesis perform better than the EstBERT model. Which was to be expected since the clinical notes' domain is quite different from the natural Estonian language domain.

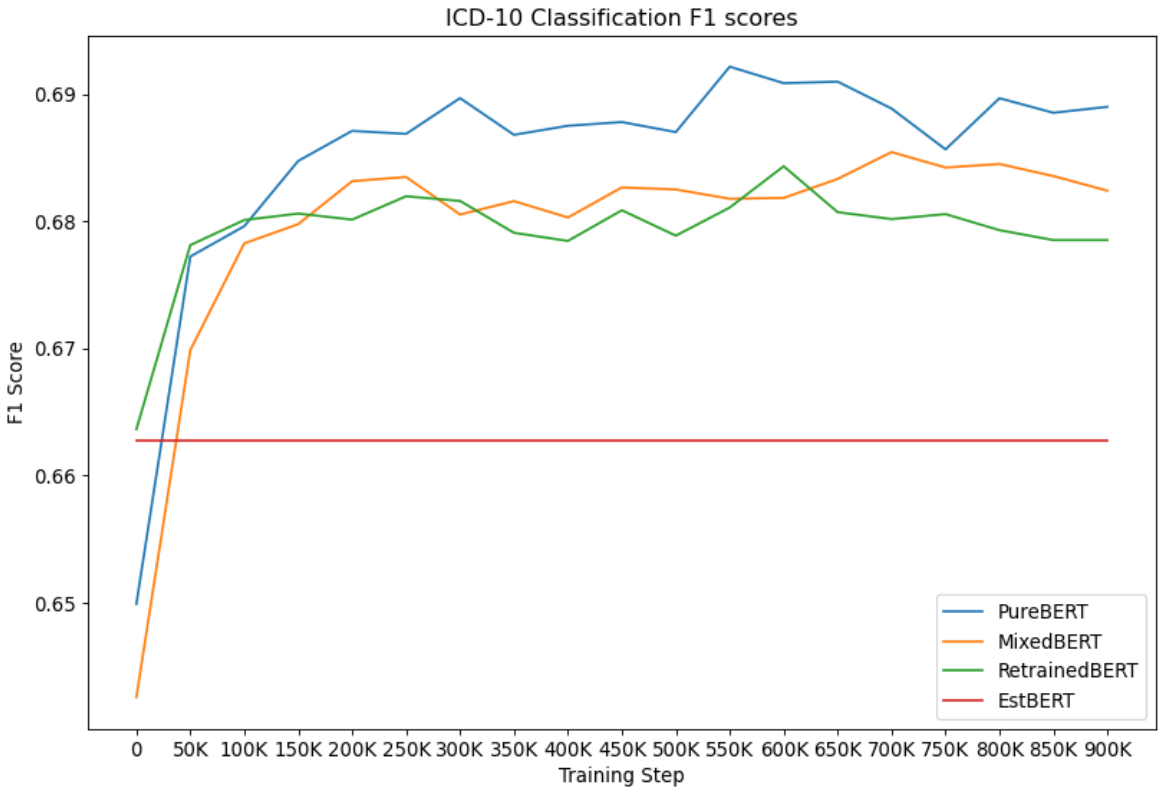

Figure 8. The comparison of the models in the ICD-10 classification task.

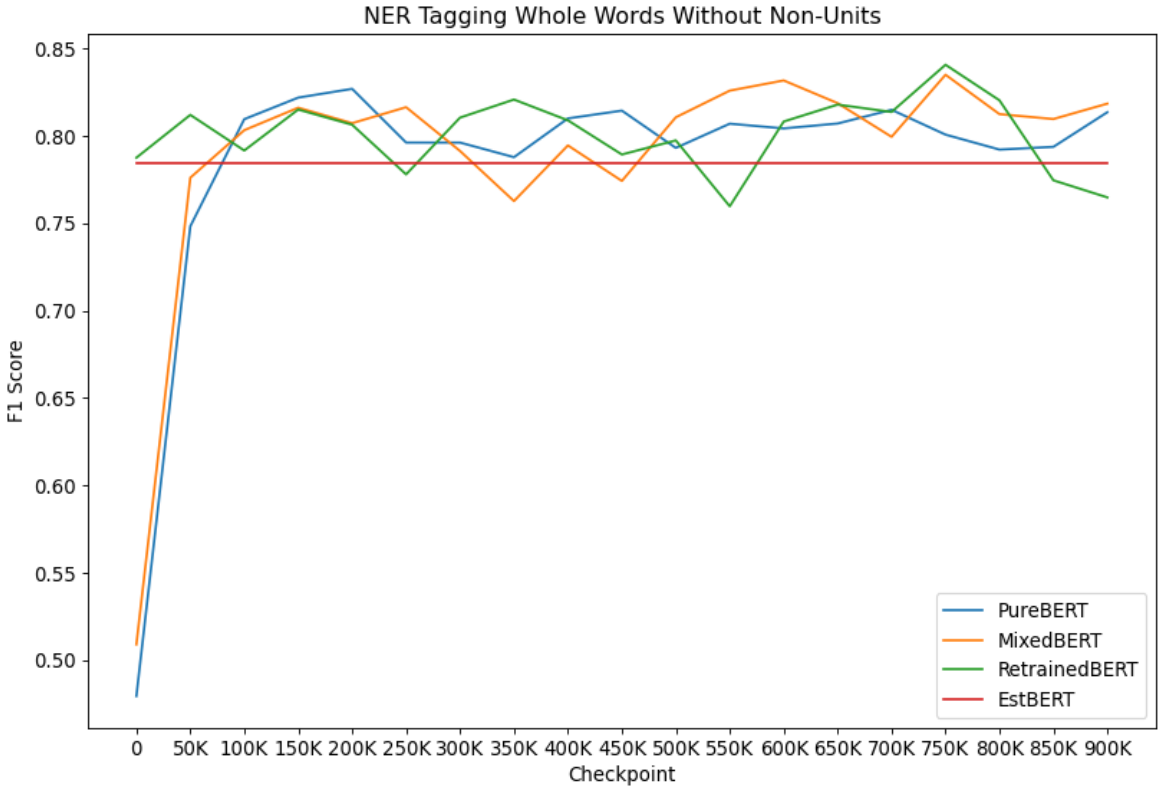

Figure 9. The comparison of the models in the NER task.

Like Gu et al. found that training from scratch purely on one domain results in the best model, the same conclusion can be drawn from this task [8]. The PureBERT model was better than the other models.

The Mixed and Retrained models were consistently a bit worse than the pure model. However, the two models performed quite similarly to each other. Maybe the reason is that both models were trained on the notes and the Estonian National Corpus 2017. But training time wise, the RetrainedBERT achieved better results with less training steps and completed the steps faster than the MixedBERT model.

Now onto the NER task. Since the amount of data was quite low and the fine-tuning took about 20 seconds each time, the cross-validation method was used to get more reliable results.

The models in this task achieved quite similar results (see Figure 9). Further testing is described in the section 6.4, where a statistical test is done to determine if the models are different or not.

6.3 Fine-Tuning Validation

The results of all the models in all four tasks can all be seen in Table 13. Note that the models were fine-tuned 50 times (with different seed every time) with 5 folds (250 times in total) on the Bad habits classifiers and Unit Recognition tasks to get even more reliable results. Without doing that, the results for those tasks varied quite a bit each time.

PureBERT turned out to be the best model in all the tasks. MixedBERT had quite similar scores to PureBERT. EstBERT turned out to be the worst model out of the four. The Retrained model also got quite good results. Keep in mind that RetrainedBERT took much less time to train than the others, so the way RetrainedBERT was trained, could be used in the future. Although DistilBERT could be even better option if a fast model is required [20].

Table 13. The results of the four fine-tuning tasks. The best results are shown in **bold.**

ICD-10	PureBERT	MixedBERT	RetrainedBERT	EstBERT
Recall	**0.695**	0.689	0.685	0.670
Precision	**0.690**	0.684	0.679	0.667
F1	**0.689**	0.682	0.679	0.663
CVD				
Recall	**0.567**	0.566	0.560	0.548
Precision	**0.576**	0.574	0.568	0.555
F1	**0.564**	0.563	0.556	0.543
Bad Habits				
Recall	**0.837**	0.836	0.829	0.823
Precision	**0.841**	**0.841**	0.837	0.831
F1	**0.836**	0.835	0.829	0.823
Unit Recognition				
Recall	**0.773**	0.772	0.764	0.749
Precision	**0.936**	0.934	0.922	0.922
F1	**0.836**	0.835	0.822	0.812

6.4 Are the Models Different?

The main goal of this thesis was to create a model that can make use of the clinical notes. Before this thesis, EstBERT was already available. It could have been used to fine-tune models on the clinical notes. Although the results of fine-tuning the three models that were pre-trained in this thesis were mostly better than fine-tuned EstBERT. The models still might be too similar to each other to consider them different from EstBERT. To show that the trained models are indeed different from EstBERT, a statistical test was conducted.

The idea is to compare the differences of model predictions. To do that, a random sample of at least 100 instances where model A and model B predict different labels are selected. Then the following conditional probabilities were estimated:

$$P(TrueLabel = 1 | A = 1 \; and \; B = 0) \quad (1)$$

$$P(TrueLabel = 1 | A = 0 \; and \; B = 1) \quad (2)$$

Models in this thesis have more than 2 classes, thus read the 0 as false label and 1 as the correct label. Once the probabilities had been estimated, a 95% confidence interval was calculated for both. In case the two confidence intervals overlapped, it could be said that the models are statistically the same. If these do not overlap, then the models are different.

Table 14. The statistical test results of the ICD-10 classifiers

ICD-10	Pure	Mixed	Retrained	EstBERT
Pure	1	0	0	0
Mixed	0	1	1	0
Retrained	0	1	1	0
EstBERT	0	0	0	1

Table 15. The statistical test results of the CVD classifiers

CVD	Pure	Mixed	Retrained	EstBERT
Pure	1	0	0	0
Mixed	0	1	0	0
Retrained	0	0	1	0
EstBERT	0	0	0	1

Table 16. The statistical test results of the NER task

NER	Pure	Mixed	Retrained	EstBERT
Pure	1	1	1	0
Mixed	1	1	1	0
Retrained	1	1	1	0
EstBERT	0	0	0	1

Tables Table 14, Table *15* and Table *16* display the results of the statistical test. 1 means that the two models are statistically the same and 0 means that they are not. The bad habit prediction task was not included here, because the number of labelled sentences was too low. The number of instances where two models predicted differently differed across the tasks. In the case of the ICD-10 task, there were about 2K of such instances. In the CVD classification, there were around 20K different predictions for each model pair. Because the dataset for the NER task was made semi-automatically in this thesis, there were around 100-200 of such instances per model pair.

In case of the ICD-10 and CVD classification tasks, it can be said, that the PureBERT model is different from all the other models (see Table 14 and Table 15). In the NER task case, it can be said that the Pure, Mixed and Retrained models are the statistically the same. However, all three models are statistically different from EstBERT in all tasks.

The PureBERT was statistically differed the most times across these tasks. That model also got the best results in most tasks. Thus, this is the EstMedBERT model that should be used for fine-tuning in the future and is the result of this thesis.

6.5 The Hypotheses

Based on these results, both hypotheses can be refuted. The first hypotheses stated that "Increasing training set size by adding text from other domains improves the performance of the resulting models." However, the opposite was observed. The Pure model achieved similar or better results most of the time.

The second hypotheses stated that "The existing EstBERT model already captures enough knowledge about language structure that slight further optimization with medical texts is sufficient". While the optimization did improve performance of EstBERT and was faster than pre-training from scratch. The models, that were pre-trained from scratch achieved even better scores.

7. Discussion

A lot could have been done to further improve the models. The notes could be processed even better. Some of the taggers, for example the Table Tagger, did not seem to work well and removed only a few tables. Also, instead of removing the tables and lists these could be modified. The sentence splitting could also be modified. Sometimes the SentenceTagger split the sentences at the wrong places. Which sometimes resulted in too short or half sentences.

The models could have been trained on more different types of down-stream tasks for example extractive question answering or summarization. However, there were no premade datasets for such tasks. Creating a dataset for these tasks would have been too time-consuming and thus were left out.

The transfer learning method discussed in this thesis could also be compared to DistilBERT model. That model is designed to be fast and lightweight but also loses some performance over BERT [20]. Would be good to know if it is better to pre-train DistilBERT from scratch or to optimize a language specific pre-trained model.

While training the models on the bad habit classification task. The PureBERT model was tested on token sequences that only contained tokens that were not in the training set or were not related with smoking or drinking. The model was still able to correctly classify some of the sequences. In the future it might be interesting to further experiment with it.

8. Conclusion

The main goal of this thesis was to use NLP methods to make the clinical notes usable found in the Estonian e-health system usable by ML methods. To achieve this goal, the clinical notes were first cleaned by using various taggers and split into sentences.

Two models were pre-trained from scratch, and one was re-trained on these sentences. The models are named PureBERT, MixedBERT and RetrainedBERT. The PureBERT was pre-trained on only the clinical notes. MixedBERT in addition to the clinical notes, was pre-trained on the sentences from the Estonian National Corpus 2017. This thesis also had a secondary goal. To see if the already pre-trained EstBERT model can be optimized on the clinical notes more efficiently than pre-training the model from scratch and achieve on par results. Thus, the Retrained-BERT model was created to test an optimization method.

All four models were fine-tuned and evaluated on three classification and one token tagging down-steam tasks. The first task was the classification of the ICD-10 main diagnose codes. The second classification task was similar. There only the diseases of the circulatory system subclasses were classified. The objective of the final classification task was to predict bad habits like smoking and drinking. In the token tagging task, the models were used to tag units and measurements in the texts. The two last tasks did not have a pre-labelled dataset, thus for those tasks, the datasets were made in this thesis. The making of datasets showed that relatively good results can be achieved with not that many datapoints.

Out of the four models, the PureBERT achieved the best results. Thus, this is the resulting EstMedBERT from this model. Since it is better than the EstBERT model when it comes to clinical notes, it can be said that the main goal of this thesis was achieved.

The secondary goal was tested, but the method that was attempted in this thesis to optimize a model did not yield on par results with the models that were trained from scratch. However, the optimized model still performed better than the EstBERT model and the optimization process was faster than pre-training from scratch.

References

[1] e-Estonia. ‘E-Health Records’. Accessed 3 March 2021. https://e-estonia.com/solutions/healthcare/e-health-record/

[2] F. Jiang *et al.*, ‘Artificial intelligence in healthcare: Past, present and future’, *Stroke and Vascular Neurology*, vol. 2, no. 4, Art. no. 4, 2017, https://doi.org/10.1136/svn-2017-000101

[3] S. T. Rosenbloom, J. C. Denny, H. Xu, N. Lorenzi, W. W. Stead, and K. B. Johnson, ‘Data from clinical notes: A perspective on the tension between structure and flexible documentation’, *Journal of the American Medical Informatics Association*, vol. 18, no. 2, Art. no. 2, 2011, https://doi.org/10.1136/jamia.2010.007237

[4] K. Huang, J. Altosaar, and R. Ranganath, ‘ClinicalBERT: Modeling Clinical Notes and Predicting Hospital Readmission’, *arXiv:1904.05342 [cs]*, Nov. 2020, Accessed: Jan. 11, 2021. [Online]. Available: http://arxiv.org/abs/1904.05342

[5] Y.-M. Kim and T.-H. Lee, ‘Korean clinical entity recognition from diagnosis text using BERT’, *BMC Medical Informatics and Decision Making*, vol. 20, 2020, https://doi.org/10.1186/s12911-020-01241-8

[6] H. Tanvir, C. Kittask, and K. Sirts, ‘EstBERT: A Pretrained Language-Specific BERT for Estonian’, *arXiv:2011.04784 [cs]*, Nov. 2020, Accessed: Jan. 13, 2021. [Online]. Available: http://arxiv.org/abs/2011.04784

[7] E. Alsentzer *et al.*, ‘Publicly Available Clinical BERT Embeddings’, *arXiv:1904.03323 [cs]*, Jun. 2019, Accessed: Mar. 17, 2021. [Online]. Available: http://arxiv.org/abs/1904.03323

[8] Y. Gu *et al.*, ‘Domain-Specific Language Model Pretraining for Biomedical Natural Language Processing’, *arXiv:2007.15779 [cs]*, Feb. 2021, Accessed: Mar. 31, 2021. [Online]. Available: http://arxiv.org/abs/2007.15779

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, ‘BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding’, in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota, Jun. 2019, pp. 4171–4186. https://doi.org/10.18653/v1/N19-1423

[10] A. Vaswani *et al.*, 'Attention Is All You Need', *arXiv:1706.03762 [cs]*, Dec. 2017, Accessed: Feb. 02, 2021. [Online]. Available: http://arxiv.org/abs/1706.03762

[11] 'Estonian National Corpus 2017', Mar. 17, 2021. https://metashare.ut.ee/ (accessed Mar. 17, 2021).

[12] T. Mikolov, K. Chen, G. Corrado, and J. Dean, 'Efficient Estimation of Word Representations in Vector Space', *arXiv:1301.3781 [cs]*, Sep. 2013, Accessed: Mar. 24, 2021. [Online]. Available: http://arxiv.org/abs/1301.3781

[13] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, 'Bag of tricks for efficient text classification', 2017, vol. 2, pp. 427–431. https://doi.org/10.18653/v1/e17-2068

[14] M. E. Peters *et al.*, 'Deep contextualized word representations', *arXiv:1802.05365 [cs]*, Mar. 2018, Accessed: Mar. 24, 2021. [Online]. Available: http://arxiv.org/abs/1802.05365

[15] A. E. W. Johnson *et al.*, 'MIMIC-III, a freely accessible critical care database', *Scientific Data*, vol. 3, no. 1, Art. no. 1, May 2016, https://doi.org/10.1038/sdata.2016.35

[16] D. Bahdanau, K. Cho, and Y. Bengio, 'Neural Machine Translation by Jointly Learning to Align and Translate', *arXiv:1409.0473 [cs, stat]*, May 2016, Accessed: May 05, 2021. [Online]. Available: http://arxiv.org/abs/1409.0473

[17] E. Wallace, Y. Wang, S. Li, S. Singh, and M. Gardner, 'Do NLP Models Know Numbers? Probing Numeracy in Embeddings', *arXiv:1909.07940 [cs]*, Sep. 2019, Accessed: Apr. 06, 2021. [Online]. Available: http://arxiv.org/abs/1909.07940

[18] S. Laur, S. Orasmaa, D. Särg, and P. Tammo, 'EstNLTK 1.6: Remastered Estonian NLP Pipeline', in *Proceedings of the 12th Language Resources and Evaluation Conference*, Marseille, France, May 2020, pp. 7152–7160. Accessed: Jan. 26, 2021. [Online]. Available: https://www.aclweb.org/anthology/2020.lrec-1.884

[19] 'Rocket Cluster – Teadusarvutuste keskus'. https://hpc.ut.ee/vahendid/rocket-cluster-ee/ (accessed May 11, 2021).

[20] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, 'DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter', *arXiv:1910.01108 [cs]*, Feb. 2020, Accessed: May 08, 2021. [Online]. Available: http://arxiv.org/abs/1910.01108

Appendix

Glossary

Word	Description
Token	Part of a sequence of text. For example, it can be a word, a piece of a word, a number, punctuation mark.
Tokenization	The process of splitting a sequence of text into tokens.
Sub-word unit	A piece of a word.
n-gram	A contiguous sequence of n tokens.
Domain	In the context of this thesis, a domain is a set of similar texts. For example, the clinical notes form a different domain from the natural Estonian Language.
Structured data	Information that is organized and easily machine readable. For example, people's names, age, gender are structured.
Unstructured data	Information that does not have a predefined structure. Such as text.
Embedding	In the context of this thesis, an embedding is a numerical representation of a word or a token. Usually a vector.
Contextual embedding	A representation of a token, that is dependent on the token's context.
Encoder	A mechanism that compresses information.
Decoder	A mechanism that translates the compressed information back into understandable form.
Attention mechanism	A mechanism that focuses on more relevant parts of the token sequence.
Self-attention	Attention mechanism that only uses the input to learn which tokens should be attended to.
Pre-training	The process of learning language representations.
Fine-tuning	Further training of a pre-trained model for downstream tasks.
Downstream task	Supervised-learning tasks that utilize a pre-trained model.
Transformer	A deep learning model that composes of a stack of encoders and a stack of decoders, that utilize multi-head self-attention mechanisms
Oversampling	The process of increasing the proportion of the target classes.

Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Meelis Perli**,

(*author's name*)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Representation Learning on Free Text Medical Data,

   (title of thesis)

   supervised by Raivo Kolde and Sven Laur.

   (supervisor's name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I know the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Tartu, **12.05.2021**