

DANISH CLINICAL EVENT EXTRACTION
DEVELOPING A CLINICAL EVENT EXTRACTION
SYSTEM FOR ELECTRONIC HEALTH RECORDS USING
DEEP LEARNING AND ACTIVE LEARNING

FREDERIK WONSILD
MATHIAS GIOVANNI MØLLER

Master's thesis

Software Development (Design)

Frederik Wonsild (frwo@itu.dk)

Mathias Giovanni Møller (mgmo@itu.dk)

June 1, 2020

Supervisor: Leon Derczynski

Project code: KISPECI1SE

ABSTRACT

Danish electronic health records contain valuable information that is not fully utilized today. Given the unstructured nature of clinical texts, natural language processing techniques are required to extract relevant entities from these records. This study develops a deep learning based system capable of recognizing clinical events in Danish electronic health records. A total of 2.090 breast cancer-related sentences are annotated, comprising 42.753 words and 5.809 events. We fine-tune a pre-trained multilingual BERT model on this data, reaching an F1-score of 87.5% on recognizing event spans.

An active learning system is implemented for sentence querying to explore how the cost of annotation can be reduced. We implement two active learning strategies, *uncertainty sampling* and *diversity sampling*. We find that both strategies lead to a higher F1-score than random sampling when trained on the same amount of sentences. Specifically, we see an F1-score increase of 1.3% and 2.6% compared to random sampling, corresponding to an error reduction of 7.2% and 14.5%. However, in terms of the amount of annotated words, we find random sampling to be less costly than both active learning strategies.

We test our fine-tuned BERT model on sentences related to pneumonia in order to establish its cross-domain capabilities. We observe a 5.7 percentage point drop in F1-score, leading to a score of 81.8%.

While the benefits of active learning in terms of annotation cost remain unclear and require more research, these findings do prove the feasibility of a Danish clinical event extraction system.

CONTENTS

Abstract	i
List of Figures	iv
List of Tables	v
Acknowledgements	vi
1 INTRODUCTION	1
1.1 Problem statement	2
1.2 Research questions	2
2 BACKGROUND	3
2.1 Natural Language Processing	3
2.1.1 Named Entity Recognition	3
2.1.2 Clinical Natural Language Processing	4
2.2 Representing words and sentences	5
2.2.1 Word embeddings	5
2.2.2 Sentence embeddings	6
2.3 Deep learning	6
2.3.1 Artificial neural networks	6
2.3.2 Long Short-Term Memory	8
2.3.3 Transformer	9
2.3.4 Bidirectional Encoder Representations from Transformers	11
2.4 Annotation	14
2.5 Active Learning	14
2.5.1 Query strategies	15
3 DATA AND MATERIAL	17
3.1 Electronic Health Records	17
3.2 Choosing a clinical domain	18
3.3 Unlabeled and labeled data	18
3.4 Packages and libraries	21
4 METHODS	22
4.1 Data handling	22
4.1.1 Ethical considerations	22
4.1.2 Working with sensitive data	23
4.1.3 Data pre-processing	23
4.2 Annotation process	24
4.2.1 Clinical event definition	25
4.2.2 Event attributes	25
4.2.3 Challenges of annotation	26
4.3 Model implementation	27
4.3.1 Implementing LSTM based models	27
4.3.2 Implementing BERT based models	29

4.4	Active learning implementation	30
4.4.1	Implementation of uncertainty sampling	31
4.4.2	Implementation of diversity sampling	31
4.4.3	Iterations	33
4.5	Evaluating models	34
4.6	Conducting experiments	35
4.6.1	Base model experiments	35
4.6.2	Active learning experiments	36
4.6.3	Final model experiments	36
4.6.4	Cross domain experiments	37
5	RESULTS	38
5.1	Base models	38
5.2	Active learning models	39
5.3	Final model	42
5.4	Cross-domain model	43
6	DISCUSSION	45
7	CONCLUSION	47
A	APPENDIX	54
A.1	Annotation guidelines	54
A.2	Base model results	56
A.3	Principal Component Analysis	58
A.4	Elbow method	58
A.5	Active Learning - Entities and sentences	59
A.6	Active Learning - Unique words and sentences	59
A.7	Data statement	60

LIST OF FIGURES

Figure 1	Basic representation of a recurrent neural network	7
Figure 2	Unfolded representation of a recurrent neural network	7
Figure 3	The memory block of a simple LSTM network	9
Figure 4	Example of a bidirectional network in a sequence labeling task	9
Figure 5	Example of how attention works	10
Figure 6	High level architecture of a BERT model and an encoder	12
Figure 7	BERT fine-tuned for token classification	13
Figure 8	Histogram of the sentence lengths in the initial unlabeled pool	19
Figure 9	Overview of the unlabeled pool and labeled pools	20
Figure 10	High level illustration of the active learning process	30
Figure 11	t-SNE visualization of the sentence embeddings	32
Figure 12	Silhouette values for the clusters	32
Figure 13	Percentage of all sentences for each cluster	33
Figure 14	A 5-fold cross validation setup	36
Figure 15	F1-score for each query strategy according to number of sentences trained on	40
Figure 16	Number of unique entities for each query strategy according to number of sentences trained on	41
Figure 17	Number of words for each query strategy according to number of sentences trained on	41
Figure 18	Bar chart of F1-scores (entity and surface) for breast cancer and pneumonia tests	43
Figure 19	Event intersection between breast cancer and pneumonia data sets	44

LIST OF TABLES

Table 1	First five rows of the raw data this project works with	17
Table 2	Columns of unlabeled and labeled pools	19
Table 3	Sentence, word, and event counts for the final labeled pools .	20
Table 4	Old vs. new sentence boundary disambiguation	24
Table 5	Layers in the three sequential LSTM based models	28
Table 6	Fixed hyperparameters for the LSTM based models	29
Table 7	Sample of an unlabeled pool with descending uncertainties .	31
Table 8	The unlabeled pool with sentence uncertainties, cluster as- signments and probabilities	33
Table 9	An example of OB tagging	34
Table 10	Specified parameters of the grid search	35
Table 11	The best performing models of each tested architecture	38
Table 12	Evaluation results of models based on query strategy	39
Table 13	F1-scores (entity and surface) for each query strategy with percentage increase from random sampling score	40
Table 14	Model performance on span labels	42
Table 15	Class balance for each multi-class attribute	42
Table 16	Model performance on other attributes	43

ACKNOWLEDGEMENTS

We would like to take this opportunity to thank our supervisor Leon Derczynski. We thank him for providing us his expertise and guiding us through this thesis. Last but not least, we thank him for all the talks, formal and informal alike, ranging from discussions on natural language processing to chats about rocket launches.

We also wish to thank Leonia Falck for her assistance with helping us better understand the medical domain.

1 | INTRODUCTION

Denmark has some of the best data when it comes to healthcare ([Copenhagen Healthtech Cluster, 2018](#)). Despite of this, much of the data is not utilized today ([Sørensen et al., 2016](#)) and Denmark lags behind many other countries when it comes to processing this data ([Névél et al., 2018](#)). Event extraction is a clinical natural language processing (NLP) task which aims to recognise and extract clinical events in electronic health records. In this study, clinical events are defined as any event relevant to a patient’s clinical timeline, such as surgeries, treatments, and tests. According to THYME ([THYME, nd](#)), a leading project on processing English clinical free text, extracting clinical events and understanding the timeline of these “[...] is key to the next generation of translational research where the importance of generalizing over large amounts of data holds the promise of deciphering biomedical puzzles”. The task of recognizing clinically relevant events is well studied for several languages but remains absent for Danish ([Kirkedal et al., 2019](#)). This project seeks to bridge this gap by developing an NLP system capable of recognizing clinical events in Danish electronic health records.

Electronic health records (EHR), *Elektronisk Patientjournal* in Danish, are records that store patient health information in a digital format. EHR are used by doctors for clinical decision making but do also provide a valuable source of information that can be used to improve patient care and health care systems ([Friedman et al., 1999](#)). As EHRs often are stored in form of unstructured free text, leveraging the information stored in EHRs requires sophisticated processing of natural language.

Recently, deep learning models such as long short-term memory (LSTM) ([Hochreiter and Schmidhuber, 1997](#)) and pre-trained models, such as Bidirectional Encoder Representations from Transformers (BERT) ([Devlin et al., 2018](#)), have shown great potential in clinical NLP, achieving state-of-the-art results on event extraction tasks for languages such as English, Italian, Spanish and Chinese (e.g. [Lee et al. \(2020\)](#); [Viani et al. \(2019\)](#); [Hakala and Pyysalo \(2019\)](#); [Zhang et al. \(2019\)](#)). These models however, require a substantial amount of labeled data to perform well. Annotated clinical corpora of such size do not currently exist in Danish, and a substantial amount of clinical text must therefore be annotated to make training and tests on deep learning models feasible. However, annotation has been found to be both challenging and costly ([Velupillai et al., 2015](#)).

Active learning is a methodology in machine learning that allows the system to take an active part in data item querying. It has shown promise in making the annotation process more efficient, by improving the system performance with less data ([Chen et al., 2015](#); [Shen et al., 2017](#)). Many of these studies however, are based on simulated experiments. In the light of the promising deep learning models and the challenges of data annotation, this project furthermore aims to investigate to what extent the annotation process can be optimised through active learning.

Proving the concept that events correctly and efficiently can be extracted from Danish EHRs is a prerequisite for the long term goal of utilizing large amount of clinical data, in order to discover new patterns and long-term correlations and support health service research.

1.1 PROBLEM STATEMENT

This study aims to develop a system based on state-of-the-art deep learning technologies that is capable of extracting events from Danish clinical notes. Currently, there exists no data set for Danish clinical notes to train such a system on. Thus, this study seeks to develop a set of annotation guidelines and use these to reliably annotate some extent of the data. Since annotation is challenging and costly, this study seeks to address this concern by employing an element of active learning in the system, in order to measure its effect on annotation costs.

1.2 RESEARCH QUESTIONS

1. Which current deep learning architecture is the most promising for a clinical event extraction system?
2. To what extent can the annotation process be optimized through active learning?
3. How well does a clinical event extraction system perform when trained on a set of Danish clinical texts?
4. How is the model performance of a Danish clinical event extraction system impacted when trained on one clinical domain and tested on a different clinical domain?

2 | BACKGROUND

2.1 NATURAL LANGUAGE PROCESSING

Natural language processing is an interdisciplinary field that combines computer science, linguistics, cognitive science and artificial intelligence. The field dates back to the 1950s, where Alan Turing proposed the Turing test that evaluates a computer's ability to show intelligent behavior based on natural language (Turing, 1950). Since then, the field has emerged rapidly, and modern technologies such as search engines, and chatbots all rely on NLP based systems.

Deng (2018) describes the development of NLP as three major waves, namely rationalism, empiricism, and deep learning. The rationalism wave took place between 1950-1980 and was based on knowledge engineering and rule based systems (Chomsky, 1957). The empiricism wave started in the early 1990s with the introduction of statistics approaches and simple machine learning concepts (Manning et al., 1999). The motivation behind these concepts was to develop more generative models. Some of the most popular models were Hidden Markov Models (HMM) (Baum and Petrie, 1966), Support Vector Machines (SVM) (Vapnik, 1999) and Conditional Random Fields (CRF) (Lafferty et al., 2001). What they all had in common was their ability to learn and improve from data. This wave led to significant progress within translation tasks and speech recognition (Pinola, 2011), as well as on problems such as part-of-speech tagging, and word/sentence disambiguation (Liddy, 2001). The deep learning wave started around the year 2000 when Bengio et al. (2001) proposed the first neural language model. Deep learning models, combined with an increase in computer power and amount of available data, have since then advanced the NLP field remarkably. Deep learning models have proven to be capable of learning sophisticated features when processing language (LeCun et al., 2015) and LSTM based models have achieved impressive results in handling sequences of data. Deep learning, neural networks, and LSTMs are described further in section 2.3. Recently, new deep learning models, such as the Transformer (Vaswani et al., 2017) and pre-trained models, for example BERT, have pushed the boundaries even further, achieving new state-of-the-art results within a variety of NLP tasks. These models are further described in section 2.3.3.

2.1.1 Named Entity Recognition

Named entity recognition (NER) is an NLP task that focuses on classifying named entities in text. Named entities are entity instances that can be denoted with a name, i.e. persons, locations, and organizations. Named entities can be thought of as rigid designators, i.e. a term that refers to the same thing in all worlds that it exists in (Kripke, 1972). Entities are most often labeled with a I, O, or B label followed by the entity class. For example, the named entities in the sentence ["Lex", "studied", "at", "MIT"] would be labeled as ["B-PERS", "O", "O", "B-ORG"].

In clinical information extraction systems, it is often desirable to extract other types of entities. Here, clinical concepts such as problems, tests, and treatments are of high interest (Sun et al., 2013). This is further elaborated on in section 2.1.2.

2.1.2 Clinical Natural Language Processing

Clinical natural language processing refers to NLP methods applied in the clinical domain. Clinical NLP attempts to better utilize the text in EHRs (Friedman et al., 1999). The field dates back to the late 1980s, when various studies showed that NLP was a feasible technique in the domain and could be used to make progress in healthcare (Hripcsak et al., 1999). As EHRs have replaced traditional health records stored in paper, valuable clinical information have become more accessible for processing and analysis. Since the records are often stored in the form of unstructured free text, novel NLP techniques that can automatically extract relevant information are required to leverage the information (Meystre et al., 2008). The clinical NLP community has addressed this with a variety of challenges. Starting in 2006, the first challenge was organized by the i2b2 (i2b2, 2006) and included two challenges about de-identifying electronic records and classifying patients as smokers or non-smokers. The ShaRe/CLEF eHealth challenge (ShARe/CLEF, 2013) started in 2013 and introduced tasks such as recognizing diseases in clinical text. In 2014, SemEval proposed a similar task about entity, acronym, and abbreviation recognition (Pradhan et al., 2014) and has since then subsequently introduced tasks regarding extraction of temporal relations (Bethard et al., 2016, 2017). These challenges provide an opportunity to compare a variety of approaches to different clinical NLP tasks. Below are highlighted some of the most interesting studies and findings in relation to this project.

Sun et al. (2013) showed in the 2012 i2b2 challenge that machine learning techniques consistently outperform rule-based methods on event detection. In the SemEval 2016 challenge (task 12) Bethard et al. (2016) found that the best performing systems achieved “near-human performance on identifying events and times” after being trained on 293 clinical notes, containing 38.890 events. The system with the highest F1-measurement achieved an F1-score of 90.3% on identifying event span using an HMM-SVM model (Lee et al., 2016). In the SemEval 2017 (task 12) challenge (Bethard et al., 2017), where systems were trained on 38.890 events from one domain (colon cancer) and tested on another domain (brain cancer), the best performing system reached an F1-score of 72.0% with an LSTM based model (Tourille et al., 2017), indicating that deep learning based approaches are more suitable for a broad corpora. Another study by Lee et al. (2020) found that BERT models pre-trained on a biomedical corpora outperforms current LSTM and CRF models in biomedical named entity recognition.

All of the studies above were conducted on clinical notes written in English. Recent studies have started to examine clinical NLP in less explored languages. A study by Viani et al. (2019) aimed to extract events from medical records written in Italian. They found that an LSTM based model achieves the best result, reaching an F1-score of 90.1% when trained on 4.288 sentences, corresponding to 57.263 tokens. In another study on Spanish biomedical named entity recognition, Hakala and Pyysalo (2019) find that pre-trained models show promise for such tasks, achieving an F1-score of 87.0% with a multilingual BERT model fine-tuned on 177.022 tokens and pre-trained on a general Wikipedia domain. Zhang et al. (2019) explore this further by pre-training a BERT model on Chinese clinical texts which they show outperforms LSTM based models, achieving an F1-score of 93.53% on breast cancer event extraction.

Danish clinical NLP

Despite the great digitization of the Danish healthcare system, Danish clinical NLP has not been a very active research area (Copenhagen Healthtech Cluster, 2018). A study by Névél et al. (2018), which offers an overview of clinical NLP for other languages than English, finds that Denmark lags behind other many other coun-

tries. With only 4 PubMed references to clinical NLP, Denmark has half as many as Finland (9) and nearly a fourth of Sweden (15). Another study by [Kirkedal et al. \(2019\)](#) concludes that Danish clinical NLP still has not explored the potential of deep learning and that timeline extraction and entity recognition still are absent for Danish.

[Rasmussen and Berggrein \(2019\)](#) took a step in this direction with their master's thesis that explores entity recognition and disambiguation in Danish electronic health records. Using a CRF model, they build a system capable of recognizing disease entities from Danish health records, achieving an F1-score of 68.55%. Furthermore, they also propose a pipeline for sentence and entity disambiguation. To the best of our knowledge, no other recent studies have been conducted in the field of Danish clinical entity recognition.

2.2 REPRESENTING WORDS AND SENTENCES

The purpose of this section is to describe how words and sentences can be represented as embeddings and why that is useful in machine learning.

When processing text, it can be beneficial to represent the words and sentences as numerical values in order to capture and represent the semantic meaning. One technique, proposed by ([Salton et al., 1975](#)), enables this with the idea of a vector space model (VSM). Here, vectors can represent words, sentences, or entities, and the space represents the *semantic space*. One major advantage of this technique, is that it becomes possible to compare the similarity of words and sentences. The following two paragraphs describe how word and sentence vectors are computed.

2.2.1 Word embeddings

Word2vec is a technique, proposed by [Mikolov et al. \(2013\)](#), that maps words to a set of dense vectors. These vector representations are referred to as word embeddings. Word2vec is based on a feedforward neural network which is trained with different language modeling objectives. Another word embedding architecture that has been widely adopted is GloVe ([Pennington et al., 2014](#)). Instead of neural networks, GloVe makes use of an optimization problem to minimize the difference between the dot product of two words and the logarithm of their number of co-occurrences ([Pilehvar and Camacho-Collados, nd](#)). Both Word2Vec and GloVe are trained using unsupervised algorithms, meaning that word embeddings can be obtained for any words that are in the input corpora. The linguistic or semantic similarity between the resulting vectors can then be measured using the euclidean distance or cosine similarity.

One limitation to the pre-trained word embeddings generated by Word2vec and GloVe is that they are static. This means that an embedding for a word is the same no matter its context. That can lead to very inaccurate embeddings for words with multiple meanings, such as *terminal*, *cell* or *sign*. This limitation is tackled by contextualized word embeddings which differentiates the embeddings depending on the context, implying that the same word can have different embeddings. Different models that introduce contextualization have been proposed, such as Context2vec ([Melamud et al., 2016](#)) and ELMo ([Peters et al., 2018](#)). Most recent studies however, are based on the Transformer model (see section 2.3.3) which makes use of dynamic word embeddings using a technique called self-attention (section 2.3.3). Generally speaking, the technique allows the model to consider the context when processing a word. The Transformer model takes as input the whole text span, which it needs to capture the context, and produces dynamic embeddings for each word (token) in the input.

2.2.2 Sentence embeddings

The previous section described how words can be represented as embeddings. Sometimes it can be useful to represent an entire sentence as an embedding, e.g. when comparing or classifying sentences. [Arora et al. \(2016\)](#) suggest that a weighted average of the word embeddings in a sentence can be used to represent the entire sentence as a single embedding. A main disadvantage of this technique however, is that it does not take the word order into account, making it unsuitable for cases where the order is important.

Again, Transformer based models have shown promise in generating representative embeddings. Different approaches can be used to generate sentence embeddings with a BERT model (section 2.3.4). [Qiao et al. \(2019\)](#) use the [CLS] token, which is prepended to the input sentence, to extract a vector representation of a sentence. As BERT_{base} has 12 layers, the resulting sentence embedding is different depending on which layer the [CLS] token is retrieved from. According to [Xiao \(2018\)](#), the [CLS] representation of the first layer is close to a sentence representation of the original word embeddings, whereas the last layer is closer to the target function.

2.3 DEEP LEARNING

This section aims to describe deep learning in general, as well as the two deep learning methods (and their overlying dependencies) implemented in this project, namely LSTM and BERT.

Deep learning is a term used to refer to a broad number of machine learning approaches based on artificial neural networks. Deep learning methods rely on representation learning, allowing them to independently discover the representations needed for feature detection and classification without being fed anything but the raw data. This gives the deep learning methods a significant advantage over more traditional machine learning approaches that are more reliant on human engineering and domain expertise, and are thus more limited in their capability to learn from raw data alone ([LeCun et al., 2015](#)).

As with more regular machine learning methods, deep learning can be supervised, unsupervised, or semi-supervised. The most common form of machine learning according to [LeCun et al. \(2015\)](#), and the primary one employed in this project, is supervised learning. In supervised learning, the system is fed labeled data and learns from these labels. For example, a simple model for classifying spam emails could have been shown a training data set of several emails, where each email is labeled the correct category, i.e. spam/not spam. Through its training, the model should learn what spam email does, and does not, look like. In reality (and very simplified) the model outputs a vector of scores representing each category. During training, these scores are applied to some cost function that measures the error between the model's predictions and the correct labels. The model then adjusts a number of internal weights to minimize its error and thus "learn" ([LeCun et al., 2015](#)). The end result is a model representing an approximate function based on its training.

2.3.1 Artificial neural networks

Deep learning methods are based on artificial neural networks. Simply put, a neural network is a collection of connected units, while the properties of the specific neural network is a result of its topology and the characteristics of the neurons within the network ([Russell and Norvig, 2016](#)). The most basic form of neural network consist

of some number of connected units. The link between units i and j has a numeric weight $w_{i,j}$ that decides the strength of the information propagated to unit j in the link through the activation a_i . The unit j then computes a weighted sum of its inputs, in_j :

$$in_j = \sum_{i=0}^n w_{i,j} a_i$$

A chosen activation function g is then used on the input in_j to compute the output of unit j , a_j :

$$a_j = g \left(\sum_{i=0}^n w_{i,j} a_i \right)$$

The way the units of the neural network are connected may differ. The most basic form of network is the feed-forward network, where the connections represent a directed acyclic graph, i.e. the connections move only in one direction. Alternatively, the neural network may contain loops, where unit outputs are fed back again as inputs, effectively allowing the network to achieve short-term memory through its hidden states (Russell and Norvig, 2016). Such a network is called a recurrent neural network (RNN). Due to its ability to process sequences and to some degree hold a memory of its processed information, RNNs are generally well suited for sequence labeling problems. Figure 1 shows the basic architecture of an RNN. It visualizes how the input layer x passes information to the hidden state s , which outputs information to the output layer y . The interesting thing to note is that along with information from the input layer, the hidden state also receives the output from the previous time step, adding to its contextual information.

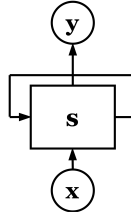


Figure 1: Basic representation of a recurrent neural network

The RNN processes sequences by applying a recurrence formula at each time step that computes the next hidden state h_t using the old state h_{t-1} , the input x_t and an activation function f with some weight matrix W (Li et al., 2017):

$$h_t = f_W(h_{t-1}, x_t)$$

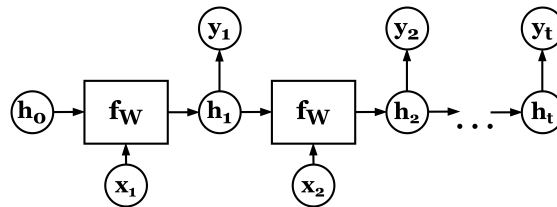


Figure 2: Unfolded representation of a recurrent neural network

Neural networks are generally built up of layers of units such that units receive input from units in the preceding layer. The units may connect directly with the network's inputs and outputs, in which case the neural network is single-layered, or it may consist of multiple layers with hidden units that are neither connected

directly to the network's inputs or outputs, in which case it is a multi-layer neural network (Russell and Norvig, 2016).

In order to learn, a neural network applies a cost function that takes the neural network's weights and biases as input, and outputs a number expressing how well or how bad the network performed for that iteration according to the true target variables. As mentioned earlier, the learning happens by minimizing the cost through weight adjustments. Gradients are used to express which changes in the weights cause the fastest change in the cost function. Since the term gradient refers to the direction of the steepest increase in some space, we seek to minimize the cost by taking the negative gradient, i.e. moving in the opposite direction of the gradient of the cost function. This descent is maintained until a local minima is reached (Ruder, 2016). This process is what is known as gradient descent (LeCun et al., 2015). The algorithm for computing the negative gradients is called backpropagation which, contrary to the regular propagation that happens in the neural network, iterates from the output layer back to the input layer, computing the negative gradients along the way (LeCun et al., 2015).

One major problem with this approach of error signals flowing backwards in time, is that the gradients may either vanish or explode over time (Hochreiter and Schmidhuber, 1997). Vanishing gradients may result in gradients so small that the weights no longer change their values in a meaningful way, effectively meaning that the model stops learning. Meanwhile, exploding gradients may result in oscillating weights (Hochreiter and Schmidhuber, 1997).

2.3.2 Long Short-Term Memory

The LSTM network is a deep neural network specifically designed to deal with the vanishing and exploding gradient problem, by ensuring constant error flow in the hidden states (Hochreiter and Schmidhuber, 1997). This helps the network more effectively learn long-term dependencies compared to a traditional RNN (Sak et al., 2014). As a result, the LSTM is better at learning from wider contextual information.

The LSTM holds memory blocks, each containing a cell state and three gates controlling the information flow between each memory block (Sak et al., 2014). The cell state is often referred to as the memory of the LSTM, as it flows through all memory blocks. The forget gate decides how much of the information from the previous cell state to keep. The input gate decides how much of the new input to the memory block will be added to its cell state. Lastly, the output gate decides how much of the memory block's information the rest of network will be exposed to (Sak et al., 2014). The memory block of a simple LSTM is shown in Figure 3. Given input x_t and the previous cell state C_{t-1} containing information from previous states, the memory block computes its output h_t and the next cell state C_t .

The output and cell state is calculated using the following functions (Pilehvar and Camacho-Collados, nd):

$$\begin{aligned} f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \\ i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\ o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \\ c_t &= f_t \circ C_{t-1} + i_t \circ \tanh(W_c x_t + U_c h_{t-1} + b_c) \\ h_t &= o_t \circ \tanh(C_t) \end{aligned}$$

Where f_t , i_t and o_t denote the three gates. A sigmoid function σ is applied for each gate, ensuring that the multiplier is a value between 0 and 1, effectively deciding how much of the information flows through the gate. Hyperbolic tangent ($\tanh$) is used to avoid that the cell state explodes, by ensuring that information

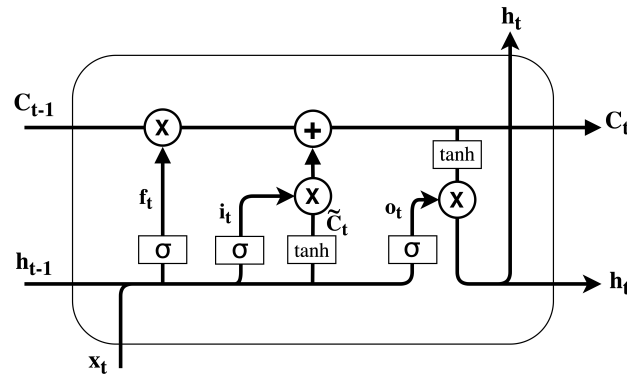


Figure 3: The memory block of a simple LSTM network which shows how the information is regulated by gates

added to the cell state is between -1 and 1 . For some sequence of words, the memory cell's input x_t is the word embedding vector at position t in the sequence. After flowing through each gate, the transformed vector h_t is outputted at that time step t , but also passed as input alongside x_{t+1} (i.e. the next word embedding vector in the sequence) in the next time step. As such, in a unidirectional LSTM, only the first word of a sequence will see no influence from the rest of the words, while only the last word will see an influence from all of the words.

A popular variant of the LSTM is the Bidirectional LSTM (BiLSTM). One limitation of a regular unidirectional neural network is that output is only influenced by previous context, and not by whatever context comes after. One solution to this issue is the BiLSTM, where two neural networks, connected to the same output layer, process the sequence forwards and backwards. This results in a collective network that at any time step has full knowledge of past and future context (Graves and Schmidhuber, 2005). Figure 4 displays the basic structure of a bidirectional network, showing how sequences are processed in both directions.

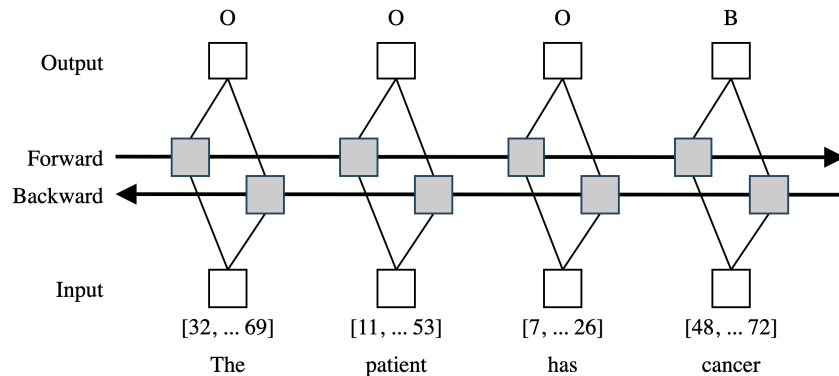


Figure 4: Example of a bidirectional network in a sequence labeling task. The input is a sequence of word embedding vectors, while the output is a sequence of predicted labels.

2.3.3 Transformer

The Transformer model was proposed by Vaswani et al. (2017), who describe the architecture of the Transformer model and the concept of multi-headed self-attention. This section describes the architecture and the important components of a Transformer model together with an example of how it is used for machine translation.

Vaswani et al. (2017) describe the overall architecture of the Transformer model as a unit of two components; an encoder and a decoder. The encoder takes a sentence, i.e. a sequence of token (word) embeddings and maps them to a sequence of continuous representations. This representation is then passed to the decoder, which generates an output sequence of new tokens based on the representation. For a translation task, this is the actual translation of the input sentence.

Encoder and decoder components

The structure of the the encoder and decoder components is described by Vaswani et al. (2017) as the following:

Encoder The encoder component is a stack of 6 identical encoder layers that all follow the same structure. The structure is build around two sub-layers, a multi-headed self-attention layer and a position-wise dense feed-forward neural network. For each of the two sub-layers, a residual connection is added to allow information to flow without passing an activation function. Finally, the information flows through a dropout and normalization layer to avoid overfitting and to control the scale of the input and output.

Decoder The decoder component is structurally similar to the encoder in that it consists of 6 identical layers. However, the layers are different in that they are decoder layers that consist of three (and not two) sub-layers. The first two sub-layers are the same as for the encoder, i.e. a multi-headed self-attention layer and a dense feed-forward neural network. The third sub-layer is a ‘masked multi-head attention’ layer, which is used for the translation task to ensure that predictions of words only are based on tokens before the current token. Finally, residual connections around each sub-layer followed by a layer of normalization are also employed.

Attention

Attention is a technique proposed by Bahdanau et al. (2014), which enables the Transformer model to capture long-range dependencies and focus on the relevant parts of the input sequence. Take the following sentence as an example: *"The test went well and according to the medical record it showed no cancer"*. Here, *"it"* can refer to both the test and the medical record. Self-attention is the attention mechanism that allows *"it"* to be associated with *"test"*. An example of the attention mechanism is visualized in Figure 5 (Alammar, 2018b).

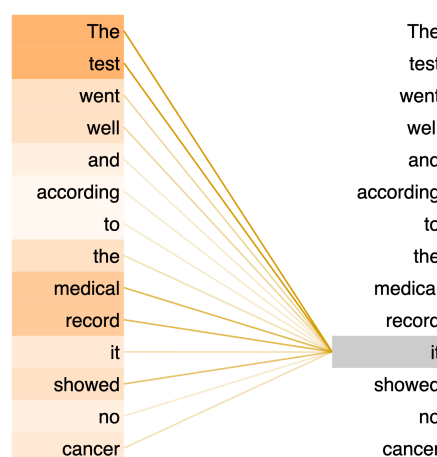


Figure 5: Example of how attention works. When the word *"it"* is encoded, the attention mechanism associates the word mostly with *"the test"*. Darker colors indicate a higher degree of attention.

Alammar (2018b) describes the attention mechanism as a function which maps a query and a set of key-value pairs to an output. The query, key, and value are all vectors that are created by multiplying the input, i.e. the embedding of every token, by three weight matrices that were created during the training process. The self-attention is calculated for each token, resulting in a score that determines how much focus that token should give to other parts of the input.

An important property of the Transformer architecture is that the attention function can be calculated simultaneously if the set of queries is stacked as a matrix Q . With the keys K and the values V also put into matrices, the attention function is given by:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

The output is a matrix that holds the attention of each token. The fact that queries, keys, and values can be stacked as matrices is one of the reasons that Transformers can achieve parallelization during training (Vaswani et al., 2017). This means that the model does not need to process the beginning of a sentence before it process the end, leading to significantly shorter training time compared to other deep learning methods, such as LSTMs.

Multi-headed self-attention

In their paper, Vaswani et al. (2017) propose to employ several attention layers by adding the mechanism called ‘multi-headed’ self-attention to the architecture. In the paper, they employ 8 attention heads (layers). This mechanism enables the model to focus on different positions by calculating the attention 8 times with different weight matrices (Alammar, 2018b). The result is a new matrix that contains information from all of the attention layers, where each row represents the attention vector of a token.

Positional representation

Because the Transformer model does not follow the RNN structure where information flows recurrently, information about the positions of the tokens in the sequence must be given to the model. Practically, this is done by adding a positional encoding vector to each input embedding at the bottommost layer, which adds information about the position and the distance to other words. This can be seen on Figure 6.

2.3.4 Bidirectional Encoder Representations from Transformers

BERT was introduced by Devlin et al. (2018). They present a new bidirectional Transformer architecture, achieving eleven new state-of-the-art results on NLP related tasks. A key property of the proposed model is that it has a unified architecture across different tasks such as NER, question-answering, and sentence classification. This means that the difference between the pre-trained architecture and the final architecture for downstream tasks is very small. Secondly, the framework is based on pre-training and fine-tuning, enabling transfer learning and domain adaptation. This section aims to describe the architecture of BERT and the concepts that the model builds and relies upon.

BERT architecture

Devlin et al. (2018) describe the BERT model architecture as a multi-layer bidirectional Transformer encoder. Two models are presented in the paper, $\text{BERT}_{\text{base}}$ and $\text{BERT}_{\text{large}}$, which both are given by three parameters: L : Number of layers (i.e. Transformer blocks/encoders). H : Hidden size (i.e. hidden units in the feedfor-

ward Neural Network). A: Number of self attention-heads. As our implementation is based on BERT_{base} ($L = 12, H = 768, A = 12$), these parameters will be used as example in the following description of the architecture.

Simply put, BERT is a stack of trained Transformer encoders where each encoder follows the same overall structure as the encoder described in section 2.3.3. BERT has 12 of these Transformer encoders that each consists of a multi-headed self-attention layer and a feed-forward neural network. An important thing to note, is that BERT does not have any decoders. The overall BERT architecture can be seen in Figure 6.

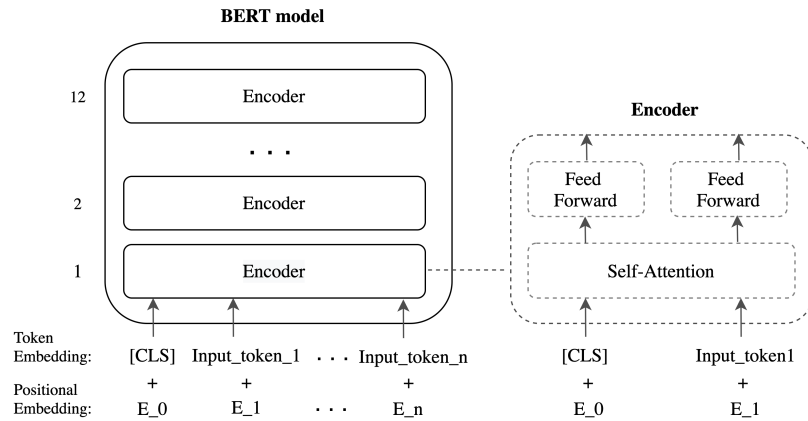


Figure 6: High level architecture of a BERT model and an encoder

Another difference in BERT is its bidirectionality. Since the architecture only consists of encoders, BERT can condition on both the left and right context. The potential drawback, is that the model can see the token it is trying to predict. This is resolved by using a masked language model in the pre-training phase (Alammar, 2018a).

Pre-training and fine-tuning

The BERT architecture is build on a framework that consists of two steps: pre-training and fine-tuning (Devlin et al., 2018). This framework is what enables the unified architecture across different tasks, and what allows transfer learning in BERT, meaning that only fine-tuning is needed for domain-adaptation for already pre-trained models .

BERT is pre-trained with two unsupervised tasks, masked language model and next sentence prediction (Devlin et al., 2018). All parameters that are learned with the two tasks are transferred to initialize the downstream tasks that follows with fine-tuning.

The idea behind the masked language model is to mask some of the input tokens and then predict on those masked tokens. The purpose of next sentence prediction is to give the model an understanding of the relationship between two sentences. The training task is therefore about determining whether a given sentence follow or does not follow another given sentence.

BERT can be fine-tuned for a variety of tasks. This is a very useful property as it leverages the pre-training and allows for a relatively cheap tuning of the model in terms of computer power, time and amount of data (Alammar, 2018a). The fine-tuning process includes setting the task-specific input and output as well as a supervised fine-tuning of all the parameters. Figure 7 (Devlin et al., 2018) is an example of a fine-tuned BERT in a token classification task.

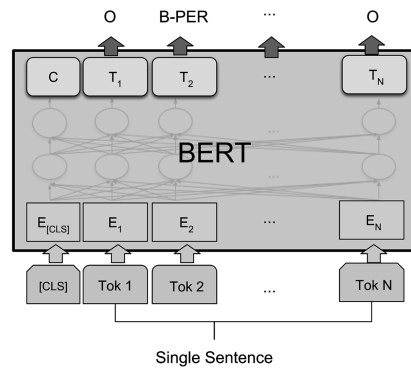


Figure 7: BERT fine-tuned for token classification

Processing of tokens

This subsection aims to describe the mechanisms that make the BERT architecture possible, namely the BERT tokenizer and its input/output representation.

The tokens are stored as WordPiece embeddings, meaning that unknown tokens are broken down into subtokens. This is done by the BERT tokenizer. For example, if the word "discharged" does not exist in the vocabulary, it is split into [dis, '##charge, ##d] where each word in the list is a wordpiece that the model eventually has. The two hash-symbols indicates that the words are subwords.

In order to facilitate the two training tasks in pre-training, BERT requires a special input representation. Besides the WordPiece embedding, BERT adds the position embedding and a segment embedding that is used for the next sentence prediction. Finally, special tokens are added to mark the first token ([CLS]) and the sentence separation ([SEP]). For example, "He was discharged from the hospital. It was a relief." would be represented as ["[CLS]", "He", "was", "dis", "##charge", "##d", "from", "the", "hospital", ".", "[SEP]", "It", "was", "a", "relief", ".", "[SEP]"]

Pretrained BERT models

The BERT architecture has become increasingly popular, not only because of the state-of-the-art results (Devlin et al., 2018), but also because of its ease of use for a variety of tasks and its accessibility through services such as Hugging Face (Hugging Face, nd). Below is a list of the pre-trained BERT_{base} models that are used in this project. The motivation for choosing these specific models was that they had been pre-trained on either Danish, Danish-Latin, or data related to bio-medicine.

BERT Multilingual (Devlin, 2019) Trained on 104 languages, including Danish and Latin. The model has a 110.000 WordPiece vocabulary.

Danish BERT_{v1,v2} (Mollerhoj, 2019) Trained on Danish language text from Common Crawl, The Danish Wikipedia, Custom scraped data from the two biggest Danish debate forums (dindebat.dk and hestenettet.dk) and Danish OpenSubtitles. It has a total of 32.000 tokens in its vocabulary. There are two versions available, without any specification on the differences between the two.

SciBERT (Beltagy et al., 2019) Trained on scientific text with a total of 1.14M scientific (mainly English) papers. The corpus consist of 18% papers from the computer science domain and 82% from the broad biomedical domain. It has a vocabulary of 30.000 tokens.

Bio-ClinicalBERT (Alsentzer et al., 2019) Trained on English electronic health records from the MIMIC-III, a database that contains records from patients from the Beth Israel Hospital in Boston. The model was trained on around 880M words.

2.4 ANNOTATION

Annotating is the act of associating some type of document with extra information. It is a process especially relevant within the field of NLP, where annotated documents may function as training and test data for machine learning models, and where the annotation itself represents the ground truth that supervised models learn from. For example, annotation is used in NER for labeling words of sentences with tags such as *person*, *location*, and *organization*. For clinical NLP, Velupillai et al. (2015) highlight ensuring a well-defined underlying schema and a reliably-annotated corpus as the most crucial step for enabling semantic analysis. However, they also point out that manual annotation is time consuming, expensive, and labor intensive. Generally, annotation is one of the major challenges for research within the field of clinical NLP. In their study on the properties of text data used in clinical NLP, Spasic and Nenadic (2020) conclude: *"We identified the data annotation bottleneck as one of the key obstacles to machine learning approaches in clinical NLP"*.

Velupillai et al. (2015) state that annotation is traditionally done by domain experts or linguists. In this study, annotation of the health records was done by software development students, greatly increasing the challenge of annotation. This also underlines the importance of a well-defined schema, or *annotation guidelines*. If they are clear, such guidelines can help make annotation easier and faster, yet they can be challenging to define. Guidelines for event annotation does exist, such as the ISO-TimeML standard (Pustejovsky et al., 2010), the THYME corpus (Styler IV et al., 2014), which is an extension of ISO-TimeML, and guidelines for Italian health records by Viani et al. (2019). However, currently no annotation guidelines exist for Danish clinical text.

The annotation process of this study is described in detail in section 4.2. To help ensure a reliably-annotated corpus, a set of annotation guidelines for Danish health records were created, which can be seen in appendix A.1.

2.5 ACTIVE LEARNING

As stated in section 2.4, annotation is challenging and costly. Therefore, this section explores the field of active learning, which has shown promise in optimizing annotation efforts.

Active learning is a branch of machine learning where the learning algorithm plays an *active* role in its learning (Settles, 2009). The key hypothesis is that if the learning algorithm is allowed to take part of the selection of the training data, it will lead to a model that performs better with less training data compared to a model trained on randomly selected data. Given the sparsity of labeled data in this project and in the clinical domain in general, this is an interesting property as it could potentially reduce the costly annotation process and lead to better performing systems.

Chen et al. (2015) found that an active learning algorithm for clinical NER could save 42 % of the annotation cost in words compared to random sampling, to achieve an F1-score of 80 % with a CRF-classifier. The study used the annotated corpus from the 2010 i2b2/VA NLP challenge (Uzuner et al., 2011) that includes discharge summaries. Another study by Kholghi et al. (2016) concluded that *"active learning is a promising approach for building effective and robust medical concept extraction models while significantly reducing the burden of manual annotation"*. The study found the annotation effort saved by active learning was up to 77% of the total number of sequences, 57% of the tokens, and 46% of the concepts. The study also used a CRF-classifier to train and test on two clinical data sets, namely the i2b2/VA and the ShARe/CLEF (Pradhan et al., 2013).

A study by [Shen et al. \(2017\)](#) found that active learning combined with deep learning can reduce the amount of labeled training data significantly. With a CNN-CNN-LSTM model for named entity recognition, the authors were able to almost match state-of-the-art performance with just 25 % of the original training data. The system was tested on CoNLL-2003 English ([Sang and De Meulder, 2003](#)) and OntoNotes-5.0 English ([Pradhan et al., 2013](#)).

One noteworthy limitation of these studies is that they are conducted in a simulated setting, i.e. the active learning process is simulated as the annotated data is already available. In the first non-simulated active learning experiment in the clinical domain, [Chen et al. \(2017\)](#) found that active learning systems do not always guarantee to save annotation costs in practice. They suggest that although active learning systems might query more informative sentences to be annotated, it is offset by an increase in the time needed to annotate these sentences. In relation to this conclusion, the authors emphasize that the performance of active learning systems is very much impacted by the active learning algorithm, i.e. the query strategy.

2.5.1 Query strategies

Finding the next data item to label is the core of active learning. This means that the way in which a pool of data is *queried* (or *sampled*) for potential training examples is very important. This section provides an overview of the query strategies used in this project. The active learning system is based on a pool-based setting, which assumes that the system samples from a large (static) pool of unlabeled data ([Settles, 2009](#)).

Random sampling

Random sampling is not an actual active learning sampling strategy, but is included in this list as it serves as a method for establishing a baseline that the other sampling strategies is compared to. According to [Munro \(2020\)](#), this is necessary to measure the effect of the different active learning strategies. The sampling is simply done by randomly selecting sentences from the unlabeled pool.

Uncertainty sampling

Uncertainty sampling is based on the idea that the system queries the sentences that it is least certain about how to label ([Lewis and Gale, 1994](#)). With uncertainty sampling the model first has to make its predictions on all sentences in the unlabeled pool. Next, the active learning system queries the sentences which contain words that the model is the most uncertain about. The model is said to be uncertain when an unlabeled word is near the decision boundary ([Munro, 2020](#)). For a binary classification task, this is when a prediction is close to 50% probability of belonging to either class.

There are different algorithms for calculating the uncertainty. One popular algorithm is Least Confidence (LC), which is the one used in this project. It is calculated by:

$$x_{LC} = 1 - \max P[y_B, y_O \mid x_0..x_i]$$

Where x_0 and x_i denote the first and last word in sentence X . Given two sentences: $X_1 = \text{"The patient is ill"}$ and $X_2 = \text{"The patient is cold"}$ with the following uncertainties: $U_1 = [0.1, 0.1, 0.1, 0.3]$ and $U_2 = [0.1, 0.1, 0.1, 0.5]$, uncertainty sampling will query the sentence with the highest LC, i.e. the one closest to 0.5. In this example X_2 would therefore be queried before X_1 . [Chen et al. \(2015\)](#) and [Shen et al. \(2017\)](#) both find that uncertainty sampling significantly outperforms random sampling.

Diversity sampling

Diversity sampling is a strategy used to expand the decision boundary into different parts of the feature space (Munro, 2020). The motivation behind the strategy is that training a model on very similar sentences does not improve the model as much as if it was done on a *diverse* set of sentences. (Ertekin et al., 2007; Munro, 2020). There are different approaches to diversity sampling. In this project, a cluster-based diversity sampling combined with uncertainty sampling is explored. With diversity sampling, the unlabeled pool is divided into clusters of semantically similar sentences. As each sentence in the unlabeled pool is represented as a vector, traditional unsupervised algorithms can be used to assign a cluster to each sentence. Once the sentences are divided into clusters, diversity can be ensured by sampling from the respective clusters. There are different approaches on how to sample from the clusters, such as uncertainty sampling explored by Chen et al. (2017) and Shao et al. (2019).

3 | DATA AND MATERIAL

This chapter will first describe the data that this project uses, namely Danish electronic health records. Secondly, the motivation for choosing a subset of this data is described. Finally, the software packages and libraries used in processing and otherwise working with this data are presented.

3.1 ELECTRONIC HEALTH RECORDS

The data that this project uses is a result of the work by [Pantazos et al. \(2011\)](#). The authors attempted to de-identify a Danish EHR database with 437.164 patients. The resulting database consists of health records from 323.122 patients with “an acceptable degree of anonymity, readability, and correctness” ([Pantazos et al., 2011](#)). They conclude that the database is adequate for use in research, development, and training, where the users are aware of the confidentiality. For this reason, an agreement was signed ahead of the data delivery for this project, ensuring that the data is handled confidentially.

The exact data worked with in this project is a subset of this database, specifically a relational table consisting of two columns, *ICD10ID* and *MEDICALRECORDTEXT*. The *ICD10ID* column contains a single ICD10 code. ICD is short for *International Classification of Diseases*, and consists of a list of disease classification codes. ICD10 is the 10th revision of this list ([WHO, nd](#)). As an example, the ICD10 code *C50* refers to breast cancer. The *MEDICALRECORDTEXT* column contains text, typically a sentence, related to the ICD10 code and some patient.

In total, this data set contains 12.351 unique values in the *ICD10ID* column and 5.822.617 rows. Table 1 shows the first five rows.

	ICD10ID	MEDICALRECORDTEXT
0	E105B	notat
1	i10	Hjertebanken og ondt i venstre arm....
2	A01	Svært at gennemskue hvad patienten fejler.
3	A01	Vi må vente på testresultaterne, der kommer senere i dag
4	A085	Denne patient er absolut ikke i koma, han er blot en dovenlars, der ikke gider gå på arbejde.

Table 1: First five rows of the raw data this project works with

Of course, some ICD10 codes are more represented than others. The count of a given ICD10 code is one of the factors taken into consideration in this project when it comes to limiting its scope to a certain clinical domain, as explained in section 3.2. Some of the most represented codes include:

- Z039 (suspected disease/condition)
- I10 (high blood pressure)
- J459 (asthma)
- J301 (pollen allergy)
- C50 (breast cancer)

It is important to mention that much of the data in the ICD10ID column is dirty, meaning it does not always represent an actual ICD10 code. Therefore, out of the 12,351 unique values found in the ICD10ID column, not all are actual disease classifications. Examples of dirty ICD10 codes include "285752", "R————+——", "MECILLINAM", and ", 172/99,". Generally, these dirty codes have a low occurrence count, while actual ICD10 classifications appear much more often in the data set.

3.2 CHOOSING A CLINICAL DOMAIN

The scope of this project is limited to working with data in the domain of breast cancer (C50). The decision to limit the data to one single domain is primarily due to three reasons:

1. It is deemed outside the possibilities of this project to annotate enough data within the time frame for a proper all-domain model to be developed
2. Other comparable studies likewise limit their scope to a single domain, such as [Bethard et al. \(2016\)](#) and [Viani et al. \(2019\)](#)
3. The annotation process is sped up as medical terms reappear, limiting the number of look-ups, which can otherwise be very time consuming

The decision to work with breast cancer as a clinical domain is primarily due to four reasons:

1. There is a sufficient amount of data available (25,263 sentences)
2. It is a societal problem. Breast cancer is the most common type of cancer for women in Denmark with 4,700 yearly cases ([Kræftens Bekæmpelse, nd](#))
3. Medical terms within this domain are often repeated, speeding up annotation as it progresses
4. Records within this domain contain many words, and less so numbers (in contrast to e.g. the high blood pressure domain, which otherwise fits the three reasons above)

3.3 UNLABELED AND LABELED DATA

This project generally works with two types of data sets, unlabeled pools and labeled pools. An unlabeled pool refers to a table of all pre-processed data that has not yet been annotated. Meanwhile, a labeled pool contains all annotated data. Table 2 gives an overview of exactly what type of data these pools contain. Unlabeled pools are used for sentence querying, i.e. extracting sentences for annotation. Labeled pools are used for model training and testing.

This projects works with a range of unlabeled and labeled pools, each built specifically for a task. Overall, these tasks are:

- Training and testing a base model for determining the most promising model architecture for clinical event extraction (research question 1)
- Training and testing random, uncertainty, and diversity sampling models (research question 2)
- Training and testing a final model on all labeled data (research question 3)
- Testing a final model on another domain (research question 4)

	UL (random)	UL (uncertainty)	UL (diversity)	Labeled pool
ID	X	X	X	X
Sentence	X	X	X	X
Words	X	X	X	X
Length of sentence	X	X	X	X
Span labels				X
Polarity labels				X
DocTimeRel labels				X
Experiencer labels				X
Uncertainty		X	X	
Sentence embedding			X	
Cluster			X	
Probability			X	

Table 2: Columns of unlabeled and labeled pools (UL = unlabeled pool)

The unlabeled pool with breast cancer related sentences from where sentences were initially sampled, contains 25.263 unique sentences. The pool is a result of a number of pre-processing steps, including sentence boundary disambiguation and tokenization (explained in section 4.1.3).

The mean sentence length in the unlabeled pool is 18.5 words, with the shortest sentence being 2 words, and the longest being 283 words. Figure 8 shows the distribution of sentence lengths in the unlabeled pool. For all the labeled pools collectively, the average number of clinical events per sentence is 2.78.

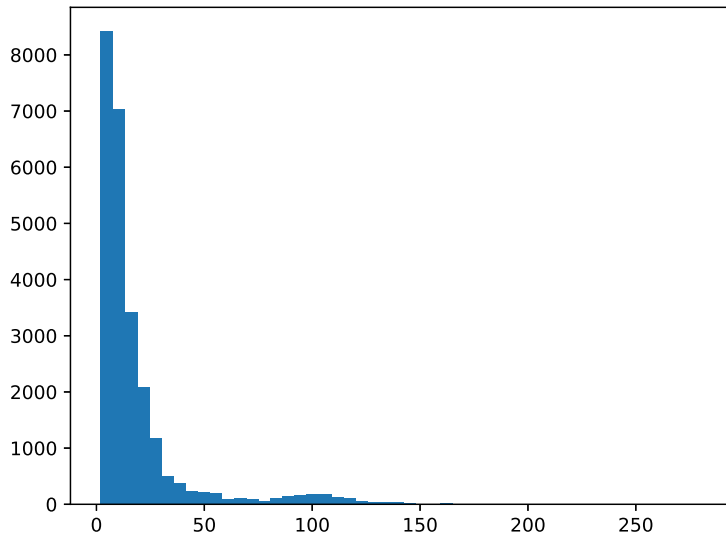


Figure 8: Histogram of the sentence lengths in the initial unlabeled pool

All three active learning query strategies were tested on a labeled pool containing 640 randomly sampled sentences from an unlabeled pool, in accordance with the recommendation by Munro (2020). The base model and random sampling based model were trained on a labeled pool of 512 randomly sampled sentences. The uncertainty and diversity based models were trained on labeled pools with up to 512 uncertainty and diversity sampled sentences respectively. See Figure 9 for an overview of these pools.

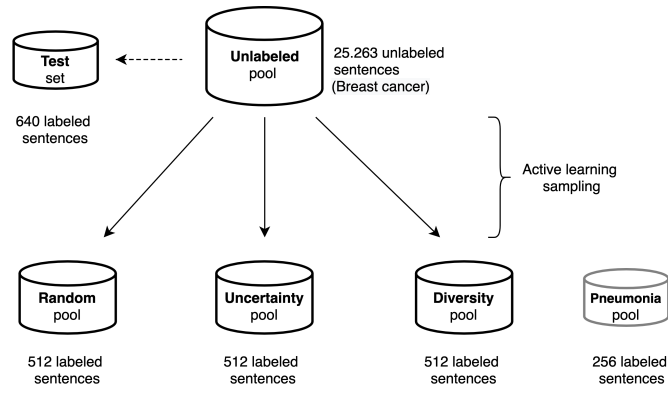


Figure 9: Overview of the unlabeled pool and labeled pools

The final model was trained and tested using a data set containing all data from previous labeled pools, with no duplicates, resulting in 2,090 sentences containing 42,754 words and 5,809 events. For the cross-domain experiment, the final model was tested on a labeled pool of 256 randomly sampled sentences from the pneumonia domain. Table 3 gives an overview of the amount of data in the labeled pools.

Pool	Sentences	Words	<i>unique</i>	Events	<i>unique</i>
Test set	640	12.919	2.345	1.865	654
Random pool	512	9.424	2.075	1.144	446
Uncertainty pool	512	12.221	2.508	1.607	601
Diversity pool	512	10.559	2.286	1.573	608
Pneumonia pool	256	3.473	1.244	467	282

Table 3: Sentence, word, and event counts for the final labeled pools

3.4 PACKAGES AND LIBRARIES

This project makes use of a range of software packages and libraries to assist with a wide number of tasks, including data pre-processing, modelling and evaluation. This section aims to provide a brief overview of the most noteworthy of these tools. Very common tools, such as pandas and NumPy, are not commented.

Task	Tool
Pre-processing	<i>Polyglot</i> (Al-Rfou, nd) Sentence boundary disambiguation and tokenization
LSTM	<i>Tensorflow/Keras</i> (Chollet et al., 2015) Building, training, and testing LSTM based models
Transformers	<i>Hugging Face</i> (Hugging Face, nd) Access to pre-trained transformer models <i>SimpleTransformers</i> (Rajapakse, 2020) Building, training, and testing transformer models. The repository was modified to return softmaxed label certainties during model predictions. See github.com/frwon/simpletransformers for the modified repository
Sentence embeddings	<i>bert-as-service</i> (Xiao, 2018) Obtaining sentence level embeddings from BERT model
Dimensionality and clustering	<i>scikit-learn</i> (Pedregosa et al., 2011) PCA dimensionality reduction and K-means clustering
Model evaluation	<i>scikit-learn</i> (Pedregosa et al., 2011) Classification reports containing F1-score, precision, and recall

4 | METHODS

The scope of this project necessitates a wide range of experiments to be carried out. The items below serve to provide an easy overview of which experiments were carried out and their overall goal.

- Experiments on a range of deep learning architectures based on LSTM and BERT. The best performing architecture is the one used for the rest of the project. At this step, the models are only evaluated on their performance in recognizing the event span, while other attributes are not considered. See section 5.1 for results.
- Experiments on three different query strategies (random sampling, uncertainty sampling, and diversity sampling) to identify the best performing active learning strategy. See section 5.2 for results.
- Experiments on the best performing architecture when training it on all labeled sentences gathered throughout the project. Here, model performance across all attributes are considered. See section 5.3 for results.
- Cross-domain experiment where a model is trained on all labeled data gathered throughout the project and tested on another domain. See section 5.4 for results.

This chapter will describe the fundamental actions required to conduct these experiments, including data handling, annotation, and model implementation. Furthermore, it will elaborate on the setup of these experiments. The results can be found in chapter 5.

4.1 DATA HANDLING

This section will first describe the ethical considerations which must be taken into account when working with clinical data. Secondly, it will describe how the data was handled in regards to privacy and confidentiality concerns. Finally, it will describe the data pre-processing that was carried out prior to annotation.

4.1.1 Ethical considerations

Several ethical considerations should be taken into account when working with EHRs. Since EHRs contain health-related data, it is considered as sensitive personal data according to the General Data Protection Regulation (GDPR) ([European Commission, nd](#)). This implies that the processing of it must not disclose any sensitive information such as names, identification-numbers, and biometrics. In this project, this is counter-measured by the de-identification of the data by [Pantazos et al. \(2011\)](#).

Another important consideration is how the data is accessed and handled. If this is done improperly, not following the clinical data agreement, it could potentially have a negative impact on the whole field of digital health care and clinical NLP. A non-responsible handling of the data could cause people to become skeptical about digital health care, hindering research and future projects in the area. It could also have significant personal consequences for some individuals if they for

some reasons were identified in the data set. These concerns were addressed in this project by strictly following the conditions stated in the clinical data agreements, including not leaving any activity viewing the data open on the computer screen, not re-distributing any data, and notifying the contact in any incident of breach.

4.1.2 Working with sensitive data

For this project, a number of precautions had to be taken to ensure that the data was handled correctly in accordance with the concerns described above. This was especially the case for data sharing. Since part of the annotation was not always done on a single machine, yet all annotated data had to be available on a single machine, it was necessary to develop a system that allowed for label sharing without sharing the confidential sentences. Also, as active learning based annotation is an iterative process and has to be done multiple times, the sharing system had to be simple and easy to use. In the end, three scripts helped achieve such a system. The respective task of these scripts were (1) write sentences for annotation based only on sentence IDs, (2) read the annotated data and create a file consisting only of sentence IDs and associated labels, (3) cross-reference the sentence IDs with the unlabeled pool, adding the actual sentences and their labels to the labeled pool, and removing the sentences from the unlabeled pool. This system made it possible to safely share the annotated data across machines such that all data was collected in one place and ready for training and testing.

During the process of the project, access to a secure remote server was given. This server allowed for even simpler data sharing, as all data could now safely be written and read from there.

4.1.3 Data pre-processing

Two major pre-processing tasks were undertaken to prepare the unlabeled pools, namely sentence boundary disambiguation (SBD) and word tokenization. While the data had already been exposed to some degree of SBD, it was found to be severely lacking. It had seemingly struggled especially with abbreviations, often cutting sentences too short. For both SBD and word tokenization, this study makes use of the polyglot library (Al-Rfou, nd), which supports 165 languages, including Danish. While several other tools were tested, including NLTK, the best results were found with polyglot.

Table 4 shows a number of problematic sentence boundaries as a result of the previous work, and how these sentences now look in the unlabeled pool. As evident by the examples, this new SBD approach helped retain the full context of sentences that had otherwise been split incorrectly. Specifically, these sentences were pieced back together through two operations:

1. String concatenation, where all records concerning the desired clinical domain were concatenated together in the order they appeared in the data set. This essentially resulted in a single, large record where all the previous sentence boundary splits were undone.
2. polyglot's SBD operation on the entire record generated above

During the SBD and word tokenization processes, it was found that the raw data contained some characters that were not able to be encoded by the polyglot package. As a solution to this problem, only characters existing in Python's `string.printable` function + the characters in "ÆØÅÄÖÜæøåäöü" were kept. With this measure, still 99.98% of the data was kept and no apparent flaws in the sentences were found during annotation or data exploration as a by-product of this action.

Before	After
Informeres iht.	Informeres iht. ovenstående notat fra mammakonf.
Mamma: Ingen tumores i ve.	Mamma: Ingen tumores i ve. mamma, kontralateralt ingen tegn til recidiv i hud eller cikatrice.
St.p.	St.p. et c.: I.a.
Pt. henvises til narkoselægerne mhp.	Pt. henvises til narkoselægerne mhp. præoperativ vurdering.

Table 4: The new SBD helps ensure that more sentences appear in full

While the capabilities of polyglot were generally found satisfactory, there were some shortcomings. For word tokenization, there were cases where words separated by a slash "/", were not tokenized into separate words, where it would otherwise have been desirable. For example, ["behandling", "/", "undersøgelse"] would have been preferred over ["behandling/undersøgelse"]. Still, there were cases where this behavior was correct, such as in blood pressure measurements ["121/81"].

For SBD, it was found that perhaps the largest challenge was the writing by the doctors themselves. Oftentimes, doctors would write long stretches of text with no periods, making it difficult to locate the sentence boundary (if there even was one). This resulted in cases with very long sentences.

4.2 ANNOTATION PROCESS

This section will describe the annotation process of this study. Annotation plays a significant role in this project, and has a direct impact on the performance of the models. For that reason, a set of annotation guidelines had to be created to ensure a consistent and reliable annotation. The guidelines can be seen in appendix A.1, along with a data statement in appendix A.7. As expected, the annotation and annotation guidelines represented a considerable challenge, as neither of the annotating researchers in this study had any particular experience with reading health records and only limited knowledge about the clinical domain of breast cancer. This especially influenced the annotation process in terms of time, as many terms had to be looked up. However, many terms reappeared as the annotation progressed, greatly decreasing the time it took to annotate. In this study, a total of 2.346 unique sentences were annotated, comprising 46.227 words and 6.276 events.

Selecting which sentences to annotate was done either randomly or through the active learning query strategies described in section 4.4. For randomly selected sentences, there were no constraints on the amount of sentences that could be annotated at a time. Typically, 100 sentences were annotated at a time. For sentences selected through the active learning implementations, the amount of sentences annotated at a time were restricted to the batch size of each active learning iteration.

Due to the difficulty of annotating clinical text, annotation was initially done collectively by the two annotators. This pair-annotation setup helped ensure that both annotators were aligned in terms of the actual, practical annotation process, in addition to the alignment that the annotation guidelines provided. As the annotators became more aligned, annotation assignments began being divided, enabling up to twice as fast annotation than before. In reality, while the division of assignments did speed up the process, extra time was taken to collectively resolve sentences that were particularly difficult to annotate.

The annotation guidelines developed in this study are inspired by [Styler IV et al. \(2014\)](#) and [Viani et al. \(2019\)](#), and adapted to Danish clinical texts. Developing

guidelines requires that the information of interest is well-defined. This study is interested in *clinical events* and certain *attributes* associated with these events.

4.2.1 Clinical event definition

The THYME annotation guidelines (Styler IV et al., 2014) defines a clinical event as “anything that would show up on a detailed timeline of the patient’s care or life”. This is a simple and easy to understand definition that captures the general idea of a clinical event, yet alone, it does not work well for annotation guidelines. This study presents a more specific list of things that are generally considered clinical events below. These definitions are elaborated on in the annotation guidelines found in appendix A.1.

Clinical events include:

- Tests and treatments
- Problems
- States
- Results and findings of tests that lead to a diagnosis
- Drugs and medication
- Hypothesized/presumed diagnoses
- Headers that represent a test, treatment, or similar
- Verbs that can be considered its own timeline point

This excludes:

- Numeric measurements such as blood pressure and heart rhythm
- Results that do not lead to a diagnosis
- Problem severity/degree
- Generic headers
- Events of legal relevance
- Verbs that only introduce the event

Including and excluding events from the clinical event definition can of course be expected to have some impact on not only the performance of the model, but also the capabilities of the model in terms of what it can, and cannot, be used for by the end users. For this reason, this study highly recommends that annotation guidelines are developed in cooperation with, or at the very least reviewed by, domain experts and end-users. This study had originally set up a meeting with a doctor working in the intersection of health care and artificial intelligence. However, due to the COVID-19 pandemic, this meeting became deemphasized and unfortunately had to be canceled.

4.2.2 Event attributes

The event attributes that this study is interested in are *span*, *polarity*, *document creation time relation* (*DocTimeRel*), and *experiencer*.

In this study, the span of an event refers to the event itself. Since only the head of the event is labeled (see appendix A.1), an event will only ever span over one word. In terms of tagging, this also implies that only “B” and “O” tags are used in this study.

The polarity of an event expresses if the event did or did not occur. Therefore, the polarity of an event will either be positive (it occurred) or negative (it did not occur).

DocTimeRel refers to the point in time the document was created in relation to the event. The possible attributes are *overlap*, *before*, and *after*. Overlap is used for events that are true at the time that the patient was consulted. Before and after are used for events that happened before or after the patient was consulted.

The experiencer attribute expresses if the given event is related to the patient or another person.

4.2.3 Challenges of annotation

As mentioned in section 2.4, annotation can be time-consuming and should preferably be done by domain experts or linguists. This study's work on annotation affirms these statements.

One major challenge faced in this project during annotation was that the annotators had very limited knowledge of clinical domains in general, including breast cancer. This meant that reliably annotating the clinical texts relied on many definition and description look-ups on terms used in the records. This was especially true for terms regarding things such as treatments, surgeries, and medicaments. This greatly increased the time it took to annotate, although the process did speed up as more and more of these terms reappeared. Additionally, many sentences contained Latin, which too demanded several look-ups. In general, the medical jargon and norms used in clinical notes were often hard to interpret as annotators from outside the domain environment. As another example, abbreviations are very commonly used, which were often difficult to decipher without domain knowledge.

Luckily, these difficulties in the annotation process were greatly alleviated through the support of a medical student, who helped by explaining medical terms and deducing the meaning of abbreviations.

Examples of challenging sentences to annotate are given below. Danish is translated into English, while Latin is kept as is.

- (1) Høveltevej diagnosen neoplasma malignum mammae sin..
'The Høveltevej diagnosis neoplasma malignum mammae sin..'
- (2) Ingen perifere palpable lymfomer i collum, periklavikulært eller i aksil bilateralt
'No peripheral palpable lymphomas in collum, periclavicular or in axillae bilaterally'
- (3) Henvisningsårsag 42 årig kvinde, som henvises fra Mammacentret, VS, u.d. neoplasma mal. mammae sin. mhp. neoadjuv. behandling iht.
'Referral cause 42 year old woman, who is referred from the Mamma Center, VS, ut dictum neoplasma mal. mammae sin. for the purpose of adjuvant treatment accordingly'
- (4) St.p. et c.: I.a.
'St.p. et. c.: Nothing abnormal'

4.3 MODEL IMPLEMENTATION

In order to determine the primary model architecture with which to work with in this study, several different models were implemented, trained, tested and evaluated prior to any active learning. These models were based on one of two overall deep learning architectures, namely LSTM and BERT. The exact models that were implemented are shown below.

LSTM based	BERT based (see also section 2.3.4)
BiLSTM	BERT Multilingual
BiLSTM + GloVe	BERT Danish _{v1,v2}
BiLSTM + GloVe + CRF	SciBERT
	Bio-ClinicalBERT

This section will describe how these models were implemented. Among the implemented models, BERT Multilingual was found to perform the best in terms of F1-score. Thus, it was this model that was used for the study's further work, such as active learning and cross-domain testing. For the experiments and results of these models, see section 4.6.1 and section 5.1.

4.3.1 Implementing LSTM based models

Tools

The implementation of the LSTM based models is done using Keras, an open-source deep learning library for Python. It is a high-level neural networks API that allows for fast experimentation (Chollet et al., 2015). In this study, it is primarily used for one-hot encoding, sequence padding, and for sequential model building, training, and testing.

Input data

The input data used for the models are found in the labeled pools described in section 3.3. Specifically, the data used from the labeled pools are the words, sentence lengths, and the labels. A model is only built to classify one type of attribute at a time, and so the implementation allows for specifying a label of interest, such as *span labels* or *polarity labels*.

Sequential models require that the input is represented as numerical values. For word representation, each unique word in the data vocabulary was therefore mapped to a unique integer referring to its embedding.

Label representation is done through one-hot encoding. The label of each word can be represented as a vector of n bits, where n is the number of unique labels. All bits are 0 except the index position of the true ("hot") label, which is 1. Thus, this categorical representation serves as the ground truth that the model outputs are compared to during training and testing. Keras' `to_categorical` function is used to achieve these categorical vectors. An example of this word and label representation transformation is given below.

["Patient", "has", "cancer", "."]	=>	[42, 91, 28, 8]
		0 B-OV B-BE B-AF
		[[1, 0, 0, 0]]
['0', '0', 'B-OV', '0']	=>	[[1, 0, 0, 0]]
		[0, 1, 0, 0]
		[1, 0, 0, 0]]

Prior to these word and label representation transformations, the word and label sequences were padded to contain the same number of tokens. In other words, all sequences (both words and labels) were giving the same length. This length was set according to the longest sentence of the input data. Word sequences were padded with a special padding token, while labels were padded with "O" labels.

Sequential model building

Three different LSTM based models were created for this study, *BiLSTM*, *BiLSTM + GloVe*, and *BiLSTM + GloVe + CRF*. These are all sequential models, meaning they consist of a linear stack of layers. The data passes through these layers sequentially, each layer carrying out a specific role before passing the data to the next layer in the stack. Table 5 gives an overview of the layers of each model.

	BiLSTM	BiLSTM + GloVe	BiLSTM + GloVe + CRF
1	Embedding	Embedding (GloVe)	Embedding (GloVe)
2	BiLSTM	BiLSTM	BiLSTM
3	Dropout	Dropout	Dropout
4	Dense	Dense	Dense
5			CRF

Table 5: Layers in the three sequential LSTM based models

The embedding layer is responsible for mapping a word (its unique numerical value to be exact) to an embedding vector that tries to capture the words semantic meaning (as described in section 2.2). For computing these vector representations, this study experiments with Keras' plain embedding layer and an embedding layer using GloVe vector representations. GloVe vectors were created in this project by training the unsupervised learning algorithm on all breast cancer records for 20 iterations, with a windows size of 8 and an output vector size of 100.

The BiLSTM layer processes the input sequences according to how it is described in section 2.3.2, and ensures that this process is done bidirectionally on the sequences.

The dropout layer helps prevent overfitting by randomly excluding a fraction of the input units for each iteration. Thus, the dropped units are not considered during that training pass. Using dropout makes it more likely that the LSTM learns features that are common across all sequences, instead of overfitting itself to the training set and being unable to correctly classify new, unseen data. If the dropout rate is set to 0.5, the model will randomly hide half of the units for each iteration.

The dense layer connects the input layer to the output layer, by transforming the dimensionality of the data to the number of categorical labels through an activation function. This implementation makes use of the the softmax activation function, which normalizes the output into a probability distribution. That is, for each possible label, the dense layer outputs the probabilities of the model's predictions.

The CRF (conditional random field) layer is a layer that can be added to the bottom of the model stack. Hence, it takes the model's label predictions as input. Through its training, the CRF layer can learn certain constraints from these label predictions. As an example, it may learn some of the constraints of IOB-tagging if such as schema is used, such as: An O-tag is never followed by an I-tag, a sentence always starts with either a B-tag or an O-tag, and that a B-tag and its associated I-tag(s) must have the same attribute.

Hyperparameters

Some hyperparameters were fixed across all three models and across all experiments. Others were part of a grid search hyperparameter optimization process described in section 4.6.1. The most noteworthy fixed parameters are shown in Table 6. These were set based on other similar studies, such as [Tourille et al. \(2017\)](#).

Layer	Hyperparameter	Value
Embedding	input dimensions	vocabulary size
	output dimensions	64 (Keras), 100 (GloVe)
	input length	max sequence length
BiLSTM	units	64
Dropout	rate	0.5
Dense	units	number of unique labels
	activation function	softmax

Table 6: Fixed hyperparameters for the LSTM based models

4.3.2 Implementing BERT based models

Tools

The implementations of the BERT based models were done using Hugging Face’s transformer library ([Hugging Face, nd](#)) which provides access to pre-trained transformer models, and Simple Transformers ([Rajapakse, 2020](#)), which simplifies the transformer building, training, and testing processes. Part of the Simple Transformers repository was modified to fit the task of this study. During a model prediction, the predict function would now also return softmaxed label certainties, which were crucial for being able to include uncertainty as an element in the active learning strategies. The modified repository may be seen at github.com/frwon/simpletransformers.

Input data

The raw input data is similar to the LSTM based models. The input data used for the models are found in the labeled pools described in section 3.3. Specifically, the data used from the labeled pools are the words, sentence lengths, and the labels. A model is only built to classify one type of attribute at a time, and so the implementation allows for specifying a label of interest.

The input data is formatted into a txt file such that it resembles the CoNLL format. Each line consists of a single word and its label, separated by a space. Sentences are separated by an empty line. For example:

```
The 0
patient 0
has 0
cancer B-OV
. 0

CT 0
scan B-AF
tomorrow 0
. 0
```

BERT model building

Building the BERT model itself is simple. Most of the work concerns choosing the desired pre-trained model and setting arguments and hyperparameters, such as learning rate, batch size, and epochs.

Below is a simple example of what a basic BERT model implementation using the Simple Transformers library may look like.

```
labels_list = ["O", "B-OV", "B-BE", "B-AF"]
overall_model = "bert"
exact_model = "bert-base-multilingual-cased"
model = NERModel(overall_model, exact_model, labels=labels_list,
                  args={'num_train_epochs': 4,
                        'train_batch_size': 16,
                        'eval_batch_size': 16,
                        'learning_rate': 5e-5,
                        'max_sequence_length': max_sentence_length})
```

Hyperparameters

Four hyperparameters were set during the BERT model implementations, namely *epochs*, *batch size*, *learning rate*, and *max sequence length*. Learning rate was fixed at $5e-5$ and max sequence length was fixed to match the longest sequence in the input sequences. The number of epochs and batch size were set according to a grid search hyperparameter optimization process described in section 4.6.1.

4.4 ACTIVE LEARNING IMPLEMENTATION

As described in section 2.5, active learning has shown promise in reducing the annotation cost. This section aims to describe how our active learning system was implemented in order to explore this potential.

A high level overview of the processes in the active learning system is illustrated in Figure 10. Broadly speaking, the framework differs from a traditional annotation and query framework by the *iterativeness* and the active querying. Below is a description of the overall flow followed by a further elaboration on the implementation of the two active learning query strategies, uncertainty sampling and diversity sampling.

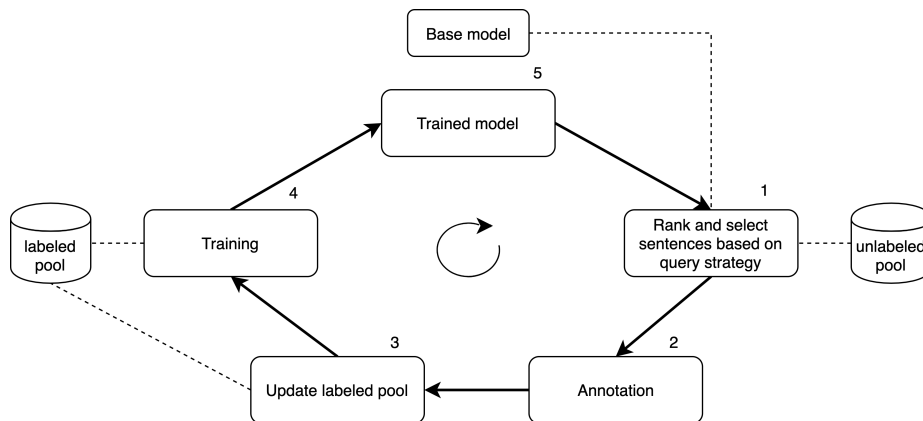


Figure 10: High level illustration of the active learning process

Starting at step 1, the active learning system employs a query strategy to query a batch of unlabeled sentences from the unlabeled pool. For both strategies, the base model is initially used to determine the uncertainty of all sentences in the unlabeled

pool. At step 2, the unlabeled sentences returned by the query are manually annotated as described in the annotation section 4.2. At step 3, the annotated sentences are fed into the respective labeled pool upon which the the model is retrained at step 4. At step 5, a newly trained model is obtained, which is used for ranking and selection in the next iteration.

This process was repeated for both random, uncertainty, and diversity sampling 6 times each with batch sizes of [8, 16, 32, 64, 128, 256] until a total of 512 annotated sentences (including the initial 8 sentences from the base model) were acquired for each strategy in their respective labeled pool. The sampling strategy at step 1 impacts the entire subsequent flow and determines what sentences that are going to be annotated. Below is an elaboration on how uncertainty and diversity sampling were implemented. Both are theoretically described in section 2.5.1.

4.4.1 Implementation of uncertainty sampling

The implementation of uncertainty sampling was centered around the model's predictions. When the model predicts the label of a word in a sentence, the *uncertainty* of that prediction is given by $1 - x_n$ where x_n denotes the softmax value of the most likely label for the word at position n in a given sentence. Predictions that the model is uncertain about are close to 0.5, while predictions that the model is more certain about are closer to 0. This uncertainty score was calculated for all words in a sentence. The final sentence uncertainty was determined by the least confident prediction, i.e. the word with the highest uncertainty within the sentence.

Practically, an uncertainty column was added to the unlabeled pool to store the uncertainty associated with each sentence. The trained model was used to make predictions on all sentences in the unlabeled pool, before the pool was sorted in descending order with respect to the uncertainty. Table 7 shows a sample of the unlabeled uncertainty pool.

ID	Sentence	...	Uncertainty
93	"Patoanatomisk diagnose/plan.."	...	0.421
7	"Calcium, magnesium, fosfat, b.."	...	0.302
43	"CT-scanning viste tegn på b��d.."	...	0.158

Table 7: Sample of an unlabeled pool with descending uncertainties

The most uncertain sentences were then obtained from the pool by sampling the top k sentences in the pool, where k is the batch size of the query. For example, a query with $k = 2$ would return the sentences with ID 97 and 7 from Table 7.

4.4.2 Implementation of diversity sampling

In the implementation of diversity sampling, the unlabeled pool was firstly split into clusters representing *diverse* groups of semantically similar sentences. This was accomplished by the following steps:

Firstly, sentence embeddings for all sentences in the unlabeled pool were generated using the BERT based tool *bert-as-service* (Xiao, 2018) that maps sentences to 768 dimensional vectors. Second, to overcome the curse of dimensionality, PCA was used to reduce the sentence embeddings to 32 dimensions, keeping 80% of the explained variance from the original embeddings (see appendix A.3). The sentence embeddings and their clustering tendencies are visualized in Figure 11 using t-SNE.

Secondly, k-means clustering was implemented to group the sentences into clusters. Through the elbow method, it was determined that the optimal numbers of clusters was 8 (see appendix A.4). The silhouette values of the 8 clusters are visual-

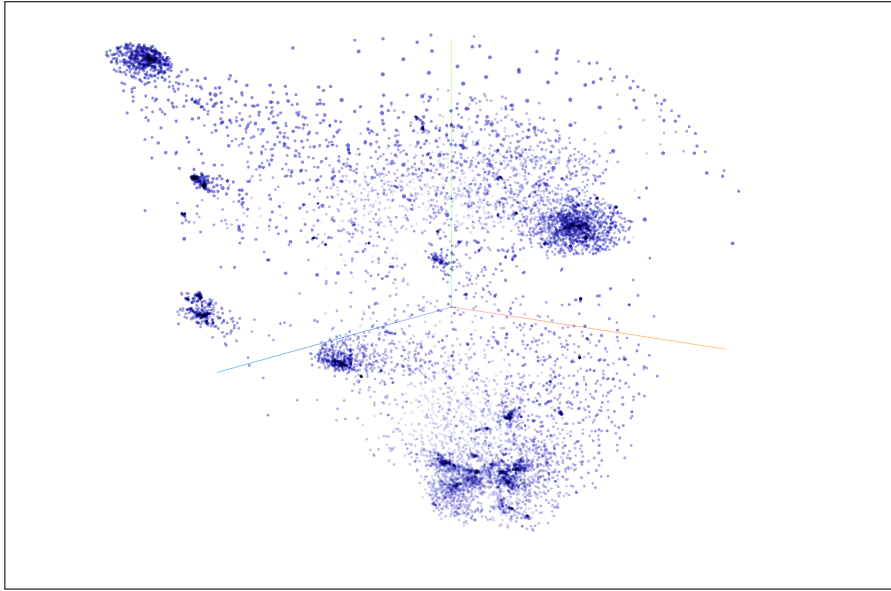


Figure 11: t-SNE visualization of the sentence embeddings

ized in Figure 12. The average silhouette score suggests that many of the sentences within each cluster are close to the decision boundary of the neighboring clusters. However, very clear decision boundaries were not expected, as grouping sentences into semantically similar clusters is a difficult task, where borderline cases must be expected.

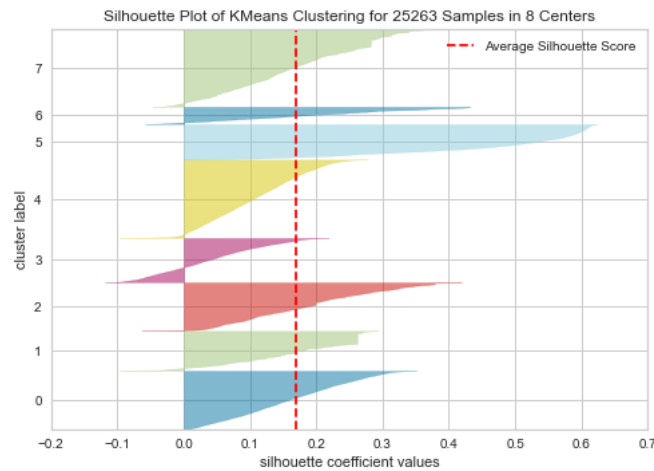


Figure 12: Silhouette values for the clusters

When examining the clusters, it was generally found that:

- Long patient summaries were grouped in cluster 5. The average number of tokens pr. sentence within the cluster was 83 words.
- Medication prescriptions, often in the form of short notes, were grouped together in cluster 6.
- Cluster 4, the largest cluster, contained a variety of notes, many which were a mix of clinical findings and treatment suggestions.
- Cluster 7, the second largest cluster, contained notes with a diverse range of content. The notes were in general short with an average of 6.2 words per sentence.

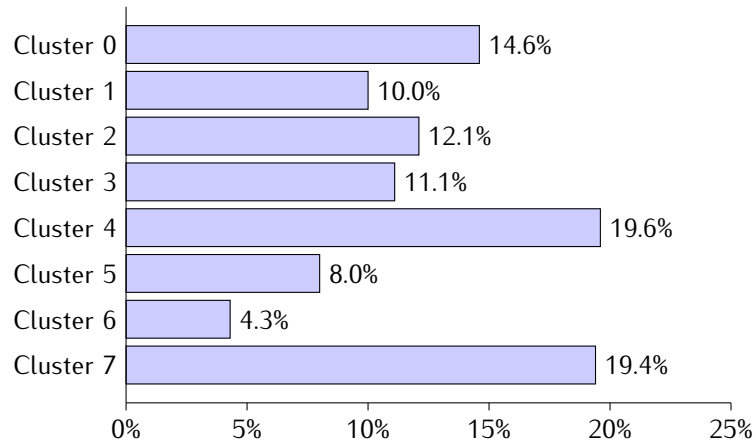


Figure 13: Percentage of all sentences in the unlabeled pool for each cluster

Finally, the cluster probabilities were used to ensure a representative cluster sampling, by keeping the original cluster proportions in the diversity pool. These proportions are visualized in Figure 13. For example, if 100 sentences were queried using diversity sampling, it would return 20 sentences from cluster 4 (19.6 %), 19 sentences from cluster 7 (19.4%), 10 sentences from cluster 1 (10 %), and so on.

In practice, the diversity sampling was done by appending three columns, *Uncertainty*, *Cluster* and *Cluster probability* to the unlabeled diversity pool. A sample of the pool can be seen in Table 8. Here, the uncertainty values are the same as described in section 4.4.1. The *cluster* represents the cluster that a sentence has been assigned to, and the *cluster probability* column represents the probability for that cluster.

ID	Sentence	...	Uncertainty	Cluster	Cl. prob.
11	"Tumor hormonreceptor ska.."	...	0.397	4	0.196
298	"Hormonreceptor positiv for.."	...	0.105	4	0.196
55	"TUL2 Tillægsdiagnose: vens.."	...	0.326	3	0.111

Table 8: The unlabeled pool with sentence uncertainties, cluster assignments and probabilities

The sentences within each cluster were sorted with respect to the uncertainty, meaning that the most uncertain sentences in each cluster were returned. For example, if one sentence was to be queried from cluster 4 in Table 8, the sentence with ID 11 would be returned as it has the highest uncertainty within that cluster. Hereby, when sampling from the unlabeled diversity pool, the batch contained a *diverse* set of the most uncertain sentences from each of the 8 clusters.

4.4.3 Iterations

The two query strategies described above were applied iteratively, i.e. they were applied several times to account for the adjustments in the model's learning, and also in order to measure the performance each strategy over time. To be able to compare the active learning tests to the experiments by [Chen et al. \(2015\)](#), the same batch sizes were used. The batch size refers to the number of sentences that are in a query within one iteration. For this strategy, the batch size grows with the number of iterations following the exponential function: 2^{i+2} , where i is the iteration number. This approach was followed up to a batch size of 256 sentences, meaning that 6 iterations were done with the following batch sizes [8, 16, 32, 64, 128, 256].

4.5 EVALUATING MODELS

The section will describe the metrics used to evaluate the models, namely precision, recall and F1-score. These metrics are useful to evaluate problems where the goal is to find a small subset of items in a large set of items (Derczynski, 2016). These metrics also allowed us to compare the results of this study to other similar studies that use the same metrics, such as Bethard et al. (2016) and Viani et al. (2019). The accuracy metric on a token level would be ill-suited for this problem, since predicting the vast majority of the tokens as non-entities would lead to a misleading high accuracy.

The precision metric measures the *quality* of the model, by indicating how correct the model is on predicted items. The recall metric measures the *quantity* of the model, indicating how many of the true items are correctly predicted. F1-score is the harmonic mean of these two measurements. It indicates the balance between precision and recall on a scale from 0 to 1, where 1 is considered a perfect score (the model has correctly predicted all entities). The three metrics are given by:

$$\text{Precision} = \frac{|S \cap H|}{|S|} \quad \text{Recall} = \frac{|S \cap H|}{|H|} \quad F_1 = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

Where S denotes the set of items predicted by the system and H denotes the set of true items.

In addition to the F1-score described above, also called F1-score (entity), we also report the F1-score (surface), inspired by Derczynski et al. (2017). The F1-score (surface) only counts each unique correctly classified item once, making it a feasible measurement for indicating how well a model recognizes a diverse set of items.

Sentence	["CT", "skanningen", "viste", "ingen", "spredning"]				
True tags	["O", "B", "O", "O", "B"]				
Predicted tags	["O", "O", "O", "O", "B"]				

Table 9: An example of OB tagging

Table 9 shows an example of a sentence with true tags and predicted tags. Using the metrics just described:

$$\text{Precision} = \frac{1}{1} = 1$$

$$\text{Recall} = \frac{1}{2} = 0.5$$

$$F1 - \text{score} = \frac{2 * (1 * 0.5)}{(1 + 0.5)} = 0.666$$

4.6 CONDUCTING EXPERIMENTS

This section provides an overview of the conducted experiments. It seeks to describe the testing environment, explain how the tests were conducted, and elaborate on the different strategies that were used.

All tests were conducted on a server provided by the IT University of Copenhagen. The server is running on a GNU/Linux 64-bit machine with two NVIDIA GeForce GTX 1080 Ti GPUs.

4.6.1 Base model experiments

The base model experiments were a series of experiments conducted in order to determine which deep learning model and hyperparameters should be the primary ones used in this study. This decision was based almost exclusively on the F1-scores of the models. That is, the model architecture with the highest F1-score in these experiments was the one used for active learning, the final model, and the cross-domain model. These experiments found the BERT Multilingual model to be the best-performing, with an F1-score of 86.6% after 4 epochs and a batch size of 8. For the results on these experiments, see section 5.1.

The experiments were conducted in the form of a hyperparameter grid search, i.e. an exhaustive search with manually specified parameters. This was possible due to the relatively short training durations. The tested hyperparameters were *epochs* and *batch size*. Since two different overall architectures were tested, LSTM and BERT, the hyperparameter values in the grid search were different for these two approaches. The exact hyperparameters in the grid search for each architecture is shown in Table 10. It is clear from the table that the LSTM and BERT based models differ greatly in their amounts of epochs. This is due to the pre-trained nature of the BERT models, which results in fine-tuning requiring much fewer epochs. In fact, [Devlin et al. \(2018\)](#) found that 2-4 epochs worked well for the tasks that they tested.

	LSTM	BERT
Epochs	[25, 50, 100, 150]	[3, 4, 5]
Batch size	[8, 16]	[8, 16]

Table 10: Specified parameters of the grid search

With three LSTM based models and five BERT based models, a total of 54 models were trained during these experiments. All models were exposed to the same data set of 512 randomly selected sentences. The exact training/test setup was a 5-fold cross validation (see Figure 14). That is, 5 folds of the data set was created, each fold consisting of a unique 80/20 split of the data. The splits are unique in the sense that there is never an overlap in any of the test splits. This cross validation technique ensures that the model is trained and tested on all the data, and thus helps give more a fair picture of the model's performance. The reported evaluation metrics are the averages of the scores for each fold.

It is important to note that these experiments were conducted only with the *span labels* designated as the labels of interest. For experiments and results of the final model, including all other attributes, see section 4.6.3 and section 5.3.

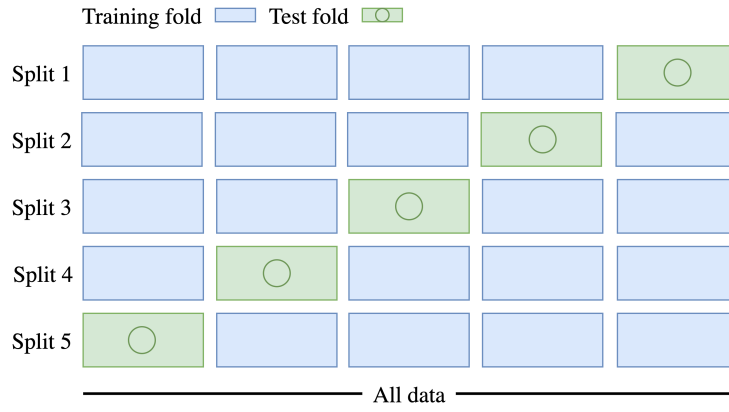


Figure 14: A 5-fold cross validation setup

4.6.2 Active learning experiments

The active learning experiments were conducted in the following way: First, a test set was obtained by randomly sampling and annotating 640 sentences from the unlabeled pool. Secondly, random sampling and the two active learning query strategies were tested by 6 rounds of sampling, annotation, training and testing. For every iteration, batch of the unlabeled pool was queried, annotated, and added to the labeled pool for the respective strategy. Thus, 3 labeled pools were created, *random labeled pool*, *uncertainty labeled pool*, and *diversity labeled pool*. These pools were shown in Figure 9.

In each iteration, the training was done by training 5 BERT Multilingual models, with 4 epochs and a batch size of 16, on the labeled pool for the given strategy. The resulting models were then tested on the test set and an average of the performance metrics was calculated for the evaluation. To rank and select new sentences from the unlabeled pool in the next iteration, the model achieving the highest F1-score among the 5 trained models was used.

All three query strategies were initialized using a base model, as described in section 4.4. This base model, which is used to rank and select the first batch of sentences, was trained on 8 randomly selected sentences.

The results of the random sampling strategy were used as a comparison baseline for the two active learning strategies.

4.6.3 Final model experiments

Experiments were conducted to establish the capabilities of a model trained on all labeled breast cancer data, regardless of query strategy. This included doing a grid search hyperparameter optimization with the same specified parameters as in section 4.6.1. Furthermore, performance on all attributes (span, polarity, DocTimeRel, and experiencer) was evaluated.

A total of 2,176 breast cancer related sentences were annotated (the union of the three labeled pools and the test set), 2,090 of which were unique. Any duplicate sentences were removed prior to training.

Training and testing was done through a 5 fold cross validation setup, similar to the setup explained in section 4.6.1.

4.6.4 Cross domain experiments

Inspired by SemEval 2017 Task 12 (Bethard et al., 2017), where the problem of domain adaptation was introduced, unsupervised domain adaptation experiments were conducted on another domain than breast cancer. The motivation behind these experiments was to explore how well a model trained on one domain (breast cancer) performs when tested on another clinical domain. The guidelines in section 3.2 were used to select a new domain for the experiments.

Pneumonia was chosen as the domain for the experiments since it is represented with a sufficient amount of sentences in the data set, consists of notes with a substantial amount of words and less so numbers, and because it is a societal problem, especially in the light of the COVID-19 pandemic.

The experiments were conducted in the following way: First, 256 sentences related to pneumonia (ICD10 code J18) were randomly sampled from the data set. These sentences were then annotated, forming the test set for the cross domain experiments. Lastly, the final model, obtained through the experiments described in section 4.6.3, was tested on this test set and evaluated using the standard metrics. To gain insight into the difficulty of the task, an evaluation with on the intersection of the events from the training and test sets was also done.

5 | RESULTS

5.1 BASE MODELS

This section will describe the results of the experiments found in section 4.6.1. As mentioned, the goal of the experiments was to determine which model architecture and hyperparameters this study should continue to work with. The results of the best performing model of each of the eight tested models are seen in Table 11 (sorted by best to worst performance in terms of F1-score). For all base model experiment results, see appendix A.2.

Model	Epochs	Batch size	F1-score	Precision	Recall
BERT Multilingual	4	16	0.866	0.865	0.867
Danish BERT _{v1}	3	8	0.861	0.855	0.867
Danish BERT _{v2}	5	8	0.852	0.872	0.833
BiLSTM + GloVe	100	8	0.834	0.859	0.811
BiLSTM + GloVe + CRF	100	8	0.833	0.851	0.817
Bio_ClinicalBERT	5	8	0.799	0.802	0.797
BiLSTM	150	8	0.799	0.871	0.738
SciBERT	5	8	0.793	0.792	0.795

Table 11: The best performing models of each tested architecture

The results show that it is the BERT Multilingual model that performs the best in terms of F1-score, and is thus the architecture that was used for active learning, the final model, and the cross-domain model. Considering this project is working with Danish health records, it is somewhat surprising that both Danish BERT models are outperformed by this model. However, it is evident when examining the health records that a significant portion of the language used in these records cannot be considered regular Danish, and many of the terms used are not Danish at all. The Danish BERT models are pre-trained on what can be considered "normal" Danish texts, that you would find ordinarily. Meanwhile, the Danish health records include unusual words such as technical clinical terms and medication names. Also, a relatively significant portion of the text in the records is in Latin. The fact that the language in the records is so varied and unusual could help explain the BERT Multilingual's superior performance, as it has been pre-trained on many different languages, including Danish and Latin, which may increase the likelihood of it having been exposed to the unusual terms found in the health records. Still, BERT Multilingual only performed 0.6% better than Danish BERT_{v1} and 1.4% better than Danish BERT_{v2}, meaning that they cannot completely be disregarded as potential model architectures in future work.

For the BERT Multilingual model, it was found that 4 epochs and a batch size of 16 performed the best. Hence, it was these hyperparameters that was used for training during the active learning iterations.

5.2 ACTIVE LEARNING MODELS

This section provides an overview of the results that were obtained in the active learning experiments. The results are used to give an indication of whether annotation efforts can be saved through active learning.

Table 12 shows the change in F1-score, precision, and recall for each of the three query strategies over the number of sentences in each iteration. The table shows that both uncertainty and diversity sampling ultimately perform better than random sampling, and that diversity sampling achieves the highest F1-score, with a score of 87.0% compared to 85.9% and 84.8% for uncertainty and random sampling. This is equal to a relative error reduction of 7.24% and 14.47%. Interestingly, the average recall scores for the diversity based models are significantly higher than for the other two strategies, meaning that the diversity based models in general are better at capturing labeled events. The precision score, on the other hand, is slightly lower compared to the other strategies. However, in the medical world, a high recall is often more desirable than a high precision, as it is safer to have more false positives than false negatives.

Number of sentences	8	16	32	64	128	256	512
Random sampling							
F1-score	0	0.183	0.547	0.692	0.784	0.810	0.848
Precision	0	0.975	0.773	0.762	0.862	0.902	0.908
Recall	0	0.103	0.425	0.635	0.720	0.734	0.796
Uncertainty sampling							
F1-score	0	0	0.499	0.725	0.806	0.841	0.859
Precision	0	0	0.909	0.843	0.867	0.882	0.898
Recall	0	0	0.345	0.635	0.754	0.804	0.823
Diversity sampling							
F1-score	0	0	0.381	0.677	0.805	0.845	0.870
Precision	0	0	0.698	0.741	0.807	0.857	0.881
Recall	0	0	0.265	0.624	0.807	0.834	0.859

Table 12: Evaluation results of models based on query strategy

Figure 15 shows the change in F1-score for each active learning strategy over the number of sentences in the training set. The two first iterations have been omitted as they are considered "warm up" rounds. The graph indicates that the learning curve for diversity sampling is steeper than for uncertainty and random sampling. Over the 5 iterations, diversity sampling seems to be the strategy that grows the most per iteration, implying that this strategy leads to the highest increase in F1-score per annotated sentence.

The F1-score (surface) is also reported to examine how well the respective models perform in recognizing unique events. Table 13 provides an overview of this performance for the three sampling strategies. The results suggest that models trained on sentences selected by diversity or uncertainty sampling overall perform better than models trained on randomly selected sentences selected. Of the two active learning strategies, it is clear that diversity sampling performs the best in recognizing a diverse set of events with a 7.5% increase in F1-score (surface) from random sampling.

Another interesting measurement that can be helpful when determining the effect of active learning and the different sampling strategies is the quality of the sampling batches. That is, the quality of the returned set of sentences from each sampling strategy. Here, we measure *quality* as the number of unique entities and

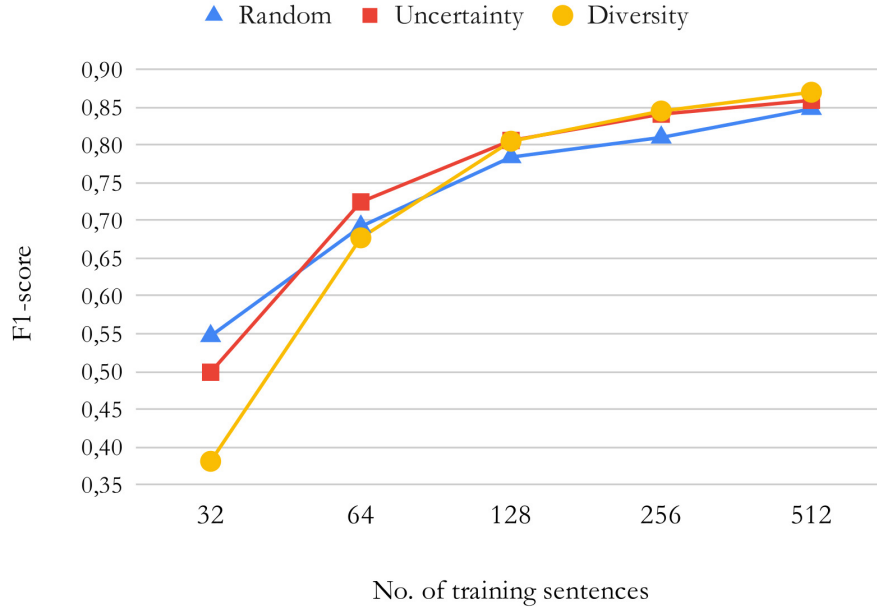


Figure 15: F1-score for each query strategy according to number of sentences trained on

Query strategy	F1-score (entity)	F1-score (surface)
Random sampling	0.848	0.670
Uncertainty sampling	0.859 +1.3%	0.710 +5.9%
Diversity sampling	0.870 +2.6%	0.720 +7.5%

Table 13: F1-scores (entity and surface) for each query strategy with percentage increase from random sampling score

the total number of words in that batch. A batch of high quality would contain a high number of unique entities but few words. Figure 16 illustrates how both diversity and uncertainty sampling significantly outperform random sampling, by having more unique entities in all of the last 4 batches. The number of total entities per batch is also better for diversity and uncertainty. This is shown in appendix A.5.

Based on these results, it is tempting to draw the conclusion that diversity sampling is superior to the other sampling strategies, especially random sampling. The following results however, suggest that this is not necessarily the case. Figure 17 shows the total number of words annotated for the respective strategies versus the number of training sentences. The plot reveals an interesting finding - that random sampling overall has fewer words per batch compared to the other sampling strategies. Specifically, a total of 9.424 words were annotated for random sampling compared to 10.559 for diversity sampling and 12.221 for uncertainty sampling in the 6 iterations, meaning that significantly fewer words were annotated with random sampling. The same goes for the number of unique words, which is shown in appendix A.6.

This implies that if the annotation cost is measured by number of words instead of number of sentences, we can divide the total number of annotated words for each strategy by their respective F1-scores, to get a number that indicates where to get the most "bang for the buck" in terms of cost per F1-score. Surprisingly, random sampling is "cheaper" than the other two strategies with a cost of 110.2 words per F1 point compared to 142.2 and 121.4 words for uncertainty and diversity, respectively.

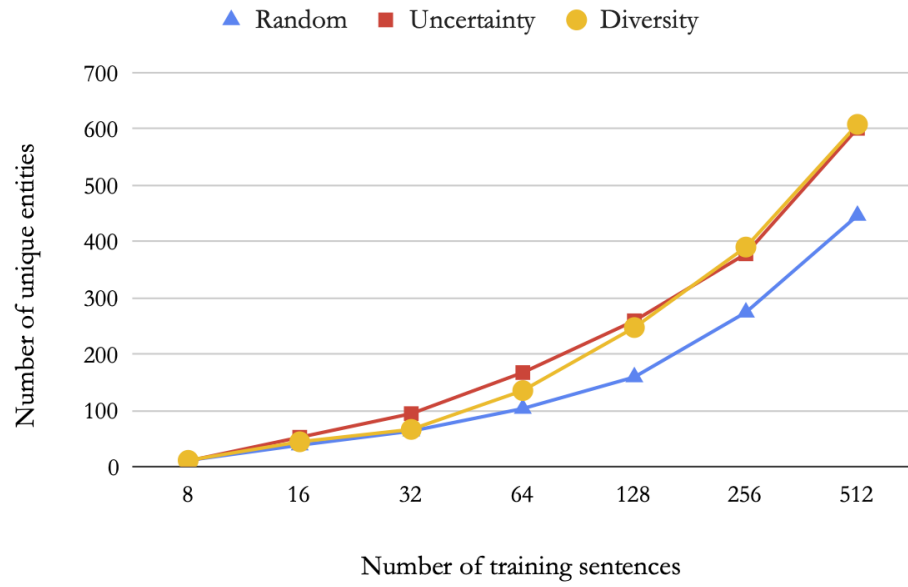


Figure 16: Number of unique entities for each query strategy according to number of sentences trained on

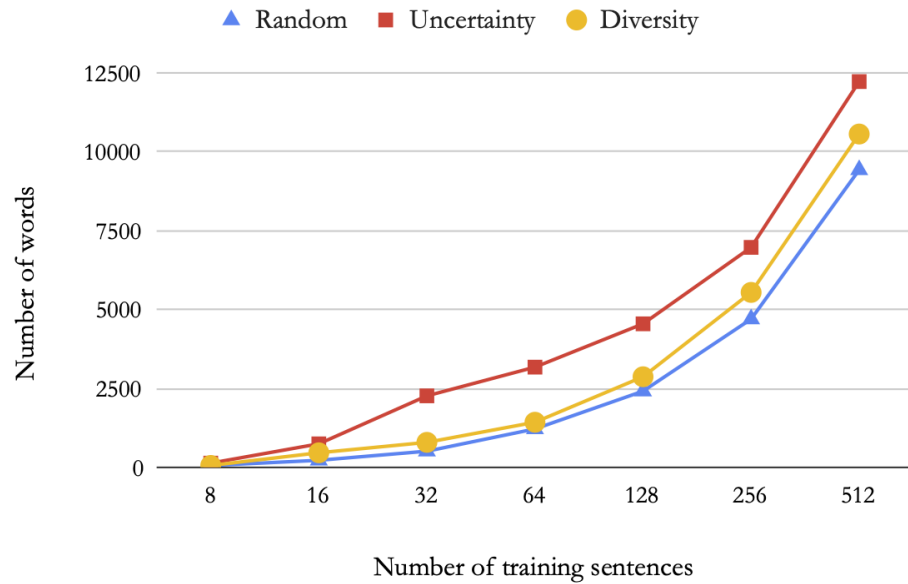


Figure 17: Number of words for each query strategy according to number of sentences trained on

5.3 FINAL MODEL

This section will describe the results of the experiments found in section 4.6.3. The goal of these experiments was to determine how well a Danish clinical event extraction model performs, having been trained on all labeled data (2.090 sentences). The results for the model's performance on span labels are shown in Table 16.

Model	Attribute	F1-score	Precision	Recall
BERT Multilingual	Span	0.875	0.865	0.885

Table 14: Model performance on span labels (epochs = 4, batch size = 16)

With an F1-score of 87.5% when recognizing event span, the final model is the best performing model created in this study. Compared to similar studies for other languages, such as Viani et al. (2019) and Lee et al. (2016) who reach F1-scores of 90.1% and 90.3% respectively, this model is slightly outperformed. Still, it is important to remember that the model in this study has been trained on significantly less data. In fact, it is generally difficult to do a comparison to other studies, which will be discussed in chapter 6.

Interestingly, the model has a lower precision than any of the three active learning models (0.908, 0.898, 0.881), yet it has a much higher recall compared to these (0.796, 0.823, 0.859). In short, when recognizing event spans, the model is now more willing to predict the occurrence of an event, making it wrong slightly more often. However, it does mean that the model recognises a larger part of the actual events.

Succeeding in recognizing event spans alone however, is not sufficient when developing a clinical event extraction system for discovering long term patterns and correlations in health records. Here, other attributes play an important role as well. The model's performance on the attributes *polarity*, *DocTimeRel*, and *experiencer* are shown in Table 16. These attributes are also evaluated in terms of weighted F1 score. This means that the scores for each possible label are now weighted according to their support. This is particularly useful during instances of high class imbalance, which is the case for the attributes in this study, especially for the *experiencer* labels (Table 15).

Attribute	Support	Percentage
Polarity		
POS	5321	92%
NEG	488	8%
DocTimeRel		
OV	5124	88%
BE	349	6%
AF	336	6%
Experiencer		
PT	5794	99.7%
OT	15	0.3%

Table 15: Class balance for each multi-class attribute (total of 5.809 events)

In terms of the polarity attribute, the model performs well regardless of class imbalance. This is evident by the closeness of the F1 and weighted F1 scores. The performance on polarity labels are slightly below the performance on span labels, meaning the model does struggle more, likely because the classification problem is no longer binary.

Model	Attribute	F1-score	Precision	Recall	Weighted F1
BERT Multilingual	Polarity	0.855	0.857	0.855	0.866
	POS	0.868	0.857	0.878	
	NEG	0.843	0.856	0.832	
BERT Multilingual	DocTimeRel	0.611	0.627	0.613	0.783
	OV	0.827	0.818	0.837	
	BE	0.530	0.491	0.580	
	AF	0.494	0.510	0.484	
BERT Multilingual	Experiencer	0.434	0.429	0.439	0.865
	PT	0.868	0.858	0.879	
	OT	0.000	0.000	0.000	

Table 16: Model performance on other attributes (epochs = 4, batch size = 16)

For the DocTimeRel attribute, the model struggles, with an F1-score of 0.611. However, this is the only attribute with three different classes. Comparing the F1 and weighted F1 scores, it is clear that the large class imbalance affects its performance. The model is decent at predicting OV-labels, yet struggles with the other two underrepresented classes.

The model struggles even more so with experiencer labels, with an F1-score of 0.434. This however, is very unsurprising given the significant class imbalance. The model has only been trained on at most 15 OT-labels and likely less at a time due to the 5-fold cross validation setup. There are simply too few cases in the data where an event is related to an individual other than the patient, meaning it cannot be expected that the model could reasonably predict these.

5.4 CROSS-DOMAIN MODEL

The cross-domain experiments aimed to examine how well our model, trained on breast cancer sentences, performed when tested on sentences from the pneumonia domain. This section will describe these results.

Figure 18 compares the F1-scores of a model trained and tested on breast cancer and a model trained on breast cancer but tested on pneumonia. The figure shows that the model performance drops 5.7 percentage points in F1-score and 5.2 percentage point in F1-score (surface) when the test domain changes.

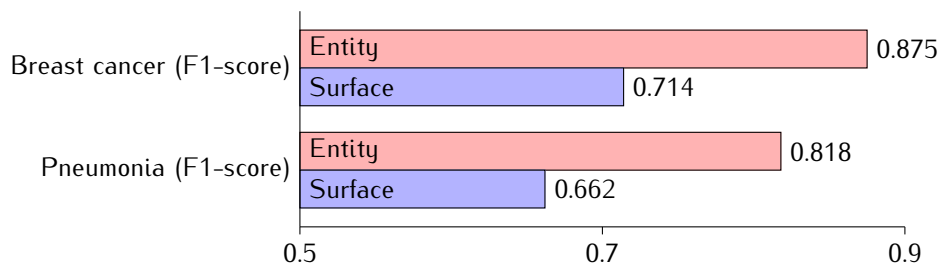


Figure 18: Bar chart of F1-scores (entity and surface) for breast cancer and pneumonia tests

The events in the two data sets were compared to explore the similarity of the two domains. This was done to get an idea of the task's complexity, assuming that a large event overlap in the two domains leads to a easier task for the model.

When comparing the events, 5,809 breast cancer events (2,090 sentences) and 467 pneumonia events (256 sentences), it was found that 53% of the events in the pneumonia test set also exists in the breast training data set. This means the the model

has been trained on a substantial amount of the events in the pneumonia test set, which can help explain the relatively high cross-domain performance. Additionally, there are certain structural factors in Danish EHRs that might further explain this performance. This is discussed in chapter 6.

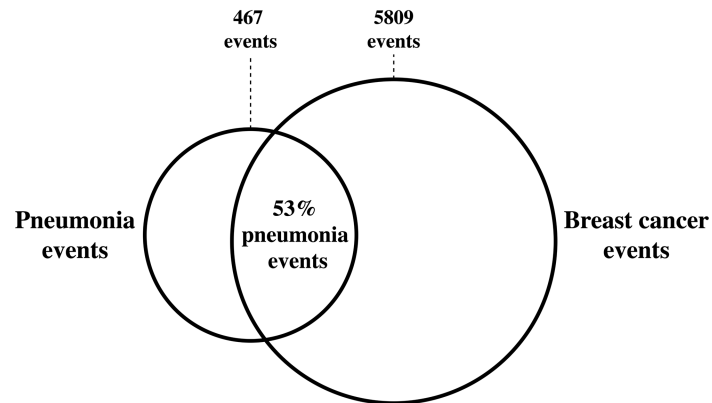


Figure 19: The intersection of events in the breast cancer domain and pneumonia domain contains 53 % of all events in the pneumonia data set.

6 | DISCUSSION

This section aims to critically evaluate and discuss the applied approaches in the project. Furthermore, it includes our reflections on future research.

The process of annotation was challenging in the sense that the data set contained a significant amount of words and phrases written in a clinical language that includes medical terms and words in Latin, completely new to any of us. In our approach to alleviate the annotation process and build a reliably annotated corpus, a set of guidelines were developed. These guidelines were heavily inspired by the THYME guidelines which were developed for EHRs written in English. It is unclear to us to what extent these guidelines should be adjusted to handle the syntactic differences between English and Danish and meet the expectations of the Danish Healthcare System. During the project, two doctors were contacted to review the guidelines. Unfortunately, due to the COVID-19 pandemic, our appointments were cancelled. For large scale studies on this topic, we suggest that guidelines are reviewed by domain experts and that the practical annotation is likewise done by people with domain knowledge, for example medical students.

In the evaluation of the active learning system and the two query strategies, uncertainty and diversity sampling, the system performance is reported as F1-scores and the number of words and unique entities in each batch. In our attempt to answer the question whether active learning *can* reduce annotation cost, the answer comes down to the definition of cost. If the annotation cost is measured in number of annotated sentences, a diversity based active learning system can improve the performance of a model compared to random or uncertainty sampling. Our results show that diversity samples contain more unique events from which the model benefits, achieving a noticeably higher F1-score. If cost, on the other hand, is measured as the number of annotated words, the same conclusion cannot be drawn. Here our results show that random sampling achieves the highest increase in F1-score per annotated word, suggesting that random sampling is where you get the most 'bang for the buck' in terms of increase in F1-score per word. Since none of the active learning algorithms penalize long sentences, one obvious adjustment would be to implement a weight that accounts for this "issue". Based on our annotation experience however, we do not consider that as a novel solution since long sentences might contain events of a specific type that are more frequent in long sentences. Instead, we propose that for future studies, a measurement of annotation time is included in the notion of annotation cost, similar to [Chen et al. \(2017\)](#).

Overall, active learning does show promise in developing more robust models, but more non-simulated studies with a larger focus on annotation cost are needed to find the most appropriate and efficient sampling algorithm.

When comparing the performance of the final model to other studies, some important differences should be mentioned. First, both the training and test sets are smaller for this study compared to other studies on clinical event extraction. For instance, this study has a total of 2.090 sentences, 42.754 tokens, and 5.809 events, compared to for example 38.890 events in SemEval 2016 ([Bethard et al., 2016](#)) and 4.288 sentences (57.263 tokens) in [Viani et al. \(2019\)](#). Secondly, the quality of the data, i.e. the the labeling from the annotation, might not be the same for this study as for the other studies, where the annotation was done by domain experts. Also, while the tagging scheme used in this project, ["O","B"] tags, is the same scheme as in SemEval ([Bethard et al., 2016, 2017](#)), it differs from some of the other stud-

ies, such as [Viani et al. \(2019\)](#) and [Hakala and Pyysalo \(2019\)](#), which are based on ["I","O","B"] tags. Finally, the structure of Danish EHRs might be significantly different from EHRs in English, Italian, Spanish, or Chinese. From our talks with a Danish medicine student, we found that doctors in Denmark are encouraged to write notes that follow a specific structure. This was also evident during annotation, where we found that some sentences, especially long sentences, seemed to be based on a certain template. Whether this is standard practice in other languages is unclear. Based on these differences, it should be highlighted that our final results only are roughly comparable to other studies on event extraction.

During the cross-domain experiments, the structural similarity across some sentence was also found to be present in the pneumonia data set. This similarity might imply that transfer learning across domains is "easier" in Danish compared to other languages. This might also explain why this study only saw a 5.7 percentage point drop in F1-score when testing on another domain, and not a 20 percentage point drop as in other studies ([Bethard et al., 2017](#)).

In terms of future work, we see several interesting areas in relation to this study. Better results could perhaps be reached by experimenting with a combination of deep learning architectures, which may include Flair embeddings ([Akbik et al., 2018](#)). The quality of our sentence-clustering could potentially also be further improved, by exploring sentence embeddings from other layers in BERT and measuring similarity with cosine similarity instead of the euclidean distance. Last but not least, pre-training a BERT model on Danish clinical data such as the one used in this project, combined with data from online medical resources such as [sundhed.dk](#) and [promedicin.dk](#), could not only lead to better results, but also enable generation of artificial health records. In a recent study by [Ive et al. \(2020\)](#) this technique was proven to be feasible, showing potential in the health care domain, leading to models achieving comparable results to models trained on original data. This could make Danish clinical data more available and accelerate research in the field of Danish clinical NLP.

7 | CONCLUSION

The ambition for this study was to develop a system capable of recognizing clinical events from Danish electronic health records. Succeeding in this requires a significant amount of labeled data, of which none exists in Danish. For this reason, this study also aimed to investigate the potential of reducing annotation costs through active learning. These tasks were explored using a subset of Danish electronic health records within the domain of breast cancer.

Initially, this study examined several deep learning architectures, in order to determine the most promising one for clinical event extraction. Here, it was found that the pre-trained model, BERT Multilingual, was superior, outperforming other models such as Danish BERT and LSTM.

The process of annotating clinical data is associated with a high cost, which active learning has shown to reduce in simulated studies. This study sought to explore active learning in a non-simulated environment. For measuring its effect, two active learning strategies, *uncertainty sampling* and *diversity sampling*, were implemented and compared to traditional random sampling. Here, it was found that both uncertainty sampling and diversity sampling outperformed random sampling, ultimately achieving a higher F1-score on the same amount of training sentences. The percentage increase compared to random sampling was 1.3% and 2.6% for the two active learning strategies respectively. This corresponds to error reductions of 7.2% and 14.5%. However, both active learning strategies generally selected sentences containing more words, resulting in random sampling being less costly in terms of the amount of words that had to be annotated. The cost saving potential of active learning therefore remains unclear, and more research is required.

To determine the capabilities of a Danish clinical event extraction system, a model was trained on all labeled data, corresponding to 2,090 sentences, 42,754 words, and 5,809 events. The model reached an F1-score of 87.5%, precision of 86.5%, and recall of 88.5%, when recognizing *event spans*. For other attributes, *polarity*, *DocTimeRel*, and *experiencer*, the model generally struggles due to the significant class imbalance. For each of the attributes respectively, it achieves an F1-score of 85.5%, 61.1%, and 43.4%. Still, we argue that the model's overall performance proves the feasibility of a large scale Danish clinical event extraction system based on deep learning.

Finally, the impact on the model's performance when changing the test domain was investigated. For this task, a test set of 256 sentences, 3,473 words, and 467 events within the pneumonia domain were annotated. When tested on this domain, the model saw an F1-score drop of 5.7 percentage points to a score of 81.8%. While the performance of the model was expected to decrease more significantly, structural similarities in Danish health records, and a large event overlap between the training and test sets, may help explain why that was not the case.

These findings prove that events can correctly and efficiently be extracted from Danish electronic health records, which we see as the foundation of the long term goal of utilizing large amount of Danish clinical text to improve health care.

BIBLIOGRAPHY

- Akbik, A., D. Blythe, and R. Vollgraf (2018). Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 1638–1649.
- Al-Rfou, R. (n.d.). polyglot GitHub repository. <https://github.com/aboSamoor/polyglot>.
- Alammar, J. (2018a). Illustrated BERT [Blog post]. <http://jalammar.github.io/illustrated-bert/>.
- Alammar, J. (2018b). The Illustrated Transformer [Blog post]. <https://jalammar.github.io/illustrated-transformer/>.
- Alsentzer, E., J. R. Murphy, W. Boag, W.-H. Weng, D. Jin, T. Naumann, and M. McDermott (2019). Publicly available clinical bert embeddings. *arXiv preprint arXiv:1904.03323*.
- Arora, S., Y. Liang, and T. Ma (2016). A simple but tough-to-beat baseline for sentence embeddings.
- Bahdanau, D., K. Cho, and Y. Bengio (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.
- Baum, L. E. and T. Petrie (1966). Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics* 37(6), 1554–1563.
- Beltagy, I., K. Lo, and A. Cohan (2019). Scibert: Pretrained language model for scientific text. In *EMNLP*.
- Bender, E. M. and B. Friedman (2018). Data statements for natural language processing: Toward mitigating system bias and enabling better science. *Transactions of the Association for Computational Linguistics* 6, 587–604.
- Bengio, Y., R. Ducharme, P. Vincent, and C. Jauvin (2001). A neural probabilistic language model. *Journal of machine learning research* 3(Feb), 1137–1155.
- Bethard, S., G. Savova, W.-T. Chen, L. Derczynski, J. Pustejovsky, and M. Verhagen (2016). Semeval-2016 task 12: Clinical tempeval. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pp. 1052–1062.
- Bethard, S., G. Savova, M. Palmer, and J. Pustejovsky (2017, August). SemEval-2017 task 12: Clinical TempEval. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, Vancouver, Canada, pp. 565–572. Association for Computational Linguistics.
- Chen, Y., T. A. Lask, Q. Mei, Q. Chen, S. Moon, J. Wang, K. Nguyen, T. Dawodu, T. Cohen, J. C. Denny, et al. (2017). An active learning-enabled annotation system for clinical named entity recognition. *BMC medical informatics and decision making* 17(2), 82.
- Chen, Y., T. A. Lasko, Q. Mei, J. C. Denny, and H. Xu (2015). A study of active learning methods for named entity recognition in clinical text. *Journal of biomedical informatics* 58, 11–18.
- Chollet, F. et al. (2015). Keras. <https://keras.io>.
- Chomsky, N. (1957). *Syntactic structures*. Walter de Gruyter.

- Copenhagen Healthtech Cluster (2018). Bedre anvendelse af sundhedsdata . <https://www.cphhealthtech.dk/~media/CHC/Downloads/PDF/Data%20Redder%20Liv%20cases/Bedre-anvendelse-af-sundhedsdata.ashx>.
- Deng, L. (2018). Artificial intelligence in the rising wave of deep learning: The historical path and future outlook [perspectives]. *IEEE Signal Processing Magazine* 35(1), 180–177.
- Derczynski, L. (2016). Complementarity, f-score, and nlp evaluation. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pp. 261–266.
- Derczynski, L., E. Nichols, M. van Erp, and N. Limsopatham (2017). Results of the wnut2017 shared task on novel and emerging entity recognition. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pp. 140–147.
- Devlin, J. (2019). Bert/multilingual. <https://github.com/google-research/bert/blob/master/multilingual.md>.
- Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Ertekin, S., J. Huang, L. Bottou, and L. Giles (2007). Learning on the border: active learning in imbalanced data classification. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pp. 127–136.
- European Commission (n.d.). Hvilke personoplysninger anses for at være følsomme? https://ec.europa.eu/info/law/law-topic/data-protection/reform/rules-business-and-organisations/legal-grounds-processing-data/sensitive-data/what-personal-data-considered-sensitive_da.
- Friedman, C., G. Hripcsak, et al. (1999). Natural language processing and its future in medicine. *Acad Med* 74(8), 890–5.
- Graves, A. and J. Schmidhuber (2005). Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural networks* 18(5-6), 602–610.
- Hakala, K. and S. Pyysalo (2019). Biomedical named entity recognition with multilingual bert. In *Proceedings of The 5th Workshop on BioNLP Open Shared Tasks*, pp. 56–61.
- Hochreiter, S. and J. Schmidhuber (1997). Long short-term memory. *Neural computation* 9(8), 1735–1780.
- Hripcsak, G., C. A. Knirsch, N. L. Jain, R. C. Stazesky Jr, A. Pablos-Mendez, and T. Fulmer (1999). A health information network for managing innercity tuberculosis: bridging clinical care, public health, and home care. *Computers and Biomedical Research* 32(1), 67–76.
- Hugging Face (n.d.). Hugging Face transformers GitHub repository. <https://github.com/huggingface/transformers>.
- i2b2 (2006). i2b2 - informatics for integrating biology and the bedside. <https://www.i2b2.org/NLP/DataSets/Main.php>.
- Ive, J., N. Viani, J. Kam, L. Yin, S. Verma, S. Puntis, R. N. Cardinal, A. Roberts, R. Stewart, and S. Velupillai (2020). Generation and evaluation of artificial mental health records for natural language processing. *NPJ Digital Medicine* 3(1), 1–9.

- Kholghi, M., L. Sitbon, G. Zuccon, and A. Nguyen (2016). Active learning: a step towards automating medical concept extraction. *Journal of the American Medical Informatics Association* 23(2), 289–296.
- Kirkedal, A., B. Plank, L. Derczynski, and N. Schluter (2019). The lacunae of danish natural language processing. In *NEAL Proceedings of the 22nd Nordic Conference on Computational Linguistics (NoDaLiDa), September 30-October 2, Turku, Finland*, Number 167, pp. 356–362. Linköping University Electronic Press.
- Kripke, S. A. (1972). Naming and necessity. In *Semantics of natural language*, pp. 253–355. Springer.
- Kræftens Bekæmpelse (n.d.). Hvad er brystkræft? <https://www.cancer.dk/brystkraeft-mammacancer/hvad-er-brystkraeft/>.
- Lafferty, J., A. McCallum, and F. C. Pereira (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data.
- LeCun, Y., Y. Bengio, and G. Hinton (2015). Deep learning. *nature* 521(7553), 436–444.
- Lee, H.-J., H. Xu, J. Wang, Y. Zhang, S. Moon, J. Xu, and Y. Wu (2016). Uthealth at semeval-2016 task 12: an end-to-end system for temporal information extraction from clinical notes. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pp. 1292–1297.
- Lee, J., W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang (2020). Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics* 36(4), 1234–1240.
- Lewis, D. D. and W. A. Gale (1994). A sequential algorithm for training text classifiers. In *SIGIR'94*, pp. 3–12. Springer.
- Li, F.-F., J. Johnson, and S. Yeung (2017). Recurrent neural networks. http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf.
- Liddy, E. D. (2001). *Natural language processing*. In *Encyclopedia of Library and Information Science*, 2nd Ed. NY: Marcel Dekker, Inc.
- Manning, C. D., C. D. Manning, and H. Schütze (1999). *Foundations of statistical natural language processing*. MIT press.
- Melamud, O., J. Goldberger, and I. Dagan (2016). context2vec: Learning generic context embedding with bidirectional lstm. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pp. 51–61.
- Meystre, S. M., G. K. Savova, K. C. Kipper-Schuler, and J. F. Hurdle (2008). Extracting information from textual documents in the electronic health record: a review of recent research. *Yearbook of medical informatics* 17(01), 128–144.
- Mikolov, T., W.-t. Yih, and G. Zweig (2013). Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pp. 746–751.
- Mollerhoj, J. (2019). Botxo/nordic-bert. https://github.com/botxo/nordic_bert.
- Munro, R. (2020). *Human-In-The-Loop- Machine learning*. S.I: O'REILLY MEDIA.
- Névél, A., H. Dalianis, S. Velupillai, G. Savova, and P. Zweigenbaum (2018). Clinical natural language processing in languages other than english: opportunities and challenges. *Journal of biomedical semantics* 9(1), 12.

- Pantazos, K., S. Lauesen, and S. Lippert (2011). De-identifying an ehr database-anonymity, correctness and readability of the medical record.
- Pantazos, K., S. Lauesen, and S. Lippert (2017). Preserving medical correctness, readability and consistency in de-identified health records. *Health informatics journal* 23(4), 291–303.
- Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12, 2825–2830.
- Pennington, J., R. Socher, and C. D. Manning (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543.
- Peters, M. E., M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.
- Pilehvar, M. T. and J. Camacho-Collados (n.d.). *Embeddings in Natural Language Processing*. Morgan & Claypool Publishers.
- Pinola, M. (2011). Speech recognition through the decades: How we ended up with siri. *Web log post. TechHive. IDGTechNetwork* 2.
- Pradhan, S., N. Elhadad, W. Chapman, S. Manandhar, and G. Savova (2014, August). SemEval-2014 task 7: Analysis of clinical text. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, Dublin, Ireland, pp. 54–62. Association for Computational Linguistics.
- Pradhan, S., N. Elhadad, B. R. South, D. Martinez, L. M. Christensen, A. Vogel, H. Suominen, W. W. Chapman, and G. K. Savova (2013). Task 1: Share/clef ehealth evaluation lab 2013. In *CLEF (Working Notes)*.
- Pradhan, S., A. Moschitti, N. Xue, H. T. Ng, A. Björkelund, O. Uryupina, Y. Zhang, and Z. Zhong (2013). Towards robust linguistic analysis using ontonotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pp. 143–152.
- Pustejovsky, J., K. Lee, H. Bunt, and L. Romary (2010, May). ISO-TimeML: An international standard for semantic annotation. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).
- Qiao, Y., C. Xiong, Z. Liu, and Z. Liu (2019). Understanding the behaviors of bert in ranking. *arXiv preprint arXiv:1904.07531*.
- Rajapakse, T. (2020). Simple Transformers. <https://simpletransformers.ai/>.
- Rasmussen, M. and N. Berggrein (2019). Named entity recognition and disambiguation in danish electronic health records. Master's thesis.
- Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.
- Russell, S. J. and P. Norvig (2016). *Artificial intelligence: A Modern Approach*. Pearson Education Limited.
- Sak, H., A. W. Senior, and F. Beaufays (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling.

- Salton, G., A. Wong, and C.-S. Yang (1975). A vector space model for automatic indexing. *Communications of the ACM* 18(11), 613–620.
- Sang, E. F. and F. De Meulder (2003). Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*.
- Settles, B. (2009). Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences.
- Shao, J., Q. Wang, and F. Liu (2019). Learning to sample: an active learning framework. *arXiv preprint arXiv:1909.03585*.
- ShARe/CLEF (2013). Share/clef ehealth. <https://sites.google.com/site/shareclefehealth/>.
- Shen, Y., H. Yun, Z. C. Lipton, Y. Kronrod, and A. Anandkumar (2017). Deep active learning for named entity recognition. *arXiv preprint arXiv:1707.05928*.
- Sørensen, H. T., L. Pedersen, J. Jørgensen, and V. Ehrenstein (2016). Danish clinical quality databases—an important and untapped resource for clinical research. *Clinical epidemiology* 8, 425.
- Spasic, I. and G. Nenadic (2020). Clinical text data in machine learning: Systematic review. *JMIR Medical Informatics* 8(3), e17984.
- Styler IV, W. F., S. Bethard, S. Finan, M. Palmer, S. Pradhan, P. C. De Groen, B. Erickson, T. Miller, C. Lin, G. Savova, et al. (2014). Temporal annotation in the clinical domain. *Transactions of the Association for Computational Linguistics* 2, 143–154.
- Sun, W., A. Rumshisky, and O. Uzuner (2013). Evaluating temporal relations in clinical text: 2012 i2b2 challenge. *Journal of the American Medical Informatics Association* 20(5), 806–813.
- THYME (n.d.). THYME Project Wiki page. https://clear.colorado.edu/TemporalWiki/index.php/Main_Page.
- Tourille, J., O. Ferret, X. Tannier, and A. Névéol (2017). Limsi-cot at semeval-2017 task 12: Neural architecture for temporal information extraction from clinical narratives.
- Turing, A. (1950). Computing machinery and intelligence. *Mind* 59(236), 433.
- Uzuner, Ö., B. R. South, S. Shen, and S. L. DuVall (2011). 2010 i2b2/va challenge on concepts, assertions, and relations in clinical text. *Journal of the American Medical Informatics Association* 18(5), 552–556.
- Vapnik, V. N. (1999). An overview of statistical learning theory. *IEEE transactions on neural networks* 10(5), 988–999.
- Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin (2017). Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008.
- Velupillai, S., D. Mowery, B. R. South, M. Kvist, and H. Dalianis (2015). Recent advances in clinical natural language processing in support of semantic analysis. *Yearbook of medical informatics* 24(01), 183–193.
- Viani, N., T. A. Miller, C. Napolitano, S. G. Priori, G. K. Savova, R. Bellazzi, and L. Sacchi (2019). Supervised methods to extract clinical events from cardiology reports in italian. *Journal of biomedical informatics* 95, 103219.
- WHO (n.d.). International classification of diseases (icd) information sheet. <https://www.who.int/classifications/icd/factsheet/en/>.

- Xiao, H. (2018). bert-as-service. <https://github.com/hanxiao/bert-as-service>.
- Zhang, X., Y. Zhang, Q. Zhang, Y. Ren, T. Qiu, J. Ma, and Q. Sun (2019). Extracting comprehensive clinical information for breast cancer using deep learning methods. *International Journal of Medical Informatics* 132, 103985.