UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Nesma Mahmoud

Fact Extraction from Medical Text using Neural Networks

Master's Thesis (30 ECTS)

Supervisor: Raivo Kolde, PhD

Tartu 2020

Fact Extraction from Medical Text using Neural Networks

Abstract:
Fact extraction from free text is a challenging task requiring a great deal of human effort to program regular expressions and build rule-based solutions. It is essential in the medical field where many care details are only stored as free text and automated fact extraction is the only way to interpret the large scale medical databases. Such medical texts represent communication between doctors and the text is often not syntactically valid, concepts are not represented consistently and the text is rife with misspellings. The described problems make it challenging to develop rule-based solutions to handle all the potential ways a fact might be written down. In this thesis, The effectiveness of neural networks was explored to do the fact extraction on texts from discharge reports on the Estonian Health Information System. We used the whole dataset of medical texts to train word embedding models. On the subsets of the data with annotations of particular facts, different classification models were tested to detect those. We found that employing pre-trained word embeddings allowed us to efficiently learn new models for fact extraction using relatively small amounts of annotated data. We managed to achieve an F1 score of 0.86% for a new tag using 732 samples as the training dataset, validate on 82 samples, and testing over 3258 samples.

Keywords:
Medical Named Entity Recognition, Fact Extraction, Word Embedding, Bi-Directional Long Short Term Memory, Interpretability

CERCS: P176 Artificial intelligence

Faktide tuvastus vabast tekstist kasutades sügavaid närvivõrke

Lühikokkuvõte: Faktide tuvastamine vabast tekstist on keeruline ja tööjõumahukas ülesanne, mida tavaliselt lahendatakse regulaaravaldiste ja reeglipõhiste süsteemidega. Meditsiini valdkonnas, kus säilitatakse paljusid ravi üksikasju ainult vaba tekstina, on automatiseeritud faktide väljastamine ainus viis suuremahuliste meditsiiniliste andmebaaside tõlgendamiseks. Sellised meditsiinitekstid esindavad arstidevahelist suhtlust ja tekst ei ole sageli süntaktiliselt korrektne, mõisteid ei kasutata järjepidevalt ja tekstis on palju kirjavigu. Kirjeldatud probleemide tõttu on keeruline välja töötada reeglipõhiseid lahendusi, et käsitleda kõiki võimalikke viise, kuidas fakte kirja panna. Selles lõputöös uurime närvivõrkude kasutamisvõimalusi, et eraldada fakte Eesti Tervise Infosüsteemi epikriisi andmetest. Kasutades suuremat tekstide andmestikku õppisime ELMO mudeli mis võimaldas parandada andmete esitust. Väiksematel annoteeritud andmestikel hindasime erinevate süvanärvivõrgu arhidektuuride täpsust ja tundlikkust. Leidsime, et eeltreenitud mudelid võimaldasid tõhusalt treenida uusi fakti eraldamise mudeleid suhteliselt väikeste annoteeritud treeningandmestike põhjal. Kasutades treeningandmetena 732 näidet, valideerides üle 82 ja testides üle 3258 näite saime F1-s skooriks 0,86.

Võtmesõnad:
rekurrentne võrk, fakti tuvastus, nimega üksuste tuvastamine

CERCS: P176 – Tehisintellekt

Acknowledgement

I would like to thank my thesis supervisor Dr. Raivo Kolde of the institute of computer science at the University of Tartu. It would not have been possible for me to defend if it was not with the support and help of Dr. Raivo Kolde. From the beginning, he led me in the right direction. I am also very thankful to STACC and Estonian Genome Bank for supplying me with the data that I used in this thesis.

I would also like to thank the professors of the Department of Natural Language Processing at the University of Tartu since the courses I took with them have taught me a lot on this topic. Finally, I must explain my gratitude to my family for their unconditional support, especially my husband and my future daughter who suffered a lot with me this semester. The achievement I had gained through the whole journey would not have been possible without them. Thank you!

Nesma Mahmoud

May 2020

Contents

1 Introduction

The volume of health data grows very rapidly worldwide drawing more attention to clinical research and application development. Although most of the data is gathered for treating patients, it is often saved as a free text, with questionable quality. Using variable criteria of writing and standards, all of which makes it difficult to implement large scale analyses, also building automatic decision support is challenging. Therefore it is required to build powerful tools, using natural language processing and workflow techniques, for cleaning the health data and transforming it into a functional format. This is the area where software technologies and applications competence center (STACC) [1] together with the University of Tartu has worked for over 10 years. The goal is to create a reliable database of medical facts based on discharge reports from the Estonian Health Information System that can be used for running clinical studies and would serve as a basis for new doctors' medical applications. Since the data is in Estonian, one just cannot run many of the models trained for other languages. The approaches of this study have to be trained and tested on Estonian data, to see what methods work the best in this particular setting and what kind of accuracy could be achieved. Automation is required to extract the information from the unstructured text.

The Named Entity Recognition (NER) is an important task in the natural language processing field which can be used in our task to tag the entities of the medical text. The first approaches applied for the NER task were based on rule-based methods, which need to be updated and adjusted regularly and also to define rules and patterns takes time, effort and they cannot be used in new domains. Moreover, rule-based systems only perform well with the particular purpose on which they have been developed for. Subsequently, the Support Vector Machine (SVM) and Conditional Random Field (CRF) methods were introduced, but these require handcrafted features, that can be costly to create, and huge amounts of manually annotated training data [LSK08]. With the advancement of neural networks, researchers started utilizing word embeddings and different feed-forward neural network architectures that can better understand the semantic and syntactic relationship between words [HWN+17]. However, the feedforward neural network has the disadvantage of not considering the sequential nature of the text. To overcome that limitation, the recurrent neural networks were introduced. Especially for text processing, these networks have the advantage of considering the context of the text. Recently, the bi-directional Long Short Term Memory (LSTM) model proved to be efficient in the NER task [CN16], as it takes effectively into account the context of the text from both sides.

In a bi-directional network, the text passes in a normal sequence to one LSTM layer and passes in reverse order to another LSTM layer. Convolutional Neural Networks (CNN) have also been used along with the BI-LSTM [CN16], as these have the benefit

[1] https://www.stacc.ee/en/

of extracting the features on the character level. Applying these methods to the fact extraction on Estonian medical records would allow to turn the task of defining manual rules, to a machine learning task, where the identification and annotation of a small training set of the facts of interest had to be done and then it could learn the rules automatically. As an additional benefit, the machine learning approach would be able to account for the context of the facts in sentences and be more robust towards misspellings and other mistakes.

1.1 Structure of the thesis

The history of the neural networks used for natural language processing is illustrated in section 2. The review of the literature is listed in the related work in section 3. The experiments that were developed for this thesis along with the setup details are explained in section 4, while section 5 represents the quantitative and qualitative results of our experiments. Section 6 addresses the discussions, challenges, and potential future work. Finally, the conclusions of this work are addressed in section 7.

2 Background

This section provides a technical overview about the RNN, LSTM, BILSTM and NER techniques.

2.1 Recurrent Neural Networks - RNN

Recurrent Neural Networks (RNNs) are well-known models that work efficiently over the sequential data, also it has shown tremendous promise for many NLP tasks. The textual data is considered as sequential data, which can be interpreted by the RNN. The meaning of recurrent is the ability of the network to perform the same function for each token of the sequential text. The typical neural networks presume that all inputs and outputs are independent of each other, whilst in RNN each element of the input is processed at a time, It also keeps track of the history of the previous elements in the text. RNN can process the sequential text using common weights shared between the input elements from the beginning of the training to the end. The RNN has a drawback that affects its efficiency in text processing which is the vanishing or exploding gradients [JZS15] that occur due to the back-propagated gradients while training over the network layers. The vanishing gradients occur due to the presence of many activation layers in the network where the gradients of the loss functions can approach zero, which leads to a vanishing gradient, while the accumulation of the gradients throughout the network layers, leads to an exponential increase of error gradients in the model which leads to an unstable network. It also has the drawback of not considering the future inputs of

the current element state. That's why LSTM and Gated Recurrent Units (GRUs) have been introduced to cope with traditional (Vanilla) RNN drawbacks. The RNN, in general, are used in many applications such as the music generation, sentiment classification, machine translation, and named entity recognition. Figure 1 shows the simple recurrent neural network [MKB+10], where the output depends on the current input along with the context of the previous input.

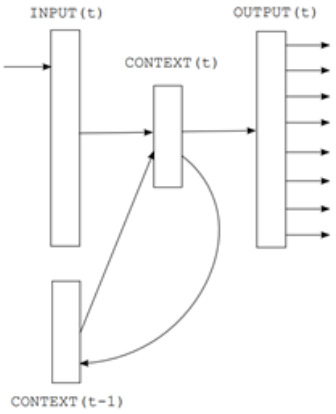

Figure 1. Simple RNN architecture [MKB+10]

2.2 Long Short-Term Memory Networks - LSTM

The LSTM is a variation of RNN that was designed to cope with the vanishing gradients' problems of traditional RNN [HS97]. Using the mechanism of gating to the input of the layer to remember the previous inputs where half of the input goes through the memory cell and the other through the working memory. So it can learn features from previous inputs while generation outputs, which is very important in the text processing since a single token from the series of text is related to others and has a sure effect on its neighbors. The main factor for the RNN's success is the sharing of parameters which utilizes the relationship between the input and its context. In short, it can take many input vectors and produce many output vectors taking into account the weights of the inputs and also the hidden vectors that represent the context of previous inputs.

Figure 2 shows the gates of the LSTM memory cell, where the forget gate decides which information to keep or to discard from the previous hidden state by transferring it with the current input to a sigmoid function and based on the value it decides how much to ignore and how much to keep. While the input gate acts as the update to the cell state using the current information. Lastly, the output gate is responsible for deciding what the next state should be.

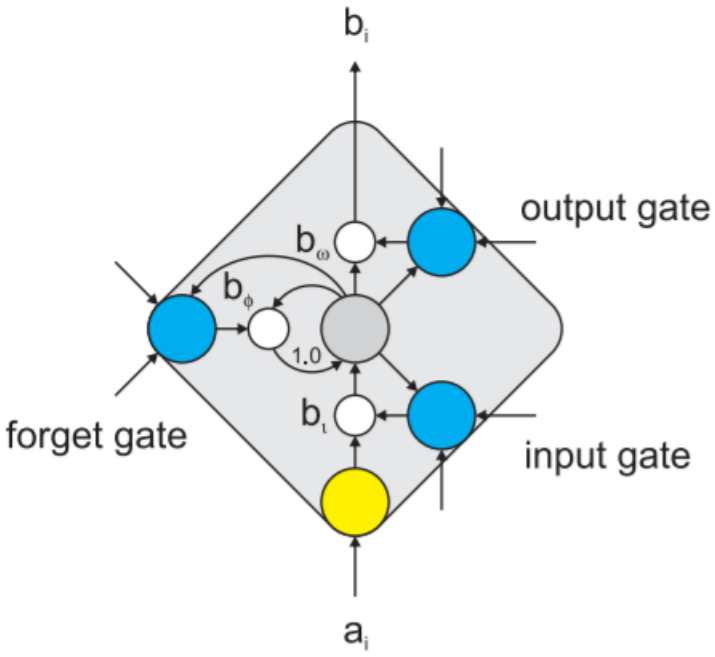

Figure 2. LSTM memory cell with gating units [SSN12]

2.3 Bidirectional Long Short-Term Memory Networks - BI-LSTM

In-text processing, it's not only about the relationship of the current state with the previous inputs in the sequence but also it needs the future inputs to repair the past. To fully grasp the meaning of the new tokens, it is needed to learn what is coming next. Bi-LSTM consists of a forward layer and a backward layer. The input sequence is fed to the forward layer in normal sequence, while the text is processed in reverse order by the backward layer, starting from the last word, until the first word of the sentence. The hidden states from both layers are concatenated for each word generating an intermediate representation sequence [ZZHY15]. So that the information from the previous and forward tokens are taken into account. This means that for each step the network has access to the complete document and can deduce the right label from that information because the output of each network at each time step goes through a softmax layer that uses log-probabilities to decide the tag of each token.

2.4 Named Entity Recognition - NER

The information extraction process involves the recognition of the entities in the text, those entities could be countries, organizations, or locations. In our case, the entities

are the measurement results of the patient health tests, such as blood pressure, various measurements, lab results, etc. Named entity recognition (NER) is a specific task of information extraction, which is a tagging task for the word-by-word sequence of data. To detect the entities, the dataset has to be annotated in a specific way, where each word in the text must have a tag. Whether the tag is the target needed or some random unknown words (See example Figure 3). Then the goal of a neural network like the Bi-LSTM is to understand those sequences of tags. Then another layer over it to get the probability distribution over all NER tags.

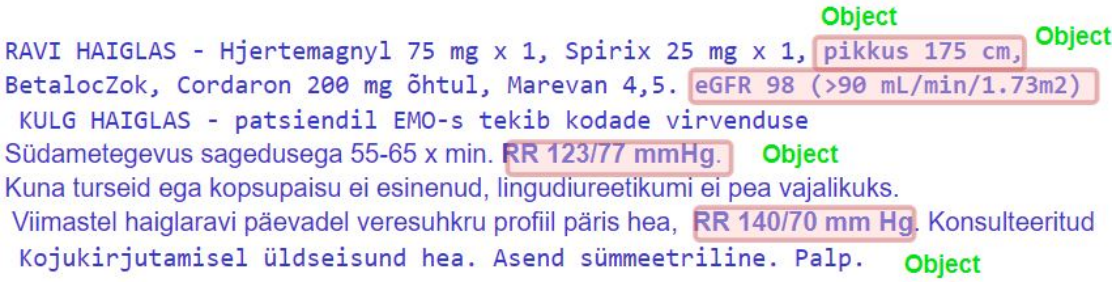

Figure 3. An example of the dataset used, where the highlighted words are the objects that needed to be to recognized.

2.5 Word embeddings

The basic way for text representation is using the one-hot encoding where the presence and absence of each word in the sequence are translated into 0 and 1 correspondingly. For text, this representation is not effective since we have many unique words, so most of the values will be zeros, and the representation will be sparse. To improve the representation of the text, the word embeddings were introduced, making it more compact. The aim is to move the high-dimensional word space into a lower-dimensional area, thus holding identical terms in the new space semantically close to one another. It helps the neural networks to improve their ability of understanding and learning from the textual data. The embeddings represent the data as a hidden layer composed of a lower-dimensional vector. The word embeddings can recognize the semantic and syntactic meanings of words [LG14]. Furthermore, Word embeddings are used most of the time in natural language processing tasks including named entity recognition tasks. There are some static embeddings such as word2vec and glove but they have a problem that they cannot understand the context of the text so they generate the same embedding for the same word in different contexts. To overcome the limitations of the static word embeddings, the dynamic embeddings such as ELMO [PNI+18], BERT [DCLT18], and FLAIR[ABV18] were developed. These are contextualized word representations as these

consider the context of words while representing it. The dynamic embeddings have proved to out-perform the static embeddings [WCZ19]. Word embeddings can be learned in an unsupervised manner from unannotated and unrelated corpora of text. Thus, by using word embeddings the transfer learning is essentially done. The pre-trained models may be used on a large dataset and then fine-tune these models for specific tasks rather than training a new model from scratch. Word embeddings take into consideration the context of the words. So each word will have a different embedding if it appeared in another context.

The dynamic embeddings language model has the objective of predicting the next token in the sequence while taking into account its context. It's useful in NLP tasks, especially, the NER. Moreover, the embedding makes a huge difference when the task has a small training dataset. With the weights trained on larger external datasets, the model can "understand" common textual features, and does not have to learn them from the existing small labeled training set. This process helps to elevate the accuracy of the model to represent the features of the input text, instead of depending only on the small input dataset. This approach is effective for the NER task. When using pretrained word embeddings, there is an option to fine-tune the model until reaching the required representations.

2.6 CNN for NLP task

It is a special kind of neural network which reflects a feature function that is applied to words to urge higher-level characteristics from the text. In sentence representation with CNN, it needs to have the sentences tokenized into words then represented as a word embedding matrix. The CNN consists of convolutions and pooling layers finely tuned with some hyper-parameters, those convolutional layers use filters that scan the inputs. The text is represented as a matrix where each row represents a word, the filters slide over the rows of the matrix with an outlined window size to provide the feature maps which then represent the entire sentence. CNN may be applied onto characters of the text directly needless of pre-trained embeddings. CNN will provide a strong result for the NLP, as it can derive the character-level features to be included in the NER and grasp the meaning in the context windows [CN16]. The convolution and max-pooling layer were used for any term to remove a new vector function from its character embeddings. Fig 4 below demonstrates how the sentence is seen in the CNN layer.

2.7 Interpretability

Usually, the neural networks and text recognition models are black boxes, with the internal details uninterpretable as many features are contributing to the prediction. Usually, we rely on the evaluation metrics to measure the performance of our model, but we should also seek to understand how the model arrived at the particular prediction.

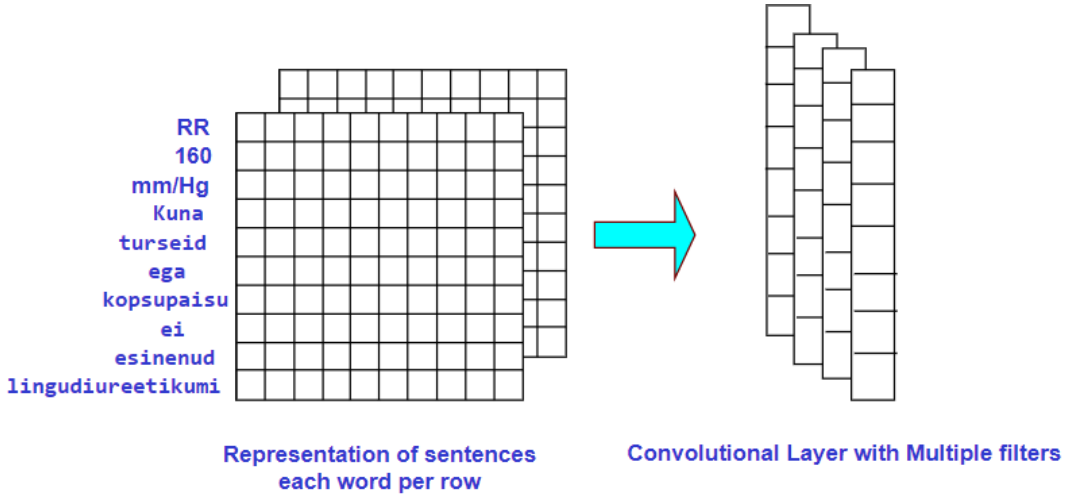

Figure 4. Representation of a sentence in convolutional layer.

Especially in text processing, having a good understanding of the semantics of the text and the reasoning process of the successful model should reflect that. Methods have been developed to visualize the decision process and that allows us to understand why the model predicted this class and which features contributed the most.

In this thesis, LIME [RSG16] library was used, which is an algorithm that can reliably represent any classifier or regressor's prediction by approximating it locally to an interpretable model. The purpose of the LIME model is to understand the reasoning behind the decision made by the black-box model. It probes the model with slight variations of the input data to understand what features are the most relevant for making the prediction. More specifically, it generates a dataset that contains some modified samples of the text that need to be interpreted, and the prediction of the black-box model for those samples. After that, Lime uses the prepared dataset to train it using an interpretable model. Ultimately, this interpretable model may more or less clarify the observations of the black box model for this local prediction.

3 Related work

There is a huge amount of research done in the field of Natural language processing, One of the sub-fields that received high attention from the researchers is the Named Entity Recognising. As the purpose of this study is to develop a Medical Named Entity Recognition, it was, therefore, necessary to review the literature to know the state-of-the-art approaches used by other researchers in this area. Throughout this section, some of the most efficient literature research will be presented. Clinical named entity recognition was

tackled by Yang et.al, [YLQ+19] they developed a bidirectional-LSTM layer along with a CRF based on multitasking attention, they also used pre-trained Elmo model for word embeddings and managed to enhance the recall of the clinical named entity recognition task. Kai et al. [XZHL17] have also used the bi-lstm layers with CRF layer for the medical named entity recognition, and applied this model over 2 different datasets and proved its efficiency among the baseline methods, They also demonstrated an interesting finding which is the consequence of increasing the number of measurements of the term embedding and the BI-LSTM layer for the Precision, Recall, and F1 ratings. They observed that the F1 value of the metric improves with the increase in the training set.

Zhang et al. [ZZL15] used character-level convolutional networks for the classification of the text, they preferred to use the convolutional layers for text processing, and applied it to different size datasets and reported the findings, this paper had a target to enhance the text classification using convolutional layers, and they managed to achieve it and proved that it outperforms the traditional models as the bag of words, TFIDF and n-grams. Xishuang et al. [DCQ+19] combined the deep transfer learning with the multitask bi-directional LSTM for applying the named entity recognition on Chinese medical electronic records, they mentioned that the tests, diseases, genes in Electronic medical records (EMR) can be extracted by the NER, but the challenge that faced them was that there are limited resources for the annotated medical entities. They managed to solve it by using data augmentation along with neural networks to enhance NER performance. The evaluation of the model that they developed has proved its efficiency in terms of F-score.

Mandhan et al. [MN+16] used Stanford Named Entity Recognition NLP libraries to extract the numerical attributes and values from the clinical texts and then associated the attributes to values using relation extraction modules. They used the conditional random fields for the extraction and SVM for relations extractions of the values. And the joint results had a good accuracy of 95% as mentioned by them. Finally, The one way to grasp the literature behind a scientific topic is by reading survey papers, one of the most recent surveys for the NER is [LSHL20]. where Li et.al listed a collection of annotated datasets for the NER task, they also showed the NER tools on the market. Furthermore, they presented the NER evaluation metrics which are divided into exact match evaluation and relaxed match evaluation. The exact match evaluation depends on the boundary detection of the entity using the confusion matrix whereas the relaxed match evaluation depends on the correct predictions without considering the boundary of the entity. They have discussed the traditional approaches for NER followed by the deep learning techniques for NER. Finally, they reported an efficient summary of the recent studies over NER.

4 Implementation and Experiments

This section describes the implementation of a baseline model and some improvements and details about the experiments. The algorithms were implemented using Keras [C+15] library over Python 3.

4.1 Objective

We explore the use of neural networks to extract the facts from the free text fields of discharge reports from Estonian hospitals. In other studies, the neural networks have shown the ability to tag the entities of the text regardless of the misspellings and problems in the annotation. Transfer learning through training the word embeddings on larger corpus medical text is also a feature that can be beneficial for our task. The main goal of the thesis is to evaluate the approaches on Estonian medical data, see how high accurate these models are, and if the theoretical advantages also realize in the practice.

There is no state of art neural network architecture that works the best with every dataset, so it is required to try different architectures from the literature to find out the best model to use on our dataset. Also, it's interesting to note that the "best results" vary from one case to another. In the results section, several ways of evaluation for the neural networks was used. The first way of evaluation that was used is the quantitative evaluation of the model, which is the confusion matrix and its associated metrics. Secondly, the qualitative results are presented by manually investigating the wrongly tagged words of the model results, which helped to know if the model could detect the problems of the dataset annotation or not. Finally, by using the interpretability methods to understand the behavior of the model and to decide whether to rely on and trust the model or not. In the experiments, first, we started with a baseline model of characters tagging which had a target to predict characters tag, then we moved to use character embeddings and word embeddings along with a bidirectional LSTM layer, after that we tried using convolutional layers to represent the input text along with BI-LSTM, and, finally, we tried using ELMO model to do the embeddings for the text.

4.2 Dataset description

The dataset that was used for this thesis is from the Estonian Genome Bank [2] which has the patients medical records from the Estonian Health Information System. It consists of free texts from the discharge reports that have been processed by STACC. The texts have been processed using EstNLTK [OPT+16], which was also used to extract common medical facts from the free text. The fact extraction was done using hand-curated sets of

[2] https://genomics.ut.ee/en/about-us/estonian-genome-centre

regular expressions for each particular fact type. This set of curated facts provides us an opportunity for training and evaluating neural network-based fact extraction techniques.

Figure 5 is a snapshot from the source dataset annotation before preparing it for the NER task which has a text column which contains the patient report, each report has some text and some readings, Each cell of the dataset contains one object with its start and end position in the report text. Later on, these positions were used to give labels for the word whether it is a target object or a regular text

text_ID	start	end	object	value	unit	min	max	text
46809	0	6	RR	130				RR 130/80 mmHg.
47482	0	5	RR	142				RR142/89mmHg, HR 76 x min, p167cm, k89kg. EKG-s siinu
53571	0	6	RR	140				RR 140/74 mm Hg Cor fr. 62 UH uuringul kaela arteritest es
108715	0	6	RR	130				RR 130/80 mmHg,fe reg 66 x min.
151969	12	18	RR	130				Kaal-103kg, RR-130/80mmHg, veresukur-8,9mmol/l 3 tund
150846	25	31	RR	130				Cor toonid regulaarsed . RR 130/ 80 . Turseid jalgadel. EKG t
268089	0	6	RR	155				RR 155/85mmhg. Tupp vaba, em.kael puhas,emakas vÃ¤ike
356968	0	5	RR	123				RR123/79mmHg, HR 69 x min, p178cm, k98kg,
356953	0	6	RR	120				RR 120/90 mmHg.fr.51x/min.ebaregulaarne,SpO2 98%,kaal
385257	0	6	RR	110				RR-110/60mmHg, cor-regulaaene , 72frekvents, jikpnÃ¤Ã¤r
547468	35	41	RR	125				SÃ¼dametegevus regulaarne,fr.77/min.RR-125/78 mm Hg,V
608377	0	8	RR	127				RR 127/92 fr 57, IPSS 5 RR 143/99 , 133/100 fr 69
709298	0	6	RR	120				RR 120/70; pulss 64 xÂ´: cor, pulm ii EKG: siinusrÃ¼tm
769781	33	39	RR	105				ole. Ãœ-98, F-31, ees pea, KTG norm. UHD- peaseis, BP-72,7
820695	6	13	RR	140				eemaldamiseks.

Figure 5. Original Dataset snapshot

4.2.1 General description

The dataset used in this thesis contains Medical Reports about patients, The reports contain blood measurements, information about patients' lab analysis results such as RR, FR, INR, and many other measurements. The original dataset is huge but we took a subset of it, which is composed of 13643 reports with 27741 unique vocabulary words. This subset contains the reports with a 150 maximum number of characters, also the subset contains 3 types of objects Pikkus, RR, and egFR, while the initial dataset comprises 2125 objects. those three objects were selected to represent different types of measurements. The eGFR object is a blood measurement that is often shown together with a range of acceptable values. Measurement "pikkus" has 2 meanings in the Estonian language, one of them is the height and the other is the length. That is why it has different measurement units in Table 1. While the RR is a blood pressure measurement that has a different representation than the others. Here are some examples for the 3 chosen objects for the training dataset: eGFR examples are shown in Table 2, Pikkus and RR examples

are shown in Table 1 for the test set, another object was selected which is FR, the FR examples are shown in Table 3

Table 1. RR and Pikkus Examples

Object	text	Object	text
RR	RR 140/74 mm Hg	Pikkus	Pikkus 180cm
RR	RR114/85 mmHg	Pikkus	Pikkus: 1.78
RR	RR 110-120/60-70 mmHg	Pikkus	pikkus 10-20 mm
RR	RR145/86 mm/Hg	Pikkus	pikkus <10 mm
RR	RR: 115/65	Pikkus	pikkus 15
RR	RR 110/70 - 102/80	Pikkus	Pikkus 50cm

Table 2. eGFR Examples

Object	text
eGFR	eGFR 98 ($90mL/min/1.73m2$)
eGFR	(eGFR) 72 ($\>90mL/min/1.73m2$)
eGFR	eGFR 82 ($> 90mL/min/1.73m2$)

Table 3. FR examples

Object	text
FR	fr 60
FR	fr 75 x min
FR	fr 83 x’
FR	fr 69xÂ
FR	fr 70/min
FR	fr 82 x/min

From the examples, It can be inferred that the representation of one object has many ways of writing it. Misspellings in the text are also common, as the text is often typed by doctors not entered through structured text entry systems. Finally, the textual representations can be ambiguous, where the true meaning of the text can be inferred only when taking the context of the text into account. All these reasons make it challenging to achieve high precision fact extractions using hand-curated regular expressions.

Here are the statistics about the dataset sentences. Treating the sentences independently from the report brings a better generalization capability, sentences from different reports were mixed up and gathered into the same learning batches. This dataset is split into a training set 60%, a test set 40 % where 10% of them are for the validation set.

4.2.2 Data annotation quality

Obtaining a sufficient amount of annotated data is the greatest obstacle to the successful application of deep learning. The number of documents needed depends on many parameters, mainly on the complexity of the problem and the quality of the annotations. In our case, the annotation has problems, as many examples have a misplaced location for the objects, and also the dataset has many misspelling and some words are merged with no spaces between them, which makes the model building challenging.

4.2.3 Dataset preparation

The processed dataset from STACC has the object and values locations, so based on those locations the data was prepared for the neural network. Using words splitting based on spaces, then giving a tag for a word as an object if it's the location was included in the range of the regex. The dataset after preparation can be shown in Figure 6.

text_ID	word	tag
46809	RR	object
46809	130/80	object
46809	mmHg.	text
47482	RR142/89mmHg,	object
47482	HR	text
47482	76	text

Figure 6. The dataset after preparation for the NER task

The dataset labeling was not ideal since the regex positions were not accurate to 100%, and some of the words were connected without space. The model was trained over the dataset, including those problems, such that later on, the model can detect the objects from the text even though they have a misspelling. The dataset used for the baseline model is the sentences that have a length of fewer than 150 characters, that is is 16764 objects and 13643 reports. Where the number of words is 27741.

4.2.4 Dataset Preprocessing

The sentences that have a length smaller than 150 characters per sentence were selected because the dataset has extremely large reports with a small number of targets, so it is preferred to train the model over a smaller dataset with a better balance between targets

and the rest of the text. Moreover, the too-long sentences have many other challenges such as the padding of the tensors and the learning. The statistics of the number of characters of the original dataset is as shown in Figure 7

count	185888.000000
mean	2267.345504
std	4234.895677
min	4.000000
25%	431.000000
50%	1058.000000
75%	2443.000000
max	102602.000000

Figure 7. Dataset length statistics

4.2.5 Tagging Scheme

The NER task aims at predicting an entity type for each word in the sentence. The output tags of the dataset were annotated based on the objects and values extracted using the regular expressions. The locations of the objects and values are presented in the dataset, which was used to tag the words that are objects and the other words were tagged as a text. Then to have an equal length sentence in our datasets, a new tag was introduced as padding, which was post added to the sentences to fill the empty words tags. The dataset structure after tagging and preparation is represented in Figure 8.

```
Kilpnääre: (text) VAEVU (text) PALPEERITAV. (text) Kardiovaskulaarsü
steem: (text) COR-REGUL. (text) RÜYM, (text) RR-140/80 (object) RR
(object) 149/89 (object) mmHg, (text) p (text) 98x`. (text) RR (obje
ct) 130/70 (object) mmhg, (text) kaal (text) 61,3 (text) kg (text) *
(text) Nahk (text) ja (text) limaskestad, (text) karvad, (text) küün
ed (text) varbaküüne (text) seenhaigus. (text)
```

Figure 8. Sentence tagging

4.2.6 New dataset preparation for training over a pretrained model

The target of this new dataset is to train it over a pretrained model as was done in Experiment 7. Another object was extracted from the reports and annotated it the same

way as the dataset that was used for training the model. The new object extracted is the FR object, which has some examples presented in Table 3. The extracted dataset has 9447 unique words, 4072 sentences. The dataset was split into 80 % test set and 20% training set, and then loaded the model with character embedding and trained our new dataset over the pre-trained model, using a validation split 0.1, the number of sentences used in testing is 3258 and 814 for training which got split to 732 training sample and 82 validation sample.

4.3 Experiments

4.3.1 Common Layers

Time Distributed layer The time distributed layer is essential with the sequential data as it can apply the same function for each word in the input sequence. It generates an output per input, where each time-distributed layer should share the same weights. The Time Distributed layer saves time because if there are 5 tokens in the sentence, the weights of those tokens will be refined only once and distributed to all the tokens instead of getting refined 5 times, In the experiments, LSTM layer was used along with the time distributed layer to handle the tokens in the same sequence. The goal is that was needed is to know the relation between the sentence tokens in a given time. Lastly, the time distributed layer produces an output of 1 dimension which is needed for the following layers after the LSTM.

Embedding layer The embedding layer represents the tokens of the sentence as numbers and the sequence of words is padded so that all the sentences in the training examples get the same length of vectors. This is important in text processing as it will be represented as a dense vector that needs to be of the same dimensions. Figure 9 shows how the embedding layer in Keras works. Usually, the embedding layer is used as the first layer of the model, as it is responsible for representing the inputs. The input dimension of the embedding layer is the size of the unique vocab that we have in the data set, and the output dimension of the embedding is the dimension of this dense representation.

Spatial Dropout It is a regularization layer that works like the regular dropout layer, except that the regular dropout layer drops the individual elements, but here it drops a 1D feature map. The dropout is intended to boost the network generalization efficiency. Additionally, It is beneficial in case of the high correlation between the feature maps, it has the ability to drop those features to leave only the independent features. [HSK+12]

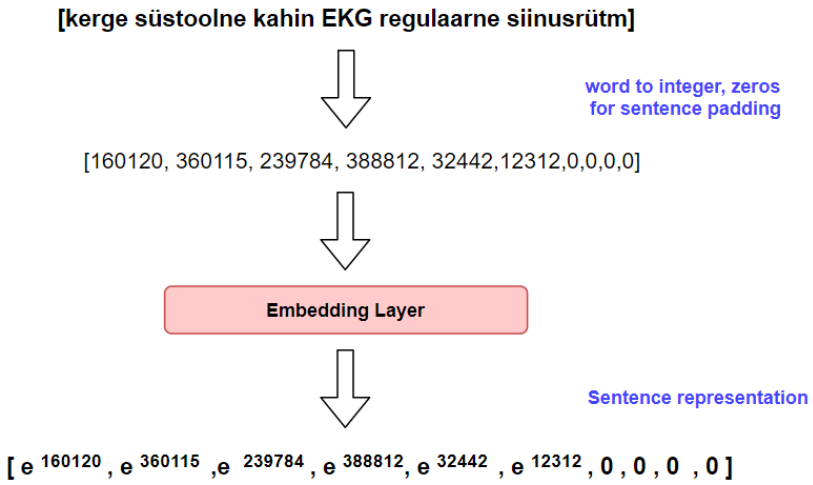

Figure 9. Embedding layer in Keras

4.3.2 Network architecture

Experiment 1: Character level architecture (Baseline Model): In this model, the input is sentences of the reports, where each sentence is represented as a sequence of characters, then it passes through a bidirectional LSTM layer to map character sequences to tag sequences, where the target is to predict a tag for each character. After the embedding layer regularization layer was used which is spatial dropout with 0.3 rate. Each character passes into the forward and backward LSTM layers and then the output of the forward and backward layers are concatenated together to produce the output which fed into the next layer. The BI-LSTM is followed by a softmax activation function which is applied to the hidden representation of the final BI-LSTM. The final layer of the model aims to get the tags of the characters.

The character embeddings that was used includes all the unique characters in the dataset tokens, while a padding tag is used to have a fixed length of all the sentences, where the small sentences will have a post-padding with a PAD tag so that it could be of the same length as the other sentences of the input. The dataset for this model was annotated character by character as shown in Figure 11, where the character has a tag as an object if it is in the location between the start and the end positions of the object token. The layers of this model are presented in Figure 12, while there is a simple representation for the task of this model presented in Figure 10, where the sentence is represented as a sequence of characters, those characters are embedded together then go through a Bi-LSTM layer, and at the end, each character gets a tag using the softmax function.

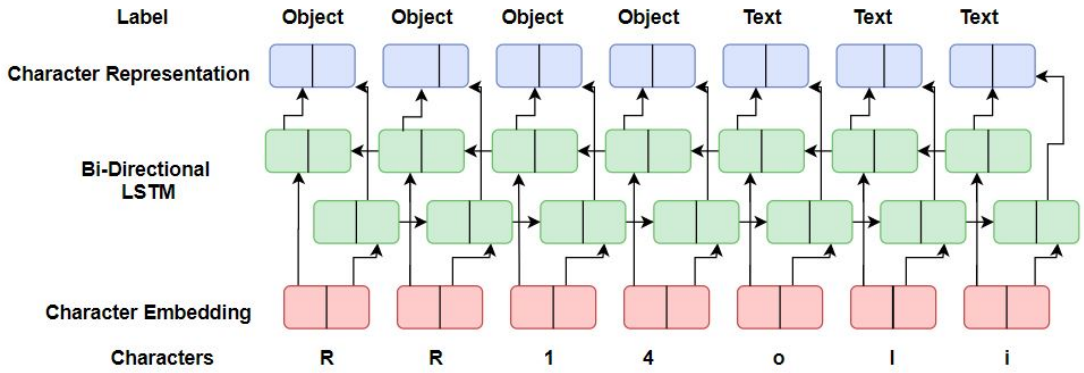

Figure 10. Experiment 1 Character Embeddings

	text_ID	char	tag
0	46809	R	object
1	46809	R	object
2	46809	1	object
3	46809	3	object
4	46809	0	object

Figure 11. Experiment 1 input dataset sample

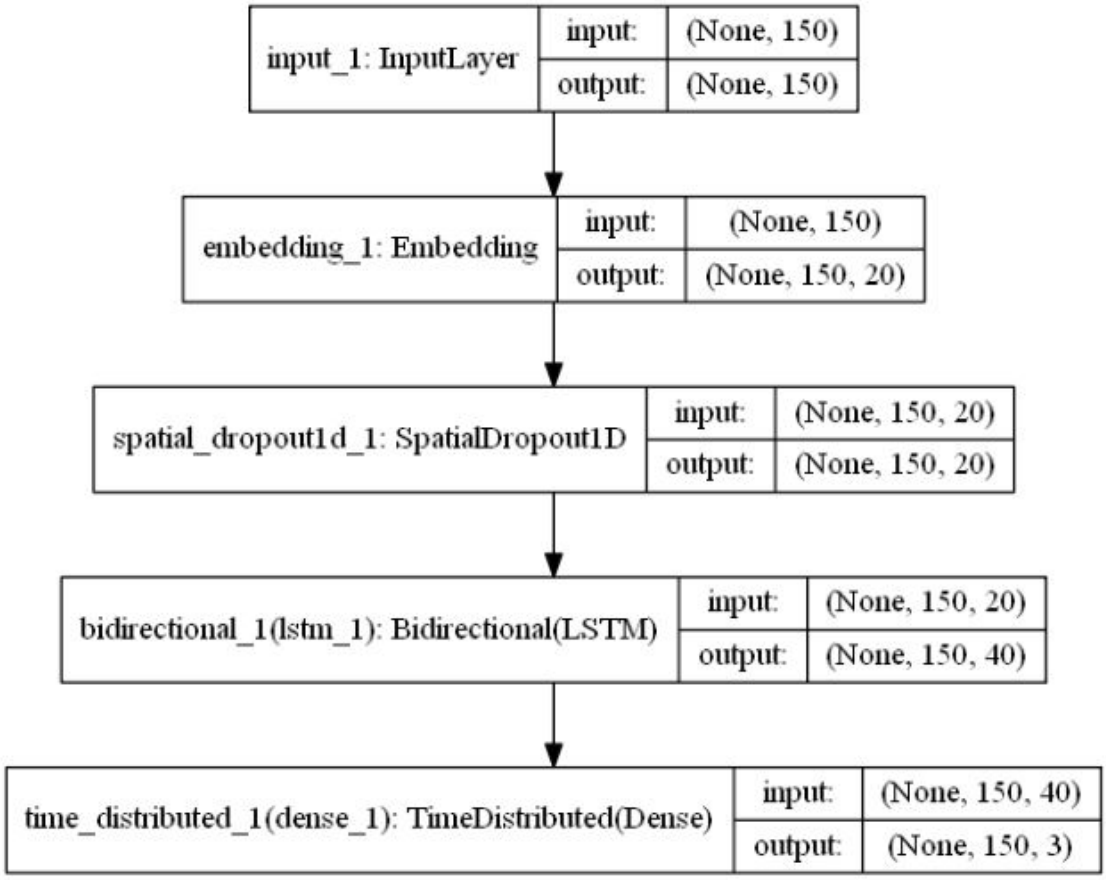

Figure 12. Experiment 1 layers

Experiment 2: Word embedding architecture In this experiment, the word embedding was applied, where the input of the model is the sentences represented by a sequence of words, the sentences of the reports are padded to have a maximum of 150 words per sentence, which is a needed step in order to have a fixed size representation of the input. Afterward, each word is represented as an index, then those indexes are converted into a dense vector of the same size utilizing Keras embedding layer. Regularization layer followed the embedding layer which is the spatial dropout of rate 0.3, then the output of the regularizer is sent to a BI-LSTM layer for sequence processing, and finally, the dense layer with a softmax activation function was applied to get the outputs of the model, which has to be one of 3 targets for each word. The configurations of this model have the same values as indicated in the next subsection. the model layers that were described above can be shown in Figure 13, while the word embedding is presented in Figure 14.

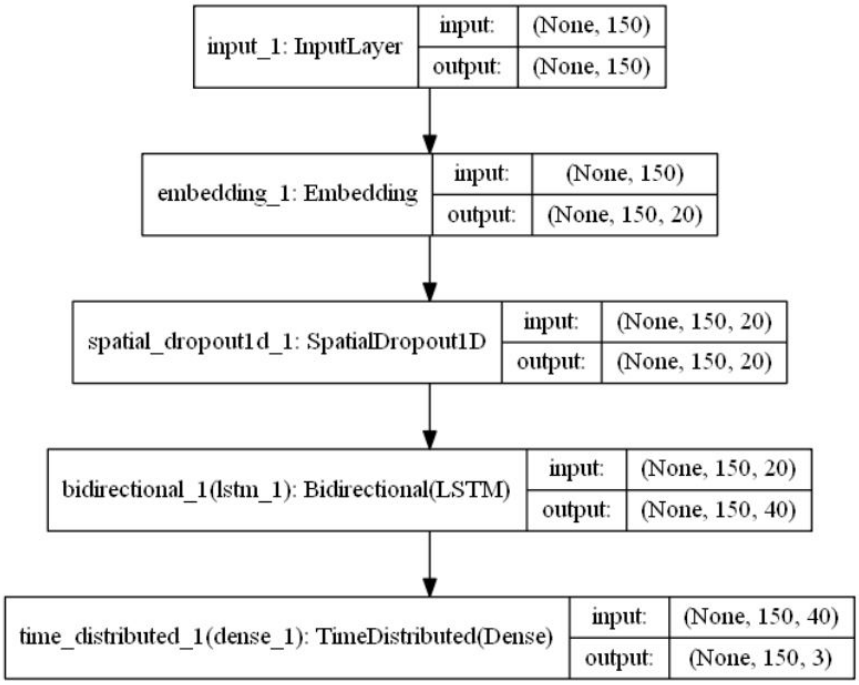

Figure 13. Experiment 2 summary

Experiment 3: Word Embedding using characters representation architecture The input for this model is the sentence words represented as a sequence of characters with a shape of (max_len, max_len_char) where maxlen represents the maximum length of words in the sentence and max_len_char represents the maximum number of characters in each word. The maximum number of words employed for this model is 150

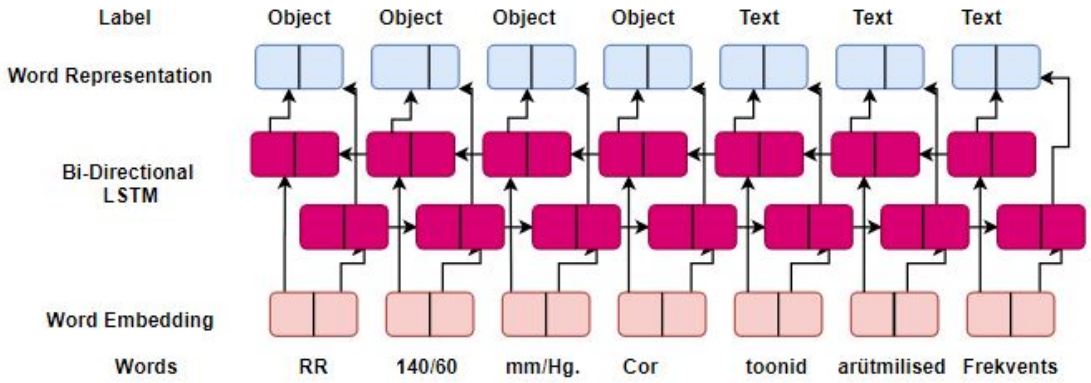

Figure 14. Schematic representation of the word embedding model.

words, the same number as the other experiments, and the maximum number of characters to be 10 charter per word, those numbers represent the average number of words per sentence and the average number of characters per word. The embeddings were used with an LSTM layer over the input. For each word, the character-level representation passes through an LSTM layer then applying some spatial dropout before passing it to the BLSTM, and finally, the softmax used to get the tag for each word. The details of the model are illustrated in Figure 15.

Experiment 4: Word Embedding concatenated with Character Embedding architecture Character-level representation of sentence words was used, along with the LSTM layer over the characters of a word to get the context information of each word. and then concatenated the character embedding with the word embedding, after that the output goes to a dropout followed by BI-LSTMs. The TimeDistributed layer was utilized to apply the same layer to every sequence of the sentence. A sequence of characters was created for every token with a maximum length of 10 characters for a word And the maximum number of words is 150 words per sentence. Finally, On top of BLSTM, a dense layer with a softmax activation function was employed to get the labels the sentence tokens, the embedding of this experiment is illustrated in Figure 16, while the layers of the model are presented in Figure 17.

Experiment 5: CNN-BILSTM-CRF architecture The input of this experiment is the sentence with a sequence of words. The neural network layer used to represent the input of the model is the convolutional neural network (CNN) layer with a kernel of size 5 which defines the size of the sliding window. While 20 filters were used for the CNN layer which indicates how many different windows we want to have. All of those filters

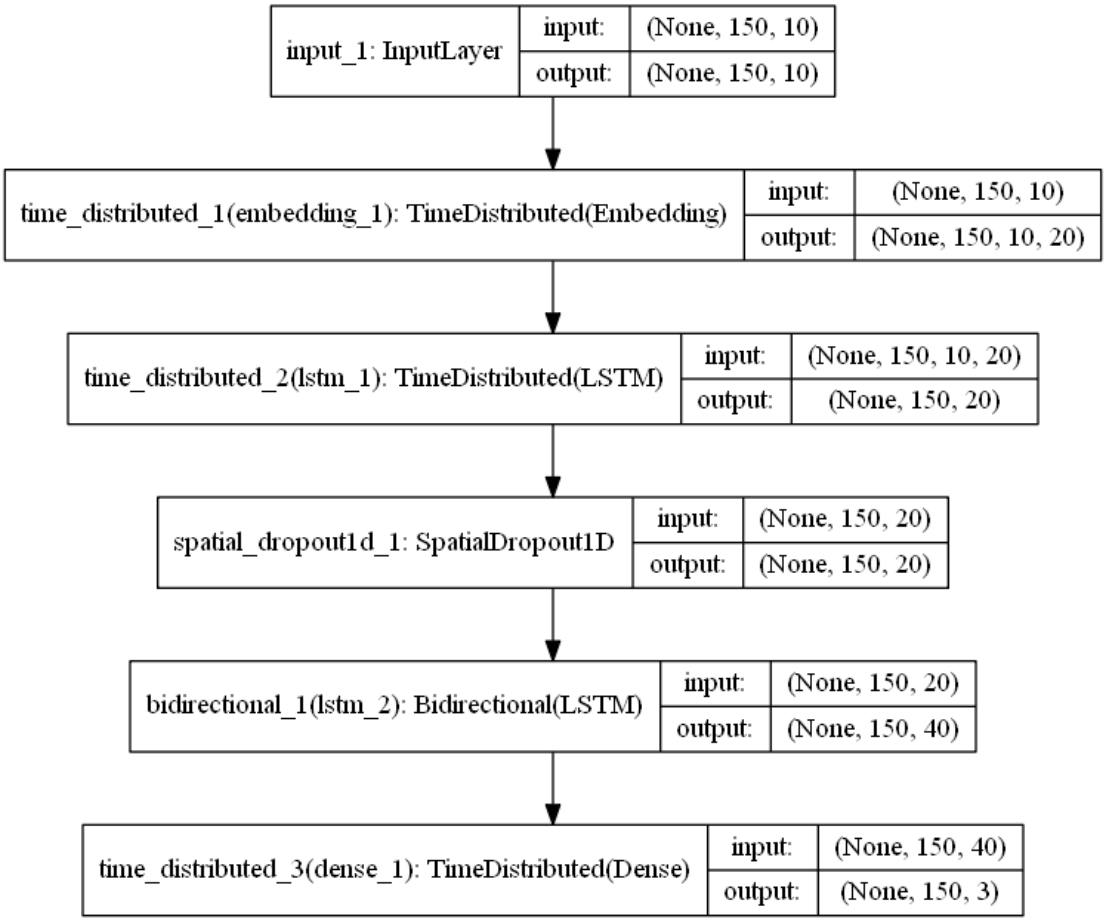

Figure 15. Experiment 3 summary

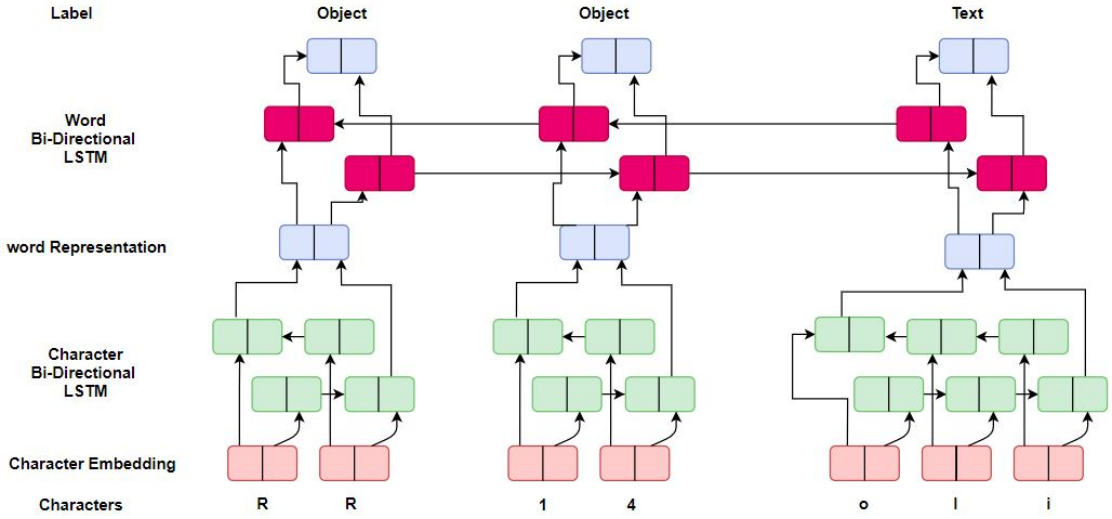

Figure 16. Experiment 4 character embedding and words representation

have the same length as indicated in the kernel_size. The result will bring 20 different convolutions. We used word-level representations that came out from the Convolution layer. Then the same amount of spatial dropout was applied, then fed into a bi-directional LSTM to get the context information of each word, After that, the output passed through a time distributed dense layer with softmax as activation function, then CRF layer was used to get the labels for the input tokens. CRF layer takes the input from the dense layer and produces a tagging score for each token, and then it selects the tag which has the highest score for each word. Figure 18 depicts the layers of this experiment.

Experiment 6: Using Elmo embeddings architecture Elmo is a contextualized word representation model. It represents the tokens of the input by using the character embedding, which is essential for the understanding of the morphological features of the text, which is usually missed by that word embeddings. It also solves the problem of the out of vocabulary words (OOV). then this character embedding representation passes through a convolution neural network layer (Conv) with some filters which allows us to know the tokens features which contribute to the better understanding of the tokens and better text representation [PNI$^+$18]. The Conv layer is followed by a max-pooling layer which summarizes the presence of the features in the input sentence. Then the output of the pooling layer passes through a Bi-LSTM layer which has 2 layers The forward layer which has the information about the token current state and the context before it. while the backward layer has the information about the token and the context after it, then the output of both layers get combined to produce the output. Finally, the output of the model was prepared, which is the dimensional vector representation of the input.

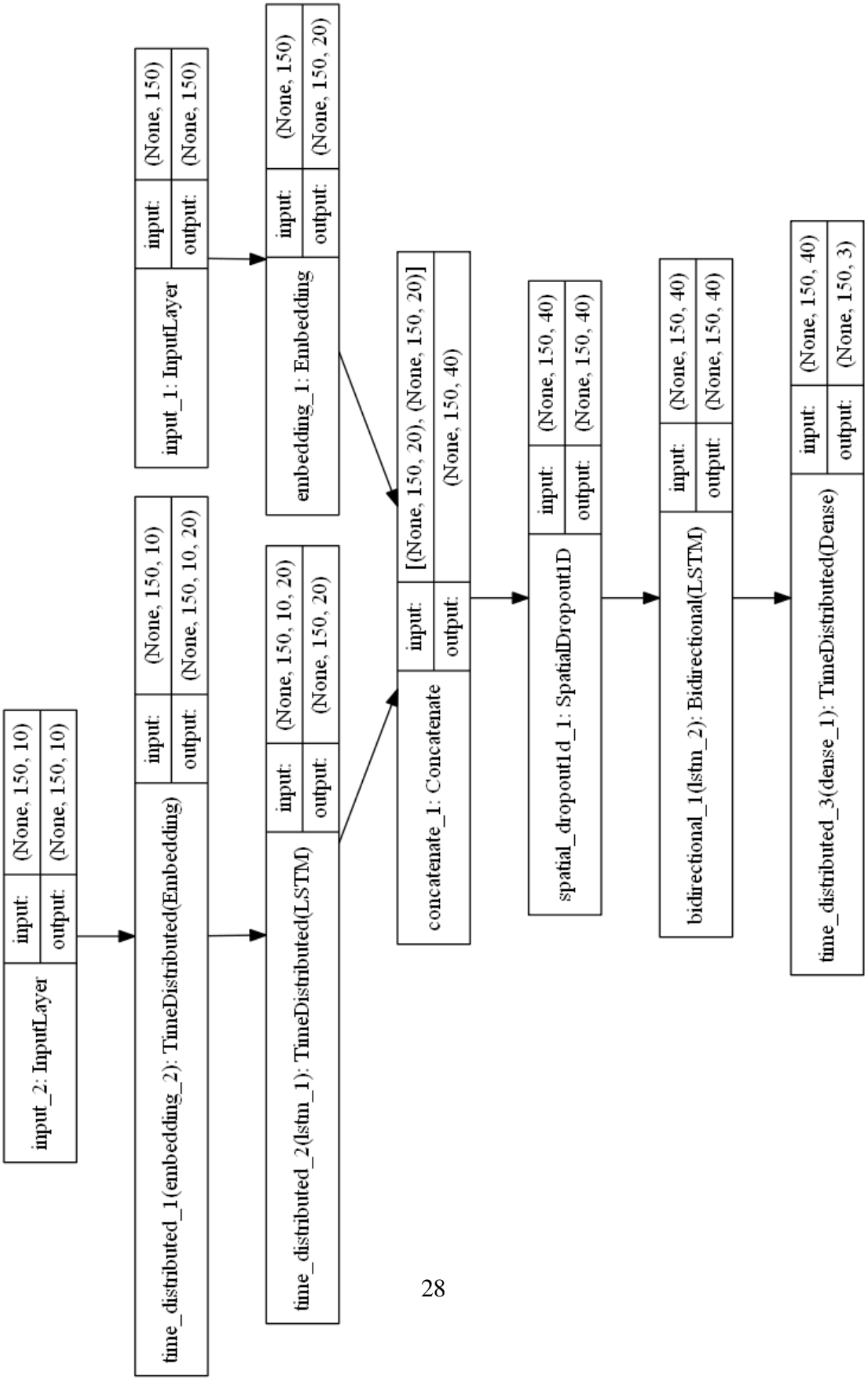

Figure 17. Experiment 4 summary

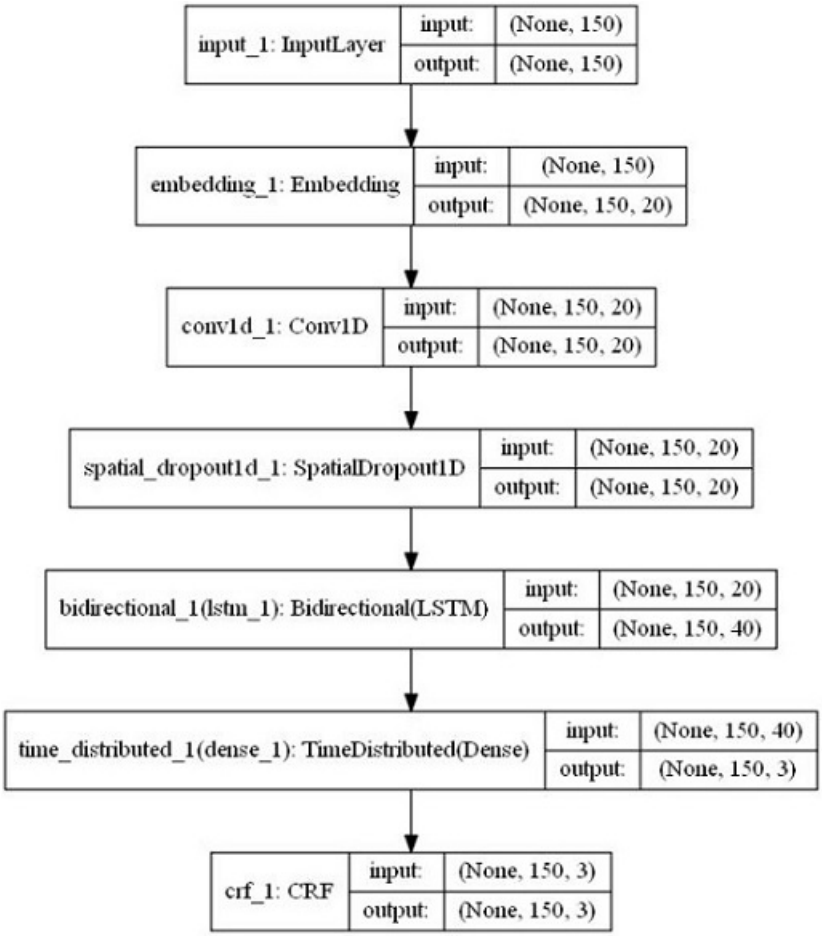

Figure 18. Experiment 5 summary

This output can be used later on to train NLP models for different tasks.

Elmo model has several already pre-trained models that can be used directly in the NLP tasks, but also Elmo's implementation allows for customization through training the model over a custom dataset. Using domain-specific word embeddings for text representation is much efficient than using a normal Estonian pre-trained word embedding and Elmo can be customized using a domain-specific dataset. We used some of the reports that we have in the original dataset, those reports are different than the ones used in our experiments, but it has the same context. The dataset we extracted for the Elmo model represents the medical context and we needed to run the Elmo model over this dataset to have domain-specific word embeddings based on Elmo model representation.

The input for the Elmo model has a special way of data preparation, the training set needs to be multiple files with 6 sentences in a file, where each sentence is on a separate line, also the same rules apply for the validation dataset. Then we needed to prepare a file with all the vocabulary which is a text file that has a word per line which includes the special tokens as UNK. The vocabulary file should be sorted in descending order by word count in the training data so that the most commonly used words are placed at the beginning of the file and the least common words are placed at the end of the file. The hyperparameters of the model have to be placed in an options file. after the preparation of those files and folders, the model starts training over them, and at the end, It produces Tf Checkpoints which are needed to be converted into an hdf5 format. This format is needed so that we can use it later for embedding the other reports. The Elmo model has a great advantage of recognizing the new tags based on the context of the word, as the words are represented differently if found in different contexts.

Configurations of the model We used 3 GPUs, over 1 node, and 450G memory.

The parameters for the Elmo model are:

- Number of tokens in Training data 85529595
- Size of Vocab 3413734
- training files 1192834
- validation files 298209

Elmo model Options: Elmo use a bidirectional LSTM layer and a char convolutional CNN layer with Relu activation function, the embedding dim is 16 with 7 filters, while the max characters per token specified are 10, and have a dropout of 0.1, the lstm dim is 4096 with 10 epochs and 32 batch_size. After getting the weights of the Elmo model, we used it along with an options file to get the embeddings of our dataset. we used the same options as the options used for Elmo pre-trained model. The dimensions of the Elmo model are 1024 which is much bigger than the dimensions that we used for embeddings

in other experiments. To have a reasonable size embeddings we had to use a smaller dataset for this experiment, we used maxlen of 35 for the sentences instead of 150, which is the median length of the sentences. Finally, the network that is used for this experiment is shown in Figure 19.

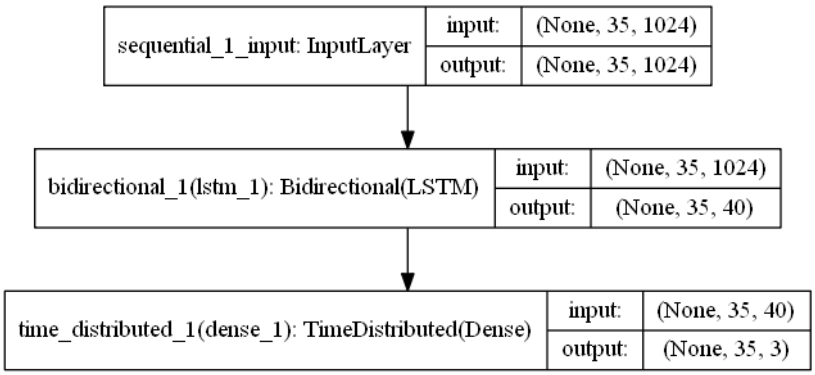

Figure 19. Experiment 6 summary

Experiment 7: Loading model and predicting a new value In this experiment, we selected a subset from the original dataset. We choose a new object called FR, we annotated this dataset subset and did the same pre-processing as we did before in our training dataset which includes using the sentences of length less than 150 characters then post-padding the smaller sentences to have a fixed length dataset. Figure 20 demonstrates the extracted dataset after tagging and preparation. We then loaded the pre-trained word embedding model which was represented by characters of the word (Experiment 3) to benefit from the transfer learning. we trained the model after loading it for our new task which is the FR object tagging. This experiment resulted in an accuracy that is very close to the pre-trained model evaluations.

Experiment 8: Applying Model 3 to a dataset that has entities as an object, value, text, PAD In this experiment we annotated the dataset differently, where the entities are split into 2 tags, one of them is the object and the other is the value. An example is demonstrated in Figure 21, while figure 22 shows the layers of the experiment. The input for this model is the sentence words represented as a sequence of characters. the shape of the input layer is (150,10), which represents the maximum number of words in the sentence and the maximum number of characters per word. Then the same layers used here as Experiment 3 architecture.

The difference here in this experiment is the tagging schema and the number of output tags, as mentioned above, the input was annotated differently, and the output for

text_ID	word	tag
51616	toonid	text
51616	rütmilised,	text
51616	fr	object
51616	72,	object
101498	rütm	text
101498	fr	object
101498	83	object
101498	ii,	text

Figure 20. Sample of FR dataset that is used for testing in Experiment 7

text_ID	word	tag
47114	70	text
47114	kg.	text
47114	Pikkus	object
47114	157	value
47114	cm.	text

Figure 21. Input dataset sample for Experiment 8

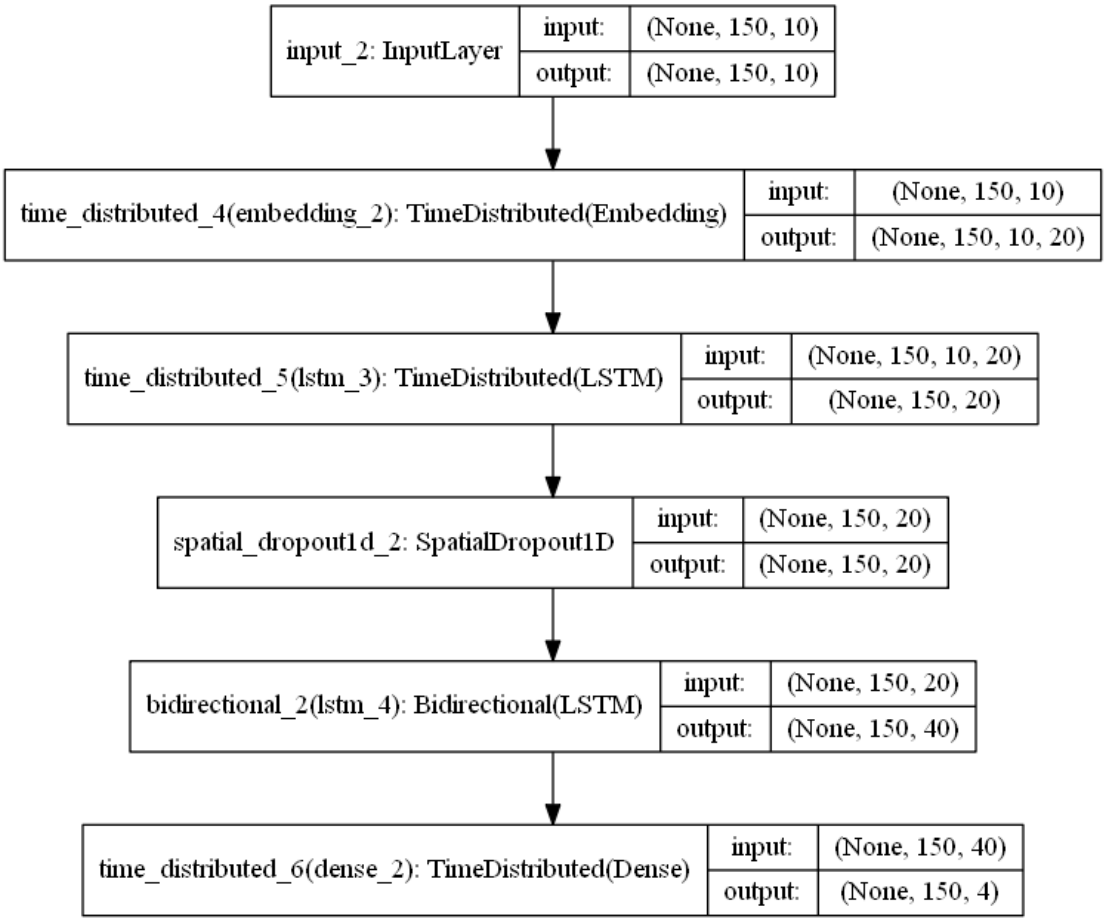

Figure 22. Experiment 8 Summary

the model has 4 tags since we added a new tag representing the values of the patients' tests.

4.3.3 Experiments settings

The neural network architectures require certain configurations and parameters to be adjusted, we used the default values most of the time, although there are several other common settings that we used in all the experiments, Table 4 indicates those common values, e.g. the spatial dropout layer was used in all the models with 0.3 rate, while the validation set only took 0.1% from the training dataset in the model fitting process. we used a recurrent dropout of rate 0.5 to drop some of the recurrent states in the linear transformation, which helps to overcome the over-fitting of the layer. And eventually, we used just one Bi-directional LSTM layer in the model architectures.

Table 4. common configurations across the experiments

Hyperparameter	Value
Batch size	32
Learning rate	0.001
Optimizer	Adam
Number of epochs	10
Validation split	0.1
Spatial dropout rate	0.3
Recurrent dropout rate	0.5
Number of BILSTM layers	1
Loss Function	sparse_categorical_crossentropy

Loss function We used Sparse_categorical_crossentropy since we have more than one label class, and our target is a sequence of integers and we don't want to change them to categorical representation. the old-fashioned way of dealing with the multi-label class was representing the labels as one-hot vectors.

Evaluation Metrics the evaluation metrics that we used are the precision, recall, and F1-score. The precision represents the percentage of the selected items that are correct while the recall represents the percentage of the correct items that were selected. Using the F1 measure acts as the trade-off between the precision and the recall taking the weighted harmonic mean between the 2 values, which is usually a balanced way to evaluate the classes of the model [SJS06]. The assessment of the named entity recognition has 2 methods to do it, one of which is the token-level approach while the

other is the evaluation for entities. sklearn_crfsuite library [3] implements the token level evaluation, As we have a multi-class, the confusion matrix is relevant to measure the model's performance. We used the classification report function of sequeval library metrics [4] for composed tags evaluation. while the Flat classification report function of sklearn_crfsuite library metrics was used for individual tags evaluation. The support of the confusion matrix represents the number of occurrences of each class in the true test set y_true. The entity-level evaluation aims to evaluate how the model predicts the whole entity not just the separate words. For example, if the entity is composed of 2 words, then the entity evaluation will take into account the predicted value of both words in the evaluation so that both of the words has to be predicted correctly to be evaluated as a TP, and if both words were predicted wrongly then it would be considered as FN.

If one of the tokens is wrong and the other is correct then it will have 2 evaluations one FN and FP will be added into account. to get the results of the evaluation matrix, we need to take the average of the results, there are 2 types can be considered, The macro-average evaluation which works separately for each class treating all of them equally and then it takes the average of all values of the system on the different tags. While the other averaging technique is the micro-average evaluation which combines the evaluation of all classes to get the average measure. it adds up the individual TP, TN FP, and FN of the different tags and uses them to get the average metric. The micro average evaluation technique is recommended in the multi-class classification tasks because it shows the proper results of the imbalanced datasets. The micro, macro, weighted average results of our models are presented in the GitHub repository mentioned in Appendix (page 48), where all the implementations and results of the experiments exist.

Optimizer We used Adam [KB14] optimizer for the model training using its default hyper-parameters in Keras. Adam is an adaptive optimization algorithm that is a development of the stochastic gradient descent which is used to update the network weights during the training process. It is named adaptive since it uses gradients estimations to improve the learning rate of the network weights. we choose the batch size of 32 sentences so that the sentences get updated each batch size.

5 Results

In this section we will present the quantitative and qualitative results of the experiments, we used the same configuration, settings, and setup in the experiments to allow a reasonable comparison between them.

[3] https://sklearn-crfsuite.readthedocs.io

[4] https://github.com/chakki-works/seqeval

5.1 Quantitative Results

Results table of objects (Individual Tags evaluation) The assessment of this table was based on the individual tags, where each token was evaluated independently from the whole entity. You can see this in Table 5. The architecture that used Elmo Embeddings (Model 6), has the highest precision, recall, and F1 among all models, which means the model has been perfectly trained to properly detect the objects. This is a great system to use when a properly annotated dataset is accessible. The results of model 3 has proved it's out-performance to model 2. We can infer that for our dataset, the word embedding using character representation is better than the word embedding using word indexing. The recall, also known as true positive rate, is found to have the best values for model 5 after Elmo model, which means that the CNN and the CRF helped the network to be able to detect the actual positive objects, this architecture can be used if we have a high cost for the false-negative objects. Model 3 has the highest precision after the Elmo model which indicates that the random variations of the model results are small compared to other models. with regard to the F1 score, we may notice that Model 2, 3, and 5 have similar ratings, implying that the average precision and recall of these models are quite similar to each other. Lastly, It is clear to see that Experiment 7 has findings that are very close to the pre-trained model (Model 3), It is an impressive finding, which indicates that the model has continued to learn over a limited new training sample and is still capable of generating successful test outcomes.

Table 5. Individual tags evaluation of the tagged object

Architecture	Precision	Recall	F1
Character embedding to target Chars + BILSTM (Model 1)	0.92	0.83	0.87
Word Embeddings (Model 2)	0.95	0.88	0.91
Word Embedding using character representation (Model 3)	0.97	0.89	0.93
Character + word embeddings +BILSTM (Model 4)	0.82	0.81	0.81
CNN + BILSTM + CRF (Model 5)	0.91	0.93	0.92
ElmoEmbedding(Customized + BILSTM) (Model 6)	0.99	0.99	0.99
FR testset using Model 3 (Experiment 7)	0.96	0.89	0.93

Results table of objects (composed tags evaluation) The evaluation of this table was based on the compound tags, where the whole entity is considered, which implies that the emphasis of this evaluation is not the specific tokens but the whole entity which can be composed of 1 token or more. In our situation, we are assessing the named entity recognition task, so the composed tags evaluation is essential to us. The list of assessments of the composite tags is provided in Table 6. Elmo model (Model 6) has also the highest precision, recall, and F1 score in the composed tags evaluation. while

Model 1 has the very lowest measures, which is expected since the entity of model 1 can be composed of 2 words which can be of length 20 characters, so the evaluation here needs to consider the results of those 20 characters, which is a harder task with lower probability of success than considering only 2 tokens. The F-score of Model 3, and Model 5 have decent and similar values to each other, suggesting that they have a reasonable overall precision and recall.

The evaluation of the composed tags of Experiment 7 is also close to the pre-trained model (Model 3) which was predicted, during the experiment we could also work to improve the metrics of the new data set by fine-tuning the pre-trained model hyper-parameters and adding some more layers [PRS19]. The strong recall of model 4 means a low false-negative rate while the poor precision implies a high false-positive rate which indicates that the model returns a lot of false positives which might not be too terrible though if a false positive cost is cheap.

Table 6. Composed tags evaluation of the tagged object

Architecture	Precision	Recall	F1
Character embedding to target Chars + BILSTM (Model 1)	0.37	0.41	0.39
Word Embeddings (Model 2)	0.88	0.75	0.81
word Embedding using character representation (Model 3)	0.95	0.83	0.88
Character + word embeddings +BILSTM (Model 4)	0.09	0.95	0.16
CNN + BILSTM + CRF (Model 5)	0.85	0.87	0.86
ElmoEmbedding(Customized + BILSTM) (Model 6)	0.99	0.98	0.98
FR testset using Model 3 (Experiment 7)	0.92	0.80	0.86

Model 8 Results: The quantitative results of the individual tags evaluation of the value and the object tags were poor. The evaluation of the individual tags is presented in Table 7, whereas the evaluation of the composed tags is demonstrated in Table 8. The individual tags evaluation of Model 8 findings is the lowest of all the studies as they have a very low precision that may be attributed to the unbalanced dataset that we have, where the percentage of the objects and values is far lower than the text in the dataset. Furthermore, because the tags were separated into objects and values, this could be an explanation for inaccuracies, since the value tag reflects a number, and we have several other numbers in our dataset that could be confusing for the model to differentiate which of them is a value and which of them is a normal text. Whilst The evaluation results of the composed tags performed stronger in the object tag than the value tag. This makes sense, as the object tag reflects the interpreted names of measurement tests as RR, Pikkus, eGFR which most of the time are written correctly without misspellings and often have the same pattern, whereas the value tag varies from one object to another, which makes it more challenging for the model to identify their values.

Table 7. Individual tags evaluation of Model 8

	Precision	Recall	F1
object	0.14	0.93	0.24
value	0.06	0.91	0.12

Table 8. composed tags evaluation of Model 8

	Precision	Recall	F1
object	0.95	0.89	0.92
value	0.88	0.82	0.85

5.2 Qualitative understanding of the models

There are several methods for understanding the neural networks, one of them is mapping the predicted class to the actual values and check whether they make sense of not [MSM18]. Investigating the words that were predicted by Model 8, we observed that the tags that were labeled incorrect were simply wrong, meaning that the model did identify the issues of the regular expressions, and also mislabeled the correct objects. Those incorrectly labeled tokens are presented in Figure 23.

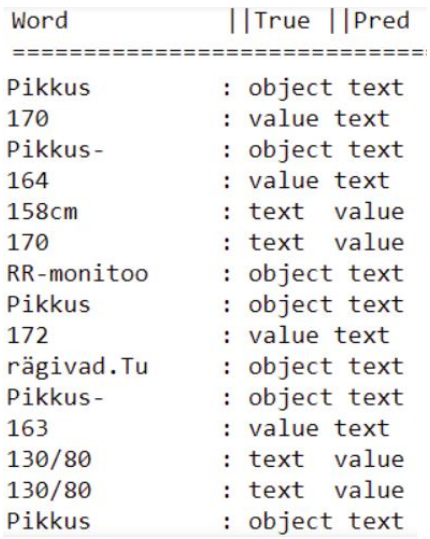

Figure 23. Model 8 incorrectly labeled tokens

Interpretability of the models We used Lime TextExplainer for interpreting our models. To use it we need to rephrase our task as a multi-class classification task, where we

create an instance of the text explainer and fit it to our document or text that we want to know its explanation. It works by generating a fake text similar to the document example by replacing some words with UNK word, and then train a classifier that is a white box to predict the output of the black box classifier which is a system where the internal workings are hidden as the CNN, RNN models. Then try to explain the original example through parameters of the white-box model. We then select the word that we want to explain it's prediction.

After the word embedding-based model of experiment 2 was trained, we used the TextExplainer class from the lime library so that we could understand it's decisions. The NER task needed to be reshaped to fit the model requirements, we treated the NER task as a text classification task. We configured the TextExplainer with the same configurations of our model, same inputs, same tags, same padding sequence. Then the TextExplainer needs a masking text sampler, where the sentence that we want to know why it's values were predicted in this way needs to have some more sentences having similar words like it. So that the TextExplainer tries to learn from those sentences and predict based on this sample. The original sentence that we chose for the interpretation is shown in Figure 24, while the sampler example that was created, with a maximum replacement of 0.7 of the original sentence can be shown in Figure 25, the number of sample sentences generated is 5, having at the end the percentage that indicates how much of the original sentence was substituted with unknowns tokens. We have to choose a particular word for the model to interpret it. For illustration, in this Figure 26, we picked "120/80" as our predictive token index, the figure shows the probability that this token is an entity. It also displays the features that contributed favorably to the model prediction decision, while the intensity of the color demonstrates how strongly the token contributed to the prediction.

```
Üldseisund: rahuldav 156 cm,90 kg Kardiovaskulaarsüsteem: RR 140/80
mmHg,fr reg 72 x min RR130/100 mmHg. Obj. liigesturset ei ole, k.a.
MTP-des. RR 160/80 mmHg, kaal 131,1 kg, eutüreoidne, kilpnääre palp
tundub veidi tihedam normist, võimalik nodoosne.
```

Figure 24. Sentence for interpretability

Another indication of interpretability was given for another token with wordindex=7: RR has seen in Figure 27, This figure illustrates the features that supported the decision of the model, one of those tokens is the "Kardiovaskulaarsüsteem" which is an Estonian term means the cardiovascular system, which is found to be a common word for predicting the RR as an object, which is fair since the RR is a blood pressure reading which is related to the cardiovascular system measures. It is also noticed that the predictions of the model are confident with high probability as mentioned, It classified the RR to be an object with 0.99 confidence.

```
(('Üldseisund: rahuldav 156 cm,90 kg Kardiovaskulaarsüsteem: RR 140/
80 mmHg,UNK reg 72 UNK min RR130/100 mmHg. Obj. liigesturset ei ole,
k.a. MTP-des. RR 160/80 mmHg, kaal UNK,1 kg, eutüreoidne, kilpnääre
palp.tundub veidi UNK normist, võimalik nodoosne.', 'Üldseisund: UNK
UNK cm,90 kg Kardiovaskulaarsüsteem: RR 140/80 UNK,fr UNK 72 x min U
NK/100 mmHg. Obj. UNK ei ole, k.UNK. UNK-des. RR UNK/UNK mmHg, kaal
131,1 kg, UNK, UNK palp.tundub veidi tihedam UNK, võimalik nodoosn
e.', 'Üldseisund: rahuldav 156 cm,90 kg Kardiovaskulaarsüsteem: RR 1
40/80 mmHg,fr UNK 72 x UNK RR130/100 UNK. UNK. liigesturset ei ole,
UNK.a. MTP-des. RR 160/80 mmHg, kaal 131,1 kg, eutüreoidne, kilpnäär
e palp.tundub veidi UNK normist, võimalik nodoosne.', 'UNK: rahuldav
UNK cm,90 kg Kardiovaskulaarsüsteem: UNK UNK/UNK mmHg,fr reg 72 x mi
n RR130/UNK UNK. Obj. UNK ei UNK, UNK.UNK. MTP-UNK. UNK 160/80 UNK,
kaal 131,UNK kg, eutüreoidne, UNK palp.UNK UNK UNK UNK, võimalik nod
oosne.', 'Üldseisund: UNK UNK cm,90 UNK UNK: UNK UNK/UNK mmHg,UNK re
g UNK UNK UNK UNK/UNK mmHg. Obj. UNK UNK ole, UNK.UNK. UNK-UNK. RR U
NK/80 UNK, UNK 131,UNK UNK, UNK, UNK UNK.UNK veidi UNK normist, UNK
nodoosne.'), array([0.95346257, 0.83937204, 0.92932035, 0.73854892,
0.56407605]))
```


Figure 25. Sample sentences for interpretability

Figure 26. Interpretability of 120/80 object

The configurations of the white box classifier used in this experiment are presented in Figure 28, which is a probabilistic white-box classifier with the default algorithm for classification which is a logistic regression model. By default, it trains with stochastic gradient descent and uses elasticnet for regularization. This is an effective regularizer since it combines L1 and L2 regularizations. The model has some hyper-parameters that were the default parameters selected by the eli5 lime library. [5] library.

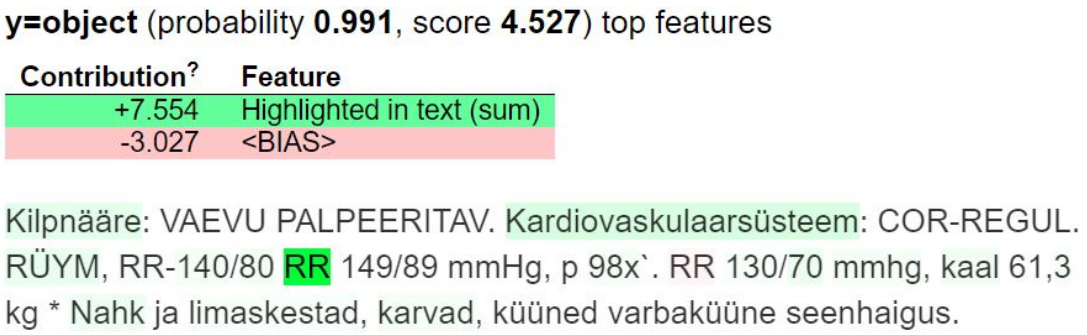

Figure 27. interpretability of wordindex 7 (RR)

```
SGDClassifier(alpha=0.001, average=False, class_weight=None,
              early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
              l1_ratio=0.15, learning_rate='optimal', loss='log', max_iter=1000,
              n_iter_no_change=5, n_jobs=None, penalty='elasticnet',
              power_t=0.5, random_state=RandomState(MT19937) at 0x22B710E7D08,
              shuffle=True, tol=0.001, validation_fraction=0.1, verbose=0,
              warm_start=False)
```

Figure 28. Interpretability model parameters

Finally, Figure 29 shows the detailed analysis of the prediction decision of word "RR", It shows the tokens that contributed to the prediction of each class so that we can understand the reason behind the decision of the model. The original table is much longer than the screenshot, but for the quality, we had to crop some of it. The idea of this table is to show with green color the tokens that contributed positively to the decision of the model, while the red color represents the bias tokens for the prediction. The figure also shows the likelihood of this token to be classified in each class. Here in this example, the model classified RR correctly as an object with almost full trust.

[5] https://eli5.readthedocs.io/en/latest/autodocs/lime.html

y=PAD (probability **0.000**, score **-19.579**) top features		y=text (probability **0.001**, score **-6.689**) top features		y=object (probability **0.999**, score **6.759**) top features	
Contribution[?]	Feature	Contribution[?]	Feature	Contribution[?]	Feature
-0.258	fr	+1.267	kardiovaskulaarsüsteem	+2.615	rr 140
-0.308	kilpnääre	+1.101	<BIAS>	+2.200	rahuldav 156
-0.326	kardiovaskulaarsüsteem	+0.951	80	+1.649	90 kg
-0.328	90	+0.933	90	+1.412	kg kardiovaskulaarsüsteem
-0.338	mtp	+0.924	72	+1.117	üldseisund rahuldav
-0.341	x	+0.808	reg	+1.036	156 cm
-0.343	palp	+0.789	x	+1.018	kardiovaskulaarsüsteem rr
-0.344	obj	+0.750	mmhg	+0.923	rr130
-0.352	min	+0.658	üldseisund	+0.818	140 80
-0.359	võimalik	+0.566	veidi	+0.787	reg 72
-0.374	k	+0.531	cm	+0.642	liigesturset ei
-0.382	160	+0.514	min rr130	+0.573	kg
-0.385	ole	+0.514	liigesturset	+0.474	tundub veidi
-0.385	140	+0.460	obj	+0.465	x min
-0.386	<BIAS>	+0.449	ei	+0.412	mmhg kaal
-0.391	üldseisund	+0.424	k	+0.393	võimalik nodoosne
-0.392	eutüreoidne	+0.422	des	+0.357	mmhg fr
-0.398	veidi	+0.401	normist	+0.354	normist võimalik
-0.402	liigesturset	+0.350	131	+0.333	kg eutüreoidne
-0.402	ei	+0.332	võimalik	+0.328	kilpnääre palp
-0.403	156	+0.312	fr	+0.328	100 mmhg
-0.403	72	+0.285	palp	+0.314	tihedam normist
-0.405	tundub	+0.279	eutüreoidne	+0.314	131 1
-0.406	des	+0.261	kilpnääre	+0.302	obj liigesturset
-0.409	kaal	+0.252	tundub	+0.300	80 mmhg
-0.410	131	+0.250	160	+0.287	rahuldav
-0.411	1	+0.237	min	+0.279	fr reg
-0.415	a	+0.221	mtp	+0.269	ole k
-0.417	reg	+0.198	1	+0.266	156
-0.422	cm	+0.180	100	+0.262	mtp des
-0.437	100	+0.167	nodoosne	+0.248	160 80
-0.439	rr130	+0.149	tihedam	+0.246	140
-0.441	rahuldav	+0.124	kaal	+0.243	ei ole
-0.460	nodoosne	+0.100	1 kg	+0.237	k a
-0.464	normist	+0.062	a	+0.236	mmhg obj
-0.496	tihedam	+0.049	ole	+0.193	72 x
-1.141	kg	-0.046	palp tundub	+0.189	des rr
-1.165	rr	-0.048	rr 160	+0.149	rr
-1.199	80	-0.092	rr130 100	+0.113	veidi tihedam
-2.042	mmhg	-0.105	rr	+0.106	kaal 131
		-0.122	veidi tihedam	+0.079	rr130 100
		-0.123	kaal 131	+0.059	rr 160
		-0.200	des rr	+0.033	palp tundub
		-0.219	72 x	-0.031	ole
		-0.234	ei ole	-0.058	a
		-0.237	k a	-0.098	1 kg
		-0.238	mmhg obj	-0.140	tihedam
		-0.251	160 80	-0.150	kaal
		-0.252	ole k	-0.162	nodoosne
		-0.256	140	-0.185	1

Figure 29. Detailed Interpretability table of word RR

6 Discussion

The evaluation results of the models vary, some of them have high precision with a low recall and some have the opposite, In order to decide which model to use we need to define the cost for each metric in our use case. The model that used the pre-trained Elmo embedding resulted in a very good quantitative result, but it requires a significant amount of resources for the training process, and also a high degree of memory specs for embedding the dataset. Therefore, based on the specifications that we have and the accuracy that we need, we should select the best model to use.

It is advised to consider the qualitative results. In our case, we had some issues with annotation, which means that high accuracy could be an indication that the model trained to detect the exactly given objects, but it could not detect the improperly annotated tokens. The figure below 30 shows some examples of cases where the predictions between our machine learning-based model differed from the annotations in the source data. It is clear that in some of the cases our model actually made the correct call and the original annotation was wrong. This shows we can learn a useful model even if the underlying annotations are not perfect.

Finally, I would like to suggest, while training a new object using a pre-trained model, try to overcome the catastrophic forgetting by including some old examples of the training dataset as a revision for the model, so that It can recall the old objects along with the new objects. Since the NN models when they get trained for a new object they tend to concentrate on it and neglect the previous targets [Fre92].

Word	True	Pred
/106	text	object
90	text	object
ebaregulaa	object	text
KORRAS.	object	text
KOORMUSTES	object	text
Hba1c-7,0%	object	text
112/60	text	object
Analüüsid.	object	text
Amb.m/õ,te	object	text
id="0"	object	text
p/s-12,5kg	object	text
120,1	object	text
*12.03.201	object	text
Nõustamine	object	text

Figure 30. Words tagged correctly by the neural network

6.1 Future work

Most of the pre-trained word embeddings are for other languages and obviously will not help with Estonian language texts. In this thesis we trained custom ELMO models on a subset of medical texts we had available, the resulting models had a very good performance. To apply these models in practice it would be advisable to spend some effort to train ELMO embeddings on all the available Estonian language medical texts. Besides, the models built in this study can be conveniently used for numerous medical entities with specific patterns. The optimal tags for the dataset we used were making the layout of the entity separated into an object, value, unit, min, max. As potential research, we need to annotate the dataset accordingly so that those tags can be identified separately, which is feasible using the pre-trained models we prepared. and then we could extract all the objects from the predicted dataset, and build out a Fact database of all the measurements.

7 Conclusion

We developed neural networks for named entity recognition on the medical reports, we found that the neural networks can detect the complicated features of the data. Also, it is robust to misspellings and misannotation of the data. Different neural network architectures were implemented, some of them used pre-trained contextualized word embeddings, and other built word embeddings based on the input dataset. we used CNN, RNN as LSTM, BI-directional LSTMs, CRF, and dense layers with regularization functions. The evaluation of those models has resulted in a very strong quantitative and qualitative results, where the models managed to correctly account for the context when predicting features. The best quantitative results were obtained using the BI-LSTM layers combined with custom ELMO character-level word embeddings.

References

[ABV18] Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, 2018.

[C+15] François Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

[CN16] Jason PC Chiu and Eric Nichols. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 4:357–370, 2016.

[DCLT18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[DCQ+19] Xishuang Dong, Shanta Chowdhury, Lijun Qian, Xiangfang Li, Yi Guan, Jinfeng Yang, and Qiubin Yu. Deep learning for named entity recognition on chinese electronic medical records: Combining deep transfer learning with multitask bi-directional lstm rnn. *PloS one*, 14(5), 2019.

[Fre92] Robert M French. Semi-distributed representations and catastrophic forgetting in connectionist networks. *Connection Science*, 4(3-4):365–377, 1992.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[HSK+12] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[HWN+17] Maryam Habibi, Leon Weber, Mariana Neves, David Luis Wiegandt, and Ulf Leser. Deep learning with word embeddings improves biomedical named entity recognition. *Bioinformatics*, 33(14):i37–i48, 2017.

[JZS15] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *International conference on machine learning*, pages 2342–2350, 2015.

[KB14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[LG14] Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308, 2014.

[LSHL20] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[LSK08] Dingcheng Li, Guergana Savova, and Karin Kipper. Conditional random fields and support vector machines for disorder named entity recognition in clinical texts. In *Proceedings of the workshop on current trends in biomedical natural language processing*, pages 94–95, 2008.

[MKB+10] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010.

[MN+16] Sunil Mandhan, Yoshiki Niwa, et al. Numerical atrribute extraction from clinical texts. *arXiv preprint arXiv:1602.00269*, 2016.

[MSM18] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15, 2018.

[OPT+16] Siim Orasmaa, Timo Petmanson, Alexander Tkachenko, Sven Laur, and Heiki-Jaan Kaalep. Estnltk - nlp toolkit for estonian. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may 2016. European Language Resources Association (ELRA).

[PNI+18] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[PRS19] Matthew Peters, Sebastian Ruder, and Noah A Smith. To tune or not to tune? adapting pretrained representations to diverse tasks. *arXiv preprint arXiv:1903.05987*, 2019.

[RSG16] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery*

and Data Mining, San Francisco, CA, USA, August 13-17, 2016, pages 1135–1144, 2016.

[SJS06] Marina Sokolova, Nathalie Japkowicz, and Stan Szpakowicz. Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation. In *Australasian joint conference on artificial intelligence*, pages 1015–1021. Springer, 2006.

[SSN12] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*, 2012.

[WCZ19] Yile Wang, Leyang Cui, and Yue Zhang. Using dynamic embeddings to improve static embeddings. *arXiv preprint arXiv:1911.02929*, 2019.

[XZHL17] Kai Xu, Zhanfan Zhou, Tianyong Hao, and Wenyin Liu. A bidirectional lstm and conditional random fields approach to medical named entity recognition. In *International Conference on Advanced Intelligent Systems and Informatics*, pages 355–365. Springer, 2017.

[YLQ+19] Jianliang Yang, Yuenan Liu, Minghui Qian, Chenghua Guan, and Xiangfei Yuan. Information extraction from electronic medical records using multitask recurrent neural network with contextual word embedding. *Applied Sciences*, 9(18):3658, 2019.

[ZZHY15] Shu Zhang, Dequan Zheng, Xinchen Hu, and Ming Yang. Bidirectional long short-term memory networks for relation classification. In *Proceedings of the 29th Pacific Asia conference on language, information and computation*, pages 73–78, 2015.

[ZZL15] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.

Appendix

I. Source code

The implementation of the baseline language model and all the experiments is available in the GitHub repository: `https://github.com/nesmaAlmoazamy/Fact-Extraction-from-Medical-Text-using-Neural-Networks`

II. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Nesma Mahmoud**,
(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,

   Fact extraction from medical text using Neural networks,
   (title of thesis)

   supervised by Raivo Kolde.
   (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 am aware of the fact that 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.

Nesma Mahmoud
15/05/2020