TU WIEN Informatics

Krankheit-Symptom Relationsextraktion aus medizinischen Texten mit BERT

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Data Science

eingereicht von

BSc Adrian Schiegl
Matrikelnummer 01306615

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Dr. Allan Hanbury
Mitwirkung: Univ.Ass. Dipl.-Ing. Markus Zlabinger

Wien, 20. Mai 2021

Adrian Schiegl	Allan Hanbury

TU WIEN Informatics

Disease-Symptom relation extraction from medical text corpora with BERT

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Data Science

by

BSc Adrian Schiegl
Registration Number 01306615

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dr. Allan Hanbury
Assistance: Univ.Ass. Dipl.-Ing. Markus Zlabinger

Vienna, 20th May, 2021

Adrian Schiegl	Allan Hanbury

Erklärung zur Verfassung der Arbeit

BSc Adrian Schiegl

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 20. Mai 2021

Adrian Schiegl

Acknowledgements

First and foremost, I have to thank my thesis supervisors Allan Hanbury and Markus Zlabinger. They offered me their assistance at every step of the way, while also giving me the freedom to steer the direction of this work. Even though the pandemic made collaboration more difficult, they always made it a priority to provide feedback swiftly.

I would also like to thank Lukas Seper and Tamás Petrovics from XUND. They never failed to show support and understanding, even at times when progress was slower than we all had hoped for.

Finally, I must express profound gratitude to my parents and my brother for supporting me throughout all the years of studying. I cannot imagine a scenario in which I would have been able to do this on my own. Thank you.

Abstract

To this day vast amounts of medical knowledge is still published in unstructured form e.g., case reports, clinical notes etc. The automated extraction of relations from unstructured sources between symptoms, diseases and other patient related information plays an important role in areas such as Evidence Based Medicine. For example, effective disease-symptom relation extraction accelerates tasks such as reviewing large amounts of medical literature to learn new disease characteristics.

In this work we present a relation extraction model based on BERT and MetaMap that extracts disease-symptom relations from over 20,000 BMJ Case Reports. Case reports are medical publications that contain clinically important information about the course of patients with specific medical conditions. Our model exploits the fact that a case report focuses on a single disease which is mentioned in the case report title. By doing so we represent the problem of relation extraction as a named entity recognition problem, which simplifies the model and the annotation of the training dataset.

We evaluate our model using the Disease Symptom Relation Collection (DSR). DSR is a set of graded disease-symptom relations from 20 diseases which was curated by medical doctors. We evaluate our model by measuring the relevance of the disease-symptom relations it extracted from BMJ Case Reports. We measure relevance by calculating the agreement with the ground truth provided by the medical doctors with the metrics nDCG@k, precision@k and recall@k. Furthermore, we compare the relevance our model achieved with the relevance of two baseline models: a word2vec model and a co-occurrence model trained on 1.5 million PubMed Central articles.

Our results show that our approach outperforms baselines by up to 25% nDCG, 27% precision and 10% recall. The agreement between our model and the ground truth is up to 64% nDCG@5 and 66% precision@5. Furthermore, our results also show that case reports are a high quality source of disease-symptom relations. Despite that, we find that they are of limited use due to the small number of openly accessible case reports.

Contents

CHAPTER 1

Introduction

Disease-Symptom relation extraction is the task of retrieving associations between diseases and symptoms from natural language text. The goal is to find relations in text sources and extract them using *Natural Language Processing* (NLP) models. A disease-symptom relation is a pair of disease and symptom concepts, representing a specific form of association e.g., causal relationship. Diseases and symptoms are often defined by concepts from openly available ontologies or vocabularies. The purpose of using ontologies is to be able to interface with the wider ecosystem of medical information processing tools and to clearly distinguish between concepts.

Disease-Symptom relations are just one of many biomedical relations. Other common biomedical relations are for example, drug-drug interaction (DDI) or drug-disease relations. DDI relations state interactions between drugs when they are taken in the same timeframe. Drug-Disease relations are statements about drugs and their relation to the cause of a disease.

1.1 Motivation

Effective extraction of relations between diseases, symptoms and other relevant medical information plays a crucial role in Evidence Based Medicine (EBM). EBM aims to integrate healthcare of patients with the best and current medical research evidence. To this day, vast amounts of medical evidence are published in unstructured form, such as case reports, clinical notes or scientific literature. Staying informed about the current research evidence becomes increasingly more difficult due to the rate at which new evidence is published. Reading hundreds of articles in order to assess the relationship between a disease and symptoms is very time-consuming. Disease-Symptom relation extraction helps speed up the search and interpretation of medical knowledge, thereby helping healthcare professionals to stay up-to-date. Search engines may be improved by annotating medical text with relations, therefore allowing users to search for evidence

of a relation between diseases and symptoms. Interpretation becomes easier because relations across many thousands of medical texts can be interpreted more quickly e.g., using aggregate statistics.

Another area which disease-symptom relation extraction models can positively impact is automated diagnostics. Automated diagnostic tools can serve as a first point of contact for patients in the healthcare system, thereby reducing the amount of resources needed for guiding patients to receive appropriate treatment. One such tool was developed by the healthcare startup XUND, who sponsored this thesis. The motivation is to enrich their medical knowledge base with biomedical relations, in order to further develop their automated diagnostic model.

1.2 Aim of the work

In recent years Deep Learning approaches provided major performance improvements in many natural language comprehension tasks. We aim to leverage the performance improvements of state-of-the-art NLP Deep Learning models for the problem of disease-symptom relation extraction. To be specific, we aim to evaluate a disease-symptom relation extraction model based on the Deep Learning model *Bidirectional Encoder Representations from Transformers* (BERT) [DCLT19]. We evaluate the model by measuring the relevance of the relations extracted from medical case reports, using the *Disease Symptom Relation Collection* (DSR). Case reports are medical publications that contain clinically important information about the course of patients with specific medical conditions. They describe information in free text, such as patient information (e.g., age, gender etc.), differential diagnoses and symptoms the patients reported. The source of the case reports is BMJ Case Reports[1]. DSR [ZHRH20] is a collection of disease-symptom relations created by medical doctors, and it serves as a ground truth for our evaluation. We measure the relevance of the relations our model extracted by computing the nDCG, precision and recall with respect to DSR. Lastly, we compare the relevance that our model achieves with baselines provided using the DSR.

Due to special requirements from our sponsor, our model must also be designed to be extensible to more complex relations. Thus, in addition to symptoms and diseases, the model must be flexible enough such that it can be adapted to extract other patient information, such as age and gender, as well.

The last goal of this work is to provide a review of biomedical extraction methods and background necessary to understand our relation extraction model. The literature on disease-symptom relation extraction methods is comparatively small, therefore we widened our scope to biomedical relation extraction methods. However, all methods reviewed can either be applied directly to disease-symptom relation extraction or adapted straightforwardly with the background we provide.

Our work answers the following questions:

[1] BMJ Case Reports — `https://casereports.bmj.com`

1. How relevant are disease-symptom relations extracted from case reports by our model compared to baselines?
2. What are the advantages and disadvantages of our model for disease-symptom relation extraction?
3. What are the advantages and disadvantages of case reports for disease-symptom relation extraction?

1.3 Structure of the work

Chapter 2 reviews biomedical extraction methods and background necessary to understand our model. We focus on neural network based methods because we use techniques and concepts from these methods for our model.

Chapter 3 describes the dataset we use to extract disease-symptom relations from. We present the source and discuss the structure of the data. Additionally, we describe the annotation process used to create the training dataset for our model.

Chapter 4 describes our relation extraction model. We describe the architecture of the model, starting with the vocabulary used to represent medical concepts. Then we dissect and explain the modules comprising our relation extraction model.

Chapter 5 describes the experiments we conduct to answer our first research question, whether our approach outperforms baseline models. We also present the results of the experiments and analyze them using hypothesis tests and visualizations.

Chapter 6 presents the conclusions we draw from the experiments. We discuss the advantages and disadvantages of our approach, as well as case reports as a source for relation extraction.

Chapter A contains information relevant to the thesis such as software versions used in the implementation of the experiment.

CHAPTER 2

Extraction methods

This chapter discusses biomedical relation extraction methods based on neural networks. The purpose of this chapter is not to provide a complete and exhaustive literature review. Instead, we aim to provide the reader with enough background to understand the approach we have taken in this work. Alongside the background, we present related work to understand how our approach compares to other biomedical relation extraction approaches.

We widened our scope from disease-symptom relations to biomedical relations in general, because the literature on disease-symptom relation extraction is comparatively small. And methods devised for other biomedical relations can either be applied directly to disease-symptom relations or straightforwardly adapted.

We only choose methods based on neural networks because our method is also based on neural networks. Furthermore, instead of simply listing methods, we generalize them into a framework/approach to highlight their modular nature. We do so because we find that many methods reuse and combine parts of other methods. Therefore, integrating the methods into a framework/approach gives a clearer picture on how these methods are composed and enable the reader to adapt methods to disease-symptom relation extraction if necessary. The Overview 2.1 describes the general approach of neural network based methods.

2.1 Overview

In the past decade interest in neural networks has sparked again in the computer science community. The resurgence of neural networks may be partly attributed to the large increase in computational power. Despite their versatility, their computationally complex design led to limited use prior to that. Another contributing factor is the massive increase in availability of training data, due to but not exclusively, the growth of the internet.

The new abundance of data allowed for increasingly larger network architectures e.g., transformers to be trained. The field of NLP has been particularly uplifted by recent advances in neural networks architecture, hardware and data availability.

Neural network based methods are designed in a modular way, thus allowing for many possible combinations. Discussing all the conceivable combinations of architectures, preprocessing steps and training procedures will be infeasible in this work. However, given the fairly narrow context of biomedical relation extraction we found most neural network based methods to follow similar design patterns. We have distilled them into an approach which is shown in Figure 2.1.

We dedicate a section in this chapter for each step in Figure 2.1. The first section, Dataset Annotation 2.2 describes what datasets are, how they can be annotated and evaluated. Preprocessing 2.3 discusses common preprocessing steps used on the unstructured text. Embedding 2.4 describes how unstructured text, as well as the additional information generated in the preprocessing step, can be represented as vectors. The last section, Classification 2.5 describes how neural networks are used to extract relations through means of classification.

2.2 Dataset Annotation

Most approaches that incorporate neural networks in biomedical relation extraction are based on some form of supervised learning. The goal of supervised learning is to fit a machine learning model that maps between input and output pairs as accurately as possible. The input in our case is usually natural language text. However, the input may be enriched with linguistic information about the text e.g. POS-tags, or any other type of information that can be derived from the input at prediction time. We present common types of additional information in Section 2.3. Outputs tend to be relations themselves or some intermediate output needed for subsequent models continuing the relation extraction. The task of mapping between input text and output relations is a classification task.

Creating your own dataset for biomedical relation extraction models based on neural networks is costly. Neural networks need large amounts of labeled data for training and evaluation. It can be very time-consuming to collect large amounts of data with decent annotation quality. Depending on the relation, annotating biomedical text may require domain specific knowledge, thus reducing the amount of potential annotators. The problem of collecting large amounts of data is further aggravated by the fact that data from the biomedical domain is often less likely to be shared due to privacy concerns. Therefore, even if access to qualified annotators is not an issue, getting large amounts of data for biomedical relation extraction can be.

Most of the research we present focuses solely on the extraction method. They evaluate their models on benchmark datasets that have already been labeled by someone else. However, when different type of relations need to be extracted or new instances of

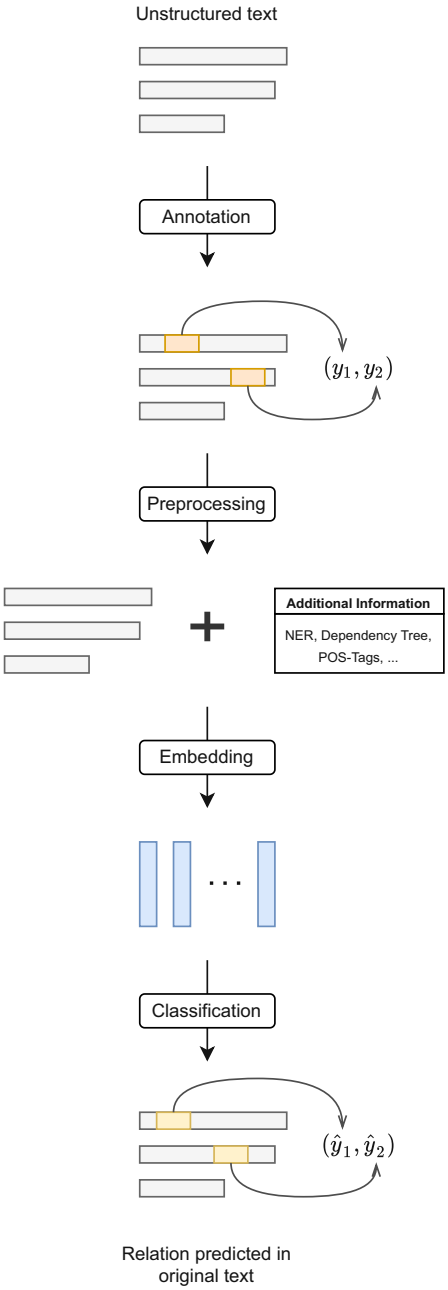

Figure 2.1: Most common steps of neural network based biomedical relation extraction pipeline. (y_1, y_2) is a relation between the biomedical entities y_1 and y_2. $(\hat{y}_1, \hat{y}_2)$ is the relation predicted by the model.

existing relations appear e.g., new drugs or diseases, datasets may need to be labeled or re-labeled. The remainder of this section discusses details that need to be considered when annotating a dataset for biomedical relation extraction.

2.2.1 Annotation level

The annotation level refers to the scope of the annotation. For example, one can create annotations per sentence, paragraph or document. Most of the methods reviewed in this work predict relation labels per sentence, thus requiring sentence level annotation. Annotation at a *sentence level* trades off relation coverage and accuracy for speed of annotation. On one hand, the speed at which annotations are generated increases because annotators do not have to read entire paragraphs or even documents before assigning annotations. On the other hand, the removal of context from the sentences may introduce ambiguities or even make inferring relations impossible altogether. Thus, some relations will be missed that were potentially discoverable given the more complete text.

As an example, we present a sentence from a medical case report[1].

> The condition improved markedly after a few days' treatment with non-steroid anti-inflammatory drugs and prednisolone.

This sentence implies the existence of a causal relation between a drug treatment and the recovery of a patient. An annotator with the goal of labeling drug-disease relations would have to skip this sentence if it were seen in isolation. But given some context, in this case the preceding sentence, the annotator would have been able to discover the drug-disease relation i.e., the relation between reactive arthritis and non-steroid anti-inflammatory drugs and prednisolone.

> Thus, the condition was compatible with reactive arthritis. The condition improved markedly after a few days' treatment with non-steroid anti-inflammatory drugs and prednisolone.

The most appropriate annotation level depends on the type of text, resources available for annotation and the neural network architecture used. If the documents are long form texts where single statements can easily span more than one sentence, then a paragraph or document level approach may be more appropriate. Another situation in which to prefer paragraph or document level annotations is when more resources can be dedicated to the dataset annotation i.e. more people or annotators with more extensive domain knowledge. Doing so will make the labeling more accurate and complete. Lastly, none of the above matters if the neural network architecture is not equipped to deal with long sequences of text. The task of learning long range dependencies between parts of texts e.g., relating words that are far apart, is still a big challenge for NLP models. Section 2.5 discusses this challenge and how it is handled by various neural network architectures.

[1] Reactive arthritis after COVID-19 — http://dx.doi.org/10.1136/bcr-2020-241375

2.2.2 Manual labeling

The most straightforward way to build a dataset suitable for supervised learning of biomedical relations is to label relations manually. The manual approach has been successfully used many times before. This section reviews the approaches taken for two biomedical relation datasets in more detail.

The *BioCreative V CDR task corpus* [LSJ+16] was created for the purpose of advancing chemical-disease relation research. The team behind the work manually annotated 1500 biomedical articles with chemical entities, disease entities and chemical-disease relations. The labels corresponded to Medical Subject Headings (MeSH). To speed up the process of annotation they used software specialized for biomedical data curation [WKL13] (Figure 2.2). PubTator speeds up the annotation process by allowing the user to annotate using a graphical user interface, instead to editing large text files manually. The corpus' annotations are built on top of an already annotated corpus from Comparative Toxicogenomics Database (CTD) and Pfizer. As a consequence some of their work consisted of modifying or deleting existing annotations. Instead of annotating sentences they chose to annotate entire abstracts. The annotation process was undertaken by 4 people, each with a medical training background and curation experience. Each sample was annotated by two individuals. In the case of conflicting annotations, the other two people were tasked to resolve the conflict. The inter-annotator agreement was above 85% for all splits of the dataset.

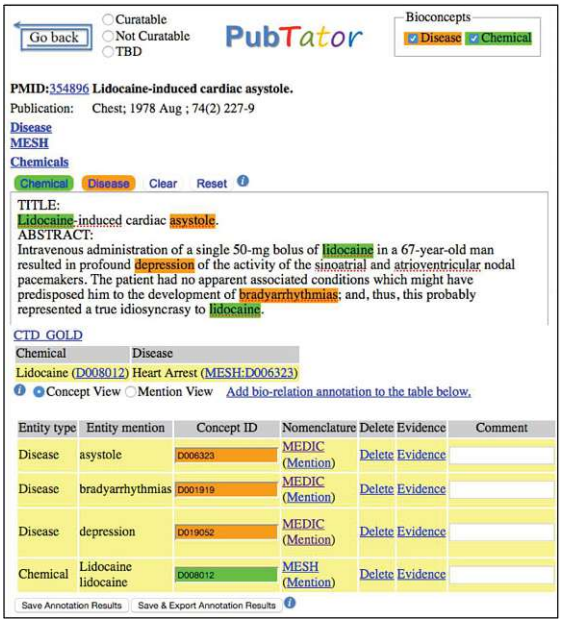

Figure 2.2: Screenshot of an annotation example in PubTator [WKL13] taken from [LSJ+16].

The *Drug-Drug-Interaction (DDI) corpus* [HSMD13] was created for the purpose of

furthering research about interactions between drugs. The corpus consists of 1025 randomly selected documents from the DDI-Drugbank database and MedLine abstracts. The annotations were performed at the sentence level, resulting in 8942 sentences. They used the tool MetaMap [Aro01] to pre-annotate drugs in the text using MeSH concepts. Before the actual annotation they defined strict guidelines for the annotation process. Focus was laid on the problem that drugs often have multiple names e.g., name brand, chemical substance name etc. The annotations were then performed by two pharmacologists with substantial background in pharmacovigilance. They were assisted by a text miner in all technical aspects. They did not make use of any graphical annotation tools, instead they curated annotations using an XML Editor[2]. The inter-annotator agreement was 80% and above for all parts of the dataset.

To finish this section, we highlight common challenges encountered by the works discussed above. This approach relies on human annotators familiar enough with the domain to inspect and label each sample individually. It is not uncommon for neural networks to be trained on thousands of samples. Assuming that labeling a sentence relation pair takes on average 30 seconds, annotating 5000 sentences will take more than 41 hours. When taking into account that each sentence should be annotated by more than one person, the number of hours increases even more. Employing experts for this many hours may become infeasible. Another challenge encountered by the works discussed above is ambiguity or the existence of multiple plausible annotations. The authors of the DDI-corpus tackled this problem by developing strict guidelines for the annotation process. Increasing the number of annotators for each sample is also way to reduce uncertainty, albeit an expensive one.

2.2.3 Distant Supervision

Traditional supervised relation extraction suffers from the problem that labeled datasets are time-consuming to create. This problem is aggravated when the domain of the text is highly technical and can therefore only be reliably labeled by experts in the field. *Distant supervision* aims to circumvent this problem by programmatically generating input output pairs for the machine learning model. Distant supervision has already been used in relation extraction before [MBSJ09]. In this work the authors used relations found in the semantic database Freebase to label large text corpora such as Wikipedia. The assumption of their method is that if two entities form a relation in the semantic database they are likely to exhibit that relationship as well when both entities can be found in the same sentence. They were able to extract 10,000 instances of 102 relations with distantly labeled dataset at a precision of 67.6%. But the method suffers from the strength of its assumption. For example the following two sentences contain both of the two entities *headache* and *concussion*.

1. A patient suffered from chronic *headaches* as a result of a *concussion*.

[2]XML Notepad `https://microsoft.github.io/XmlNotepad/`

2. The patient reports *headaches* but showed no sign of a *concussion*.

If the relation we are interested in is a causal relation then the method will fail. The first sentence states a causal relation between headache and concussion, but not the second one. The method will potentially introduce a high number of false positive instances. One way to reduce the number of false positives is to make use of syntactical or lexical features [BZKA20]. The methods described by the work are based on heuristics. They are very similar to the methods described in the section about linguistics- and rule based methods. The problem of false positive labels generated by this type of method can also be tackled by during training of the machine learning model. For example, Tran and Kavuluru [TK19] implemented a loss function that is more resistant to label noise.

2.2.4 Annotation Quality

Once the dataset is annotated, it is desirable to measure the quality of the annotation. One indicator of quality is whether the dataset was manually annotated by experts. Datasets annotated this way are considered *gold standard*. Experts are people with considerable experience in the domain e.g., a medical doctor.

A practical way to increase annotation quality is to increase the number of annotators for each sample. Having multiple annotations per sample allows us to pick the final annotation by majority voting, thus reducing noise of the labels. Another advantage of having multiple annotations per sample is that we can quantify the quality of the labels. The *inter-annotator agreement* quantifies the agreement of labels from multiple annotators. For this to be a measure of annotation quality we need to assume that when more people agree on a label, the label is more likely to be correct.

A popular way to assess the inter-annotator agreement is *Fleiss' Kappa*. Fleiss' Kappa [Fle71] is a statistical measure to compute the agreement of more than two annotators. The measure requires the number of annotators to be fixed and annotations to be on nominal scale.

2.3 Preprocessing

This section describes the most common preprocessing methods used by biomedical relation extraction methods based on neural networks.

2.3.1 Dependency Parsing

Dependency parsing is a common task in NLP pipelines. The goal of dependency parsing is to enrich plain text with its grammatical structure. The result of dependency parsing is a graph structure called dependency parse tree (Figure 2.3).

Definition 2.3.1 (Dependency Parse Tree)**.** A dependency parse tree is a directed graph $G = (V, E)$ where words are represented by vertices V and grammatical relations between

words by edges E. Punctuation marks may be part of V. All vertices apart from the root have incoming edges.

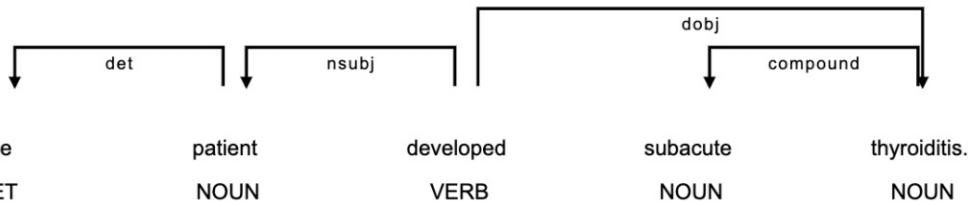

Figure 2.3: The sentence "The patient developed subacute thyroiditis." transformed into a dependency tree by spaCy with `en_core_web_sm` pipeline. The root of the tree is the word "developed".

Parsing of natural languages proves to be difficult due to ambiguous or complex rules. State-of-the-art models for dependency parsing are therefore deep neural networks instead of handcrafted rules. The current state-of-the-art model for dependency parsing is based on transformers and recurrent neural networks, as is the case with many natural language tasks. The model based on XLNet [MDT+20] achieved 96.26 LAS[3] on the Penn TreeBank benchmark dataset [MSM93]. Biomedical dependency parsing benchmarks show a similar picture. The state-of-the-art for the biomedical benchmark corpus GENIA [KOTT03] was achieved by Bi-LSTM CRF [NV19] and SciBERT [BLC19].

2.3.2 Part-Of-Speech Tagging (POS)

Part-of-speech tagging or POS-tagging is the process of assigning grammatical descriptors to words. The choice of tags is dependent on the context the word e.g., *work* may be a noun or a verb depending on its usage. Since natural languages differ in their complexity and usage, the sets of tags may also differ. However, a common set of tags may still be useful (Table 2.1).

As with dependency parsing, the state-of-the-art in POS-tagging is also dominated by deep neural networks ([BMS+18], [CZ19]).

2.3.3 Named-entity Recognition (NER)

Named-entity recognition (NER) is the task of classifying entities in unstructured or natural language text. The goal is to assign words or entire parts of the text into a known set of classes. In the case of biomedical relation extraction, the classes will likely include the entities in question e.g., symptom-disease relation extraction will include the classes *symptom* and *disease*.

[3] *Labeled Attachment Score* (LAS) is the ratio between number of correctly predicted dependencies/relations and the number of possibilities

Symbol	Description
ADJ	adjective
ADP	adposition
ADV	adverb
AUX	auxiliary verb
CONJ	coordinating conjunction
DET	determiner
INTJ	interjection
NOUN	noun
NUM	numeral
PART	particle
PRON	pronoun
PROPN	proper noun
PUNCT	punctuation
SCONJ	subordinating conjunction
SYM	symbol
VERB	verb
X	other

Table 2.1: List of universal POS-tags from Python NLP library spaCy.

As with most language understanding tasks, regardless of domain, the current state-of-the-art for NER tasks is also deep neural networks. Nooralahzadeh et al. achieved 89.93 F1-score [NLØ19] in NER on the chemical-disease relation extraction dataset BC5CDR [LSJ+16]. Their approach was based on distant supervision, described in Section 2.2.3, to tackle the problem of scarcity of high quality labeled datasets in the medical domain, as well as neural reinforcement learning for denoising. Lee et al. trained the transformer based model BERT on biomedical text corpora and achieved state-of-the-art with 89.72 F1-score [LYK+19] on the biomedical NER benchmark NCBI Disease [DLL14].

2.4 Embedding

Neural networks cannot use the characters directly in the text as input. Neural networks only accept numerical input in the form of vectors, matrices or more generally tensors. We must therefore convert sequences of characters into numerical form.

Definition 2.4.1 (Embedding)**.** The process and the result of transforming text into a numerical structure is called *embedding*. Embeddings are also commonly referred to as vector representations. An embedding function f transforms a sequence of characters $s \in S$ and (learnable) model parameters $\theta \in \Theta$ into a vector in $\mathbb{R}^n$, where n is the dimensionality of the embedding.

$$f : (S, \Theta) \to \mathbb{R}^n$$

There are a plethora of ways to embed text into vector spaces, with new ones being developed regularly. This means we will only be able to cover popular methods. We will group them by the characteristics they exhibit and discuss how newer methods improved upon older ones.

2.4.1 Context independent embedding

The defining feature of this class of embeddings is that tokens, usually words, will be transformed without taking into account their context. For example, the word *tree* will have the exact same vector representation whether it is used in a computer science textbook or in a biology text book.

One-hot encoding is one of the simplest methods to embed text. Even though this method is now rarely used in NLP — why will become apparent after this paragraph — it helps to appreciate the details of more complex methods being discussed later. In one-hot encoding each dimension in the vector representation corresponds to a distinct word in the text corpus. The representation of a word is a sparse vector of mostly zeros and a single one in the dimension that represents the word. The advantage of this method is that embeddings can be generated efficiently. However, as the text corpus grows, the number of distinct words will grow as well, causing the dimensionality of the vector embedding to increase linearly with the number of distinct words. The result will be a vector space with possibly thousands of dimensions, and the model will likely suffer from the curse of dimensionality [VF05]. Another disadvantage is that each embedding is orthogonal and of equal magnitude. This is, each word is equally similar to all other words. Machine learning models relying on distance will therefore not be able to exploit similarities between words in their modeling.

Word2Vec [MSC+] does not suffer from the two problems discussed before. The dimensionality of the word embeddings can be predefined and is independent of the distinct number of words in the corpus. The second problem is overcome by learning how to position semantically similar words closer together in the vector space. In order to achieve this Word2Vec exploits the fact that semantically similar words tend to be used in similar contexts. For example, due to the semantic similarity of the words *dog* and *cat*, both words tend to be used in similar sentences e.g., “I fed my cat.” and “He fed the dog.” Word2Vec fits the embedding function by learning to either predict surrounding words from the current word (skip-gram) or the current word given surrounding words (continuous bag of words). Similarity of the embeddings may then be quantified with geometric functions such as cosine similarity or Euclidean distance. The embedding function itself is a shallow two-layer neural network. While neural networks often require specialized hardware such as GPUs, Word2Vec is able to learn efficiently on the CPU. Later on, models such as *GloVe* and *FastText* were developed to improve on the approach that Word2Vec took. One shortcoming of Word2Vec is that previously unseen words cannot be embedded.

FastText tries to solve this problem by learning vector representations of n-grams or subwords [BGJM17] instead of full words. GloVe aims to improve on similarity tasks by leveraging statistical information about the global co-occurrence between words [PSM14]. This is in contrast to the local approach that Word2Vec uses.

2.4.2 Context dependent embedding

While Word2Vec and similar models led to better performance for many NLP tasks they were held back by the fact that each word had a single vector representation. The problem is that words often have multiple meanings and can be reused in different situations. For example, the word *cell* clearly has multiple meanings. It can refer to biological cells, prison cells or cell phones. In this case, Word2Vec must find a single vector representation that can be used all these scenarios. *ELMo* [PNI+18] and *BERT* [DCLT19] solve this problem by taking word order into account and generating word embeddings from not only the current word but also its surrounding words. This type of embedding is referred to as contextualized word embedding. In contrast, the family of context independent embedding models, which Word2Vec, GloVe and FastText are a part of, only takes surrounding words into account when training. ELMo achieves contextualization with deep bi-directional LSTM models (Figure 2.4). BERT does so through the help of deep bi-directional transformers. Both models require specialized hardware such as GPUs or TPUs in order to train efficiently on large corpora such as Wikipedia. Even though ELMo and BERT achieved state-of-the-art performance in many NLP tasks, they might not always be preferable to context independent embeddings. Context independent embeddings make sharing of embeddings easier as they are just vectors. There is no need to share the trained model (usually large and computationally complex) since embeddings do not need to be contextualized.

2.4.3 Subword and character-level embeddings

A common way to transform text into vector representations is to embed each word into a separate vector. This is the way Word2Vec or GloVe work. Word embedding algorithms consider words to be atomic units. Both of these models are still used today and provide acceptable performance for many NLP tasks. However, one major drawback of these methods is that they are not able to handle out of vocabulary words (OOV). These are words that are not part of the model vocabulary. Reasons that words are not included in a vocabulary are e.g., they were not encountered during training, or we imposed a limit on the size of the vocabulary. The consequence is that OOV words must be handled separately e.g., by removing them from the text, since the model does not handle them. Another less obvious shortcoming of word-level embedding models is that they are not able to take advantage of the internal structure of words. Words found in natural languages such as English or German are rarely understood as atomic tokens. Many words share parts of themselves with other words. This is often an indicator that the words are related in some way. This is especially the case in scientific domains where new words are needed regularly e.g., for a new process, new concept etc. A commonly

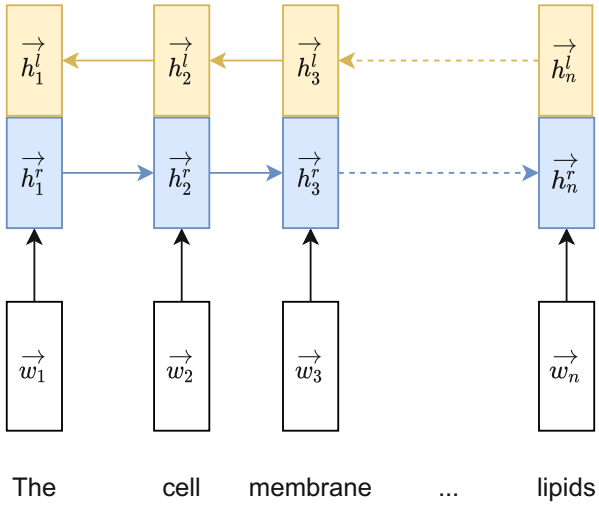

Figure 2.4: This diagram shows how a Bi-LSTM contextualizes the word *cell*. The initial word embedding for *cell* $\vec{w_2}$ will be endowed with information from words to the left and right of it. The result are two vectors $\vec{h_2^r}$ and $\vec{h_2^l}$ which will be concatenated. The information propagation and aggregation happens sequentially due to the design of Bi-LSTMs. The new embedding $\vec{h_2}$ will contain information from *membrane* and *lipids*, thus moving the meaning of *cell* into the biological domain.

used way to generate new words is to compound already existing words e.g., *Coronavirus* from the words *corona* and *virus*.

FastText has shown that enriching text embeddings with subword information improves performance [BGJM17]. FastText has also been successfully used in the biomedical domain where it outperformed word-level embeddings such as Word2Vec on a variety of biomedical information extraction tasks [ZCY+19]. To overcome the shortcomings mentioned in this paragraph, newer models such as ELMo [PNI+18] and BERT [DCLT19] also operate on the subword-level. All three of these models achieved state-of-the-art performance in many NLP tasks at the time of their release.

The smallest possible subwords are the characters themselves. Character-level embeddings also have shown to improve performance over word-level embeddings. Akbik et. al [ABV19] achieved state-of-the-art performance in a range of sequence labeling tasks such as CoNLL-03.

2.4.4 Embedding of linguistic information

A great strength of neural networks is their ability to process multiple types of data at once. We can take advantage of this by not only embedding the text but also additional linguistic information. This section reviews a few additional sources of information successfully used in biomedical relation extraction.

Shortest Dependency Paths (SDP) were found to improve performance for relation classification tasks when combined with word embeddings [LWL+15]. A SDP is the shortest path between two entities in a dependency graph. Dependency graphs are usually constructed with a dependency tree parser model, separate from the neural network relation classification model. The two entities connected by the SDP are defined by the relation we want to extract e.g., symptom and disease. While it would be technically possible to add other SDPs to the input, it is not obvious which ones the model would benefit from more than the two entities that are part of the relation. The SDP can be represented in vector form in various ways. A simple way to represent the SDP is to embed each word of the SDP with the same text embedding model that is used for the full sentence. Another way to embed an SDP is to embed the edges of the SDP graph [ZLY+18] e.g., *nsubj*, *prep* etc.

POS-tagging information can be added in a straightforward manner through concatenation with the word vectors ([ZZL+18], [LZFJ17]). This is because each word in the input sentence has a one-to-one correspondence with a POS-tag. Note that this way of including POS-tags makes the use of subword tokens more difficult, as POS-tags only exist for words. We can include POS-tagging information independently like we did with SDPs.

Figure 2.5 shows a general framework of how to combine syntactic and lexical information with word embeddings in a neural relation classification model.

2.4.5 Pretrained embeddings

In order to gain more natural language understanding, NLP models often use pretrained text embeddings generated during training of neural network models on very large text corpora. A popular text corpus to generate text embeddings from is Wikipedia. At the time of writing English Wikipedia contains more than 3000 million words or 6 million articles[4]. Generating text embeddings from large text corpora such as this is very time-consuming and computationally expensive. For this reason it is common to generate the embeddings once and then share them with the NLP community. Widely used libraries such as spaCy[5] and huggingface transformers[6] provide embeddings trained on various text corpora.

While using text embeddings created from large general purpose text corpora like Google News can be beneficial, it is preferred that text embeddings stem from a domain that is closely related with the domain the final model will be used in. This makes sense intuitively since the language and words used in any given text can differ significantly among domains. For example, Wang et al. [WLA+18] found that text embeddings from biomedical corpora capture the semantics of medical terms better than from general language corpora such as Google News. Text embeddings from the biomedical domain performed better in a variety of information retrieval tasks than ones from general purpose

[4] https://en.wikipedia.org/wiki/Wikipedia:Size_in_volumes
[5] https://spacy.io
[6] https://huggingface.co

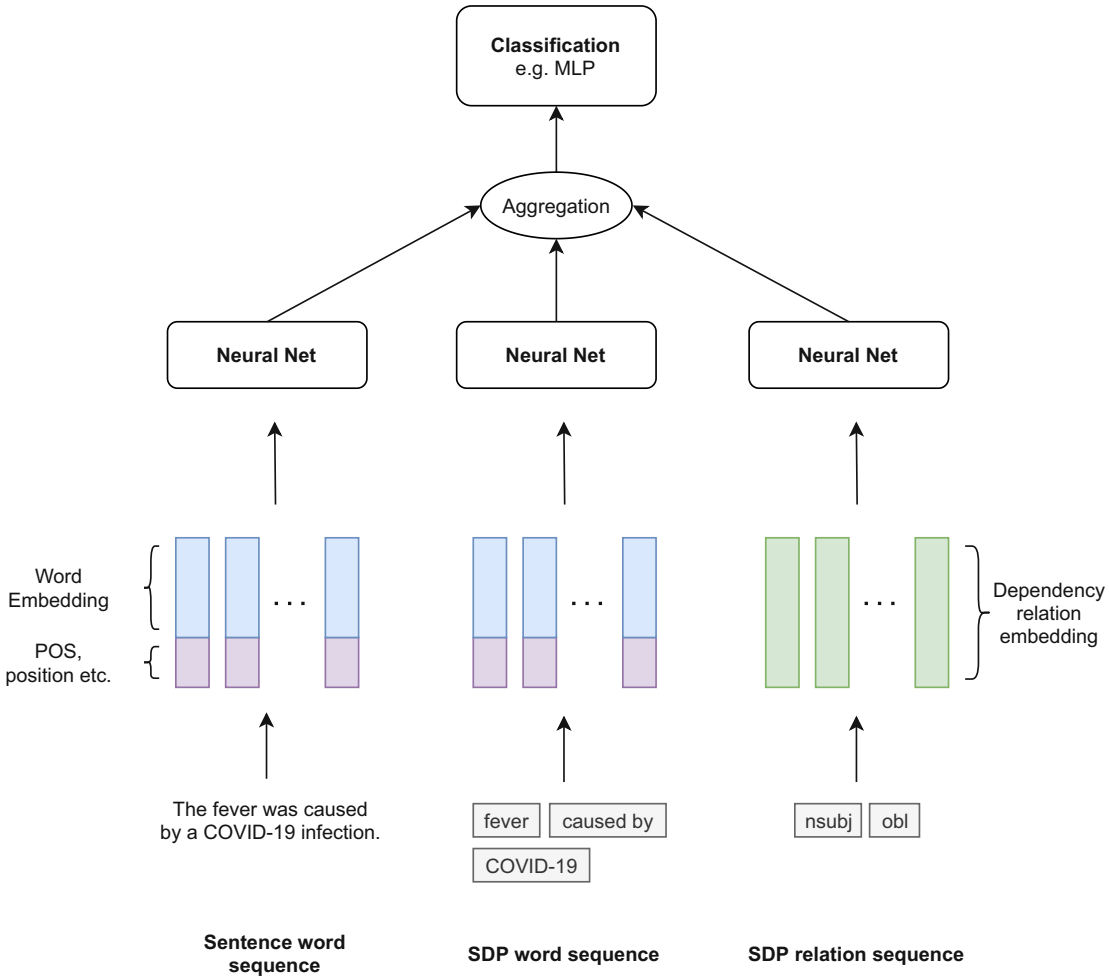

Figure 2.5: This graphic shows how to combine additional linguistic information with text embeddings such as SDP, POS-tags etc. The architecture is based on work done by [ZLY+18], [ZZL+18] and [LZFJ17]. The processing steps are laid out very generally e.g., *Neural Net* and *Aggregation* instead of *Bi-LSTM* and *Max-Pooling*. The reason is that there are many viable options to choose from.

domains. For this reason, widely used models such as Word2Vec or FastText often offer text embeddings for various domains like biology, medicine or law.

The type of pretrained embeddings discussed so far are context independent or global. Every word or subword will be transformed into the same vector, regardless of where in the text it happens to be. Consequently, embeddings can be assigned to each word or subword directly. No model is necessary to embed the text. But, as mentioned in Section 2.4.2, contextualized or context *dependent* embeddings are preferred due to their superior performance for many NLP tasks. These types of embeddings must be generated by the machine learning model that was pretrained. This is because every word will be assigned multiple different vector representations throughout the text, each assignment dependent on its surrounding words. The most popular models to produce contextualized text embeddings are based on deep neural networks.

2.5 Classification

In this section we review major concepts and ideas of neural network models used to extract biomedical relations. We describe what they are, why they are important and how they are used in biomedical relation extraction. By doing so we acquire the background needed to understand the model we present in Chapter 4 and extend it to more complex relations. Extensibility is a side objective we introduced in Section 1.2.

The models we describe in this section all follow the same steps:

1. Generate candidate pairs
2. Encode candidate pairs
3. Classify candidate pairs to extract relation

Relations between biomedical entities are not extracted in the literal sense, instead they are extracted implicitly through classification. Given a piece of text, a neural network based relation extraction model will output the relation type present in the text e.g., disease-symptom or no disease-symptom. However, just detecting whether a relation is present in the text does not qualify as extraction. Therefore, the model also generates relation candidates or *candidate pairs*. A candidate pair is a potential relation present in the text. To illustrate what candidate pairs are, we find candidate pairs in the following sentence:

> The patient reported chest pain and nausea before being diagnosed with a heart attack.

This sentence has two candidate pairs: (heart attack, chest pain) and (heart attack, nausea). We can find these candidate pairs using NER models. The next step is to encode

candidate pairs. There are multiple approaches to this problem. For example, after detecting entities with the NER model, we can substitute diseases and symptoms in the text with dummy names e.g., replace "heart attack" with "DISEASE". By encoding the candidate pair in the sentence we are no longer classifying the sentence but the candidate pair. This is because during training the model learns to map sentences containing our dummy names to relation classes. Another way to encode the candidate pair is to extract the SDP of the pair and include it in the input.

2.5.1 Recurrent Neural Network

Recurrent Neural Networks (RNN) are a class of neural networks designed to process variable length sequences using an internal state. RNNs are a natural choice for biomedical relation extraction because text is a variable sequence of characters and words. The most common variants of RNNs in biomedical relation extraction are *Long Short-Term Memory* (LSTM) [HS97] and *Gated Recurrent Unit* (GRU) [DS17].

LSTMs and GRUs were developed to alleviate the *vanishing gradient problem* [Hoc98], which previous types of RNNs were known to suffer from. The problem describes the phenomenon of gradients becoming vanishingly small during gradient based training of very deep neural networks, and therefore preventing the update of network weights. RNNs are more prone to this problem because the effective depth[7] of the network depends on the length of the input sequence. Therefore, long sequences e.g., text paragraphs, result in very deep networks. The vanishing gradient problem prevents RNNs from learning long range dependencies in the sequence i.e., relating words further apart in the text. LSTMs and GRUs are capable of learning longer range dependencies and achieve higher performance on sequence modeling tasks compared to standard RNNs [CGCB14].

Further improvement of LSTMs and GRUs is achieved by introducing bidirectional processing. To signal that sequences are processed in both direction the prefix "bi" is used e.g., Bi-LSTM. Normally, the model transforms each element (e.g. word or subword) from the sequence into a vector by processing the sequence from left-to-right or right-to-left. This way, previous elements in the sequence affect the output vector of future elements. On the other hand, bidirectional RNNs process the sequence in both directions and output two vectors for each element. In order to obtain the same number of vectors as before, both vectors are aggregated e.g., concatenation. We already discussed the motivation behind this approach in Section 2.4.2. To summarize, processing in both directions allows us to take into account context from the entire text instead of just one side. The remainder of this section discusses literature that utilized Bi-LSTMs or Bi-GRUs for biomedical relation extraction.

Galvan et. al [GOMI18] uses Bi-LSTMs to extract temporal relations from clinical text in an end-to-end fashion. As a consequence, a single model captures all steps after annotation in our framework (Figure 2.1). They adapted the LSTM model from Miwa et. al [MB16] in order to evaluate it on 2016 Clinical TempEval. The model consists of

[7] Depth of the directed acyclic graph representing the RNN after it is unrolled.

two Bi-LSTMs stacked on top of each other which we call M_1 and M_2. M_1 generates candidate pairs (i.e. candidate relations) by classifying each word in the input text. If two words from the input text are classified with the entities we are interested in e.g., disease and symptom, they are considered a candidate pair. M_2 classifies each candidate pair among possible relation types e.g., disease-symptom relation or no relation. However, it does not directly classify the candidate pairs from M_1. Instead, an SDP between the pair elements is generated, embedded and classified. For example, let us consider the candidate pair $(word_x, word_y)$. We generate an SDP between $word_x$ and $word_y$ with elements $word_i \in SDP(word_x, word_y)$. Then we represent each element $word_i$ in the SDP as a concatenation of the M_1 output of $word_i$, M_1 hidden state of $word_i$, and a dependency graph arc embedding going out of $word_i$.

2.5.2 Hybrid models

Hybrid models are models which use more than one type of neural network. Li et al. [LZFJ17] uses a *Convolutional Neural Network* (CNN) in combination with a Bi-LSTM for various biomedical relation extraction tasks. The purpose of the CNN is to generate word embeddings from character-level embeddings. The advantage of character-level processing is that the CNN can extract morphological information from characters of the words. As a motivational example they provided the prevalence of common prefixes and suffixes in the biomedical domain. For example, the suffix "bacter" is a strong indicator that the word concerns itself with bacteria e.g., "campylobacter", "helicobacter". If we were to treat words as atoms instead of sequences of characters, we would not be able to access morphological information.

Zhang et al. [ZLY+18] also uses CNNs in combination with Bi-LSTMs. The model they propose is an ensemble of three models: Bi-LSTM which processes the full text, CNN which processes the SDP of the full text and a CNN which processes the dependency relations from the SDP. The outputs of the three neural networks are then reduced in size with Max-Pooling operations, before they are aggregated and classified in the final MLP[8] layer of the network. Zhang et al. motivates the usage of CNNs over RNNs because they are better at processing short sequences e.g., SDP.

2.5.3 Attention

Attention is a method to encode elements of a sequence which resolves a serious drawback of RNN-based models. RNNs process elements sequentially and need to "remember" information over many steps to form long range dependencies between elements. Information from very early elements may be overwritten in the "memory" of the RNN by the time later elements are processed. This can be problematic when encountering long sentences where the meaning of a word is dependent on far away sections of the sentence. Attention resolves this problem by accessing elements in parallel, instead of sequentially. Therefore, information does not need to be "remembered" anymore. The

[8] Multi Layer Perceptron

encoded elements are weighted combinations of other elements or themselves. There are various ways to compute the weights of the combination such as e.g., dot product between the to be encoded word and other words [Hu20].

Zheng et al. [ZLL+17] use self-attention in combination with Bi-LSTMs to extract interactions between drugs in the biomedical literature. They only use attention in the embedding layer to re-encode the word embeddings of input sentences. Each word becomes a weighted combination of all words in the input. The rest of the model is similar to RNN-based models discussed in Section 2.5.1. Zheng et al. found that input self-attention improved relation extraction performance for long range relations (50 or more words apart), while performing similarly for close range relations.

Zhang et al. [ZZL+18] also use self-attention to re-encode the input embeddings. However, their approach only attends to the candidate pair instead of all words in the input sequence. Word embeddings are scaled with weights that are derived from the dot product between the word embedding and the candidate pairs. Thus, words that are not similar i.e., small dot product, to the candidate pairs are scaled down compared to those that are.

2.5.4 Transformer

All the biomedical relation extraction models discussed so far in this section are based on RNNs. Introducing bidirectional processing in form of Bi-LSTMs or Bi-GRUs, as well as introducing attention mechanisms to these models greatly improved performance. Transformers [VSP+17] further improve on these models by forgoing recurrence and relying entirely on attention mechanisms. The result is that transformers are easier to parallelize and learn longer range dependencies in sequence modeling tasks than RNNs.

Transformer based neural networks are currently state-of-the-art for many NLP related tasks. Models such as GPT-3 [BMR+20] and XLNet [YDY+19] show never before seen natural language understanding in many NLP tasks, such as question answering and reading comprehension. Models based on BERT [DCLT19], such as SciBERT [BLC19] and BioBERT [LYK+19] showed state-of-the-art performance in a variety of biomedical NLP tasks.

Aside from being transformer based, state-of-the-art NLP models also use *transfer learning*. Transfer learning is a type of machine learning where a model is pretrained on a related task before it is adapted to the actual task. NLP models such as BERT can be pretrained on large text corpora e.g., PubMed Central articles on language modeling tasks. After pretraining BERT the model is trained again on the actual task e.g., biomedical relation extraction. Pretraining large models such as BERT on large text corpora is very expensive. Training BioBERT takes 23 days on eight NVIDIA V100 GPUs [LYK+19].

Peng et al. [PYL19] evaluated transfer learning models on ten biomedical language understanding tasks. The tasks included named entity recognition, relation extraction and document classification. They found that transfer learning in combination with transformer models outperformed transfer learning with RNN-based models. BERT

pretrained with the biomedical text corpora PMC and MIMIC [JPS+16] performed best on the ten biomedical language understanding tasks.

CHAPTER 3

Dataset

This chapter discusses how we created a dataset for the task of disease-symptom relation extraction. Section 2.2.2 already describes how to create datasets using manual labeling for various biomedical relation extraction tasks. As will later become apparent, our model architecture is designed with the structure of its input data in mind. For this reason we describe the process of annotating the dataset as well as the data itself.

We distinguish between two datasets in this chapter. First, a dataset used for model training which we call *finetuning dataset*. This dataset was manually annotated by us. Second, a dataset that we refer to as the *case report dataset*. The case report dataset is used for relation extraction using our model.

To illustrate the purpose of the two datasets we describe our relation extraction approach:

1. Annotate *finetuning dataset*, which is a small sample of *case report dataset* sentences
2. Teach model how to extract relations from *finetuning dataset*
3. Use model to extract relations from *case report dataset*

In the past, PubMed Central (PMC) articles were successfully annotated for various biomedical relation extraction tasks ([LSJ+16], [DLL14]). We decided to use case reports for disease-symptom relation extraction because they predominantly describe the relation between diseases and symptoms in real life patients. To avoid unnecessary domain adaptation, we also decided to sample the training dataset from the case reports themselves, instead of other sources.

3.1 Case Reports

Case reports are publications from healthcare professionals in the medical domain and serve as an educational resource for other healthcare professionals. Topics of case reports are, but are not limited to, the course of a pathology in one or more patients, clinical guidelines or aspects of differential diagnoses. Due to the goal of this work, we are most interested in case reports that describe the course of a pathology in a patient, along with diagnoses made by medical doctors that treated the patient.

These types of case reports often contain the following information:

- Age
- Gender
- Symptoms reported by the patient
- Observations made by the healthcare professional
- Medical history such as prior treatments, pre-existing medical conditions etc.
- Conducted laboratory tests
- Differential diagnosis

This information is available as *free form text* partitioned into sections such as abstract, background, differential diagnosis etc. The free text may be mixed with other forms of media such as images e.g., x-ray image, photo of the medical condition etc. Furthermore, each case report is accompanied by a title describing the topic of the report. The title usually contains the diagnosis or the name of the pathology.

3.1.1 Source

A quick internet search will reveal that there are dozens of medical case report journals to choose from. We decided to build our dataset from case reports found in the case report journal BMJ Case Reports. BMJ Case Reports[1] is a collection of clinical case reports.

There are numerous reasons why we build our dataset from BMJ Case Reports. First and foremost, BMJ Case Reports is one of the largest case report journals, allowing us to gather a large amount of case reports from a single source. The journal offers descriptions of real world cases from a wide variety of medical areas such as Oncology, Neurology and Infections Diseases. Furthermore, case reports published in BMJ Case Reports are peer reviewed and copy edited before publication. BMJ Case Reports is therefore a great source of medical information for our purposes.

[1] BMJ Case Reports — `https://casereports.bmj.com`

3.1.2 Advantages for Finetuning

There are a few reasons why we preferred to manually annotate case reports, instead of other biomedical text. One reason is that case reports have consistent structure due to the fact they are a specific type of biomedical literature. In contrast, sources like PMC are less homogenous, because they include more types of biomedical literature. Case reports often contain the same sections such as clinical representation or differential diagnosis. This gave us the option to remove sections with non-relevant information.

Furthermore, we expect that training on text from sources other than case reports will degrade the performance of the relation extraction model, because the goal of the relation extraction model is to extract relations from case reports. Machine learning models often assume that the training data has the same distribution as the test data. If we train the model on other sources e.g., PMC articles, we break this assumption and require domain adaptation. Domain adaptation [SSW15] is a subfield of machine learning that is concerned with accounting for changes in distributions between the source and the target data. By keeping the distribution of the training data and test data the same, we avoid the need for domain adaptation. Our expectation is reinforced by a preliminary inspection which showed that most case reports exhibit a similar writing style. For example, sentences of the following form

> We report a [**age**] year old [**person**] with [**symptom**].

appear frequently in case reports. It is therefore important that the finetuning dataset includes this type of writing.

3.1.3 Advantages for Disease-Symptom Relation Extraction

The reason why case reports are a good source of disease-symptom relations is that a large proportion of them aim to convey associations between diseases and symptoms to other healthcare professionals. Case reports are often documentation of treatments and observations made by medical doctors about a specific patient encounter. This type of documentation must necessarily emphasize diagnoses and symptoms. Additionally, the fact that case reports are peer reviewed and stem from consultations with healthcare professionals ensures a certain degree of quality.

Another advantage of case reports for disease-symptom relation extraction is that most, if not all, case reports mention the disease in the title. Thus, case reports convey the relation between diseases and symptoms not entirely through free text form. We take advantage of this fact in our model to simplify the relation extraction task.

3.1.4 Disadvantages for Disease-Symptom Relation Extraction

Due to the dynamics of scientific publishing, case reports tend to be about topics that are worthy of discussion to healthcare professionals. The consequence is that common

medical conditions are less likely to be published. In other words, we are more likely to find a case report about a rare form of cancer treatment than the treatment of an uneventful case of the common cold.

Another disadvantage of case reports is that they are only available in small numbers, compared to less specific biomedical literature sources such as PMC. The PMC Open Access Subset contains more than a million documents. On the other side, as of writing, the BMJ Case Reports Journal provided about 20,000 reports.

3.2 Annotation

The structure of case reports implies the disease-symptom relation i.e., title mentions the disease and the report body describes the symptoms. Extracting disease-symptom relations from case reports can therefore be simplified from a relation extraction task to a NER task. In other words, instead of mapping text to disease-symptom pairs, we map the title to diseases and the report body to symptoms separately.

Case report titles often consist of a short single phrase. We expected existing information retrieval tools such as MetaMap to achieve acceptable performance for extracting diseases from titles. We therefore did not annotate case report titles with diseases. In contrast, symptoms are described in paragraphs and may be scattered over the entire report. In order to extract symptoms we annotated case report text with symptoms. Our definition of symptom also includes descriptions of severity, duration and localization of the symptom. We also annotated age and gender due to project requirements from our sponsor. However, age and gender were removed for the purposes of this work.

We decided to annotate at the sentence level to be able to distribute the workload more evenly among annotators and thus more fairly. The length of case reports can vary from 100 words to multiple hundreds of words. Sentences were selected randomly for annotation for a fixed set of diseases. The sentences were sampled from abstracts from over 20,000 BMJ case reports. To improve efficiency of the annotation process, the labeling was done using a web-interface, developed specifically for the Amazon Mechanical Turk platform (Figure 3.1).

The steps involved to annotate a sentence are

1. Select one of the three available entities for Age, Gender or Symptoms
2. Highlight the words in the sentence to assign an annotation with the currently selected entity label
3. Submit the annotation

Figure 3.1: The annotation web interface used to annotate the dataset

3.2.1 Annotators

In order to ensure an acceptable quality, we recruited Amazon Mechanical Turk (MTurk) workers with the following qualifications

- Worker needs at least 500 accepted HITs[2]
- Worker needs at least an acceptance rate of 95% on previous tasks

Furthermore, before the workers were allowed to annotate the full dataset they had to pass a small test run. The MTurk workers had to annotate 10 sentences with a 50% inter-annotator agreement or more to pass the test. The inter-annotator agreement was measured using Fleiss' Kappa. The agreement was derived from sentences annotated by two medical experts. The expert annotators were medical doctors familiar with the jargon used in case report journals. In total, 27 workers qualified for the annotation of the finetuning dataset. To further increase accuracy of the annotations, each sentence was labeled by five workers, followed by a majority-voting. Labels were only accepted if the majority of the five annotators were in agreement about them.

3.3 Results

In this section we describe the two datasets we created. The role of the two datasets is described in the beginning of this chapter.

[2]HIT is a *Human Intelligence Task*

3.3.1 Finetuning Dataset

In total, 2,447 sentences were annotated by the MTurk workers. The inter-annotator agreement (Fleiss' Kappa) between the expert annotators and the crowd-worker annotations was 0.82 for labeling symptoms. The most common annotation disagreements were due to different views on what symptoms are. For example, concepts such as *chronic coughing* could be interpreted as symptoms or diseases. Other disagreements resulted from our definition of symptoms. We expected workers to include severity, duration and localization of the symptoms in the symptom annotation as well. Complex symptoms such as "Intermittent sharp pain on the back of the head over the last two weeks" were less agreed upon.

3.3.2 Case Report Dataset

We also gathered 20,414 case reports from BMJ Case Reports for the purpose of extracting relations from them. Since we are only interested in symptom descriptions, we omitted all paragraphs with headings other than "abstract", "description" and "case representation" from the case reports. The result is a set of case report titles associated with their contents. The titles are often just a single sentence mentioning the disease the case report discusses. The length of the case reports themselves ranges from a single sentence to 181 sentences. The average number of sentences in a case report is 15, excluding the title.

CHAPTER 4

Model

This chapter discusses the model architecture and the vocabulary used to represent symptoms and diseases. The final section in this chapter summarizes the model and discusses differences between our model and models discussed in Chapter 2.

4.1 Medical vocabulary

So far in this work we referred to symptoms, diseases and other medical concepts using the English language. However, the medical terminology is complex and different terms can refer to the same concepts e.g., *abdominal pain* and *stomachache* refer to the same concept. To address this problem, standardized language, thesauri, ontologies or vocabularies may be used. A medical vocabulary allows us to communicate medical concepts clearly and distinctly between humans and computer systems alike. There are a number of vocabularies suited to the task of relation extraction.

We chose the MeSH thesaurus for this work. But, because MeSH is a subset of UMLS and parts of the processing pipeline involve UMLS, we describe UMLS as well.

4.1.1 Unified Medical Language System (UMLS)

UMLS is a set of files and software that integrate many health and biomedical vocabularies. The biggest part of UMLS is the metathesaurus [SHTS93]. The UMLS metathesaurus is organized by concepts and aggregates nearly 200 biomedical vocabularies. In total, there are over a million biomedical concepts in the metathesaurus. Concepts are identified by a *Concept Unique Identifier* (CUI). Since the vocabularies within UMLS provide their own identifiers, UMLS also provides a mapping from each concept to CUIs. Aside from a CUI, each UMLS concept has attributes specifying its meaning, as well as relations to other concepts. Among its relations are *isa* and *ispartof*, which are used to create a hierarchy of concepts. The meaning of a UMLS concept is defined by its semantic type.

A semantic type is a broad entity category such as “Chemical”, “Temporal Concept” or “Pathologic Function”. UMLS defines more than 100 semantic types and integrates them in a shallow entity hierarchy.

4.1.2 Medical Subject Headings (MeSH)

MeSH is a thesaurus and hierarchically organized vocabulary produced by the National Library of Medicine in the United States of America [Lip00]. Its concepts are a subset of UMLS. Similar to UMLS each MeSH concept has a unique identifier code. Furthermore, concepts are taxonomically and hierarchically organized as a tree. The location of a concept in the tree is defined by an alphanumerical code with dots, called tree number. To illustrate how tree numbers are constructed, we show in Figure 4.1 the concept *Appendicitis* with tree number C01.463.099 in the MeSH tree hierarchy. Note that a concept may appear multiple times in the hierarchy because it has multiple uses e.g., as a symptom and a disease.

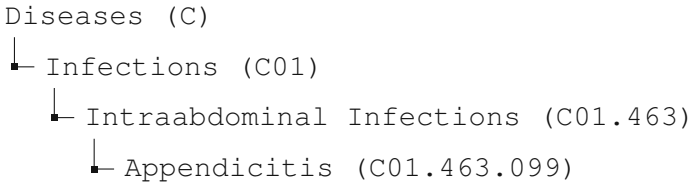

Figure 4.1: Path of one tree number for the concept *Appendicitis* in the MeSH tree

The smaller size of MeSH is a disadvantage over UMLS, as more specific diseases and symptoms are missing. In our experience, this occurred frequently when symptoms are compounded with qualifiers such as *sudden chest pain* compared to regular *chest pain*. But the smaller size is also an advantage because it leads to less fragmentation of concepts e.g., UMLS has multiple concepts of fever such as *fever* (C0015967) and *fever symptom* (C0424755). Ultimately, the level of granularity necessary depends on the application.

Ultimately, we preferred MeSH over UMLS in our work for the following reasons. Concepts in MeSH are part of a deep tree hierarchy which future applications may take advantage of e.g., generalization of concepts via the hierarchy. Furthermore, the thesaurus can be downloaded in its entirety in a standard format such as RDF without the need of a verification step. The official website provides an easy and accessible way to browse the thesaurus[1]. The coverage of the vocabulary was sufficient for our task, as it contained the most common diseases and symptoms.

4.2 Architecture

Our approach is dependent on the fact that we are extracting relations from case reports. Chapter 3 went into detail how case reports are structured. The structure of case reports

[1] MeSH Browser — https://meshb.nlm.nih.gov/search

allows us to treat the task of relation extraction as a NER task.

Figure 4.2 gives a high level overview of the model. The input of the model is a case report and the output is a set of disease-symptom relations. The model consists of three modules which will be discussed in detail in the remainder of this section. The first two modules, concept annotation and symptom detection can work in parallel. The final module aggregates the results from both modules to extract the relations. The result is a set of disease-symptom relations.

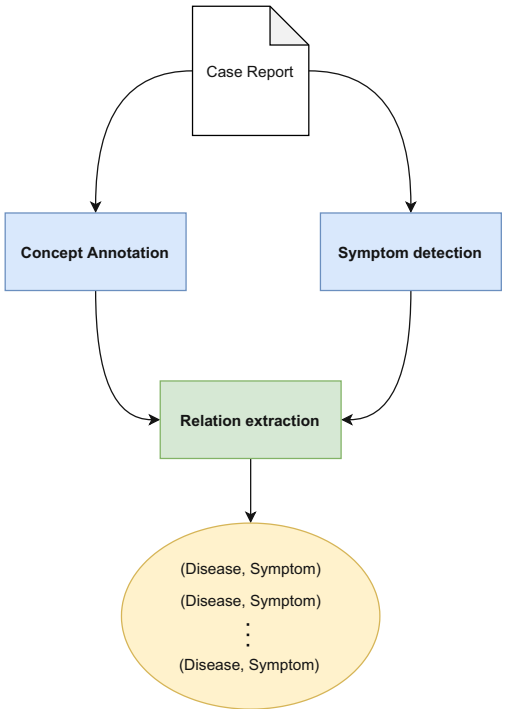

Figure 4.2: Full disease-symptom relation extraction model

4.2.1 Concept annotation module

This part of the model recognizes and labels medical concepts in the title and body of the case report. The purpose of this part is similar to the generation of candidate pairs discussed in Section 2.5. We preprocess the text by annotating it with medical concepts to generate entities that may be part of a disease-symptom relation i.e., candidates for a disease-symptom relation.

We decided to use the well established biomedical information retrieval and data mining software *MetaMap* [Aro01]. MetaMap is developed by the National Library of Medicine (NLM) in the United States. It is capable of mapping natural language text to concepts of the UMLS metathesaurus.

MetaMap has been available for more than a decade, and has been gradually improved over the years. Its approach to processing text is based on symbolic, NLP and computational-linguistic techniques. Discussing the inner workings of MetaMap in detail is out of the scope of this thesis. The most relevant steps in the NLP pipeline for our usage of MetaMap are shown in Figure 4.3.

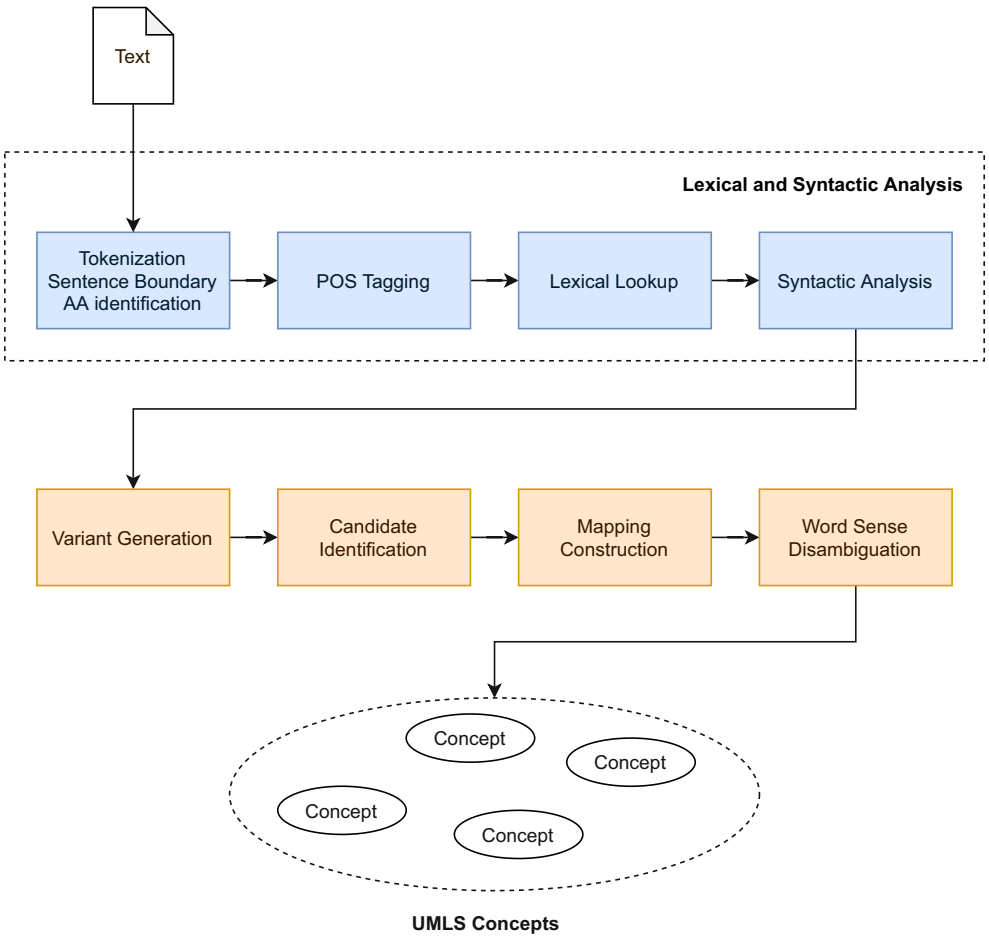

Figure 4.3: MetaMap processing pipeline from text to UMLS concepts [AL10]

The rest of this section discusses the hyperparameters we used for MetaMap.

Semantic Types

By default, MetaMap annotates texts with all the concepts in the UMLS metathesaurus. In order reduce the chance of annotating text with incorrect or irrelevant labels, we only used concept types that are related to symptoms. As discussed in Section 4.1.1, semantic types are used to categorize concepts based on their meaning. We selected the semantic types shown in Table 4.1 to cover concepts relevant to symptoms and diseases.

Abbreviation	ID	Full Name
sosy	T184	Sign or Symptom
dsyn	T047	Disease or Syndrome
inpo	T037	Injury or Poisoning
mobd	T048	Mental or Behavioral Dysfunction
fndg	T033	Finding
inpr	T170	Intellectual Product
anab	T190	Anatomical Abnormality
cgab	T019	Congenital Abnormality
neop	T191	Neoplastic Process
patf	T046	Pathologic Function
irda	T130	Indicator, Reagent, or Diagnostic Aid
lbtr	T034	Laboratory or Test Result
socb	T054	Social Behavior
menp	T041	Mental Process
bsoj	T030	Body Space or Junction
bpoc	T023	Body Part, Organ, or Organ Component
blor	T029	Body Location or Region
spco	T082	Spatial Concept

Table 4.1: Subset of UMLS semantic types used for concept annotation

UMLS subsets

We were only interested in concepts from the MeSH thesaurus therefore we restricted MetaMap to only use UMLS concepts also found in MeSH.

MetaMap Dataset

The number of diseases increases every year. In order to detect as many diseases as possible the vocabulary used by MetaMap needs to be up to date. We used the updated MetaMap dataset *2020AB*. Most notably, this version includes COVID-19.

Word Sense Disambiguation

In order to resolve ambiguity when mapping from text to concepts we enabled *Word Sense Disambiguation*. Ambiguity occurs when two or more concepts share a common synonym e.g., “cold” could be mapped to the concepts *cold temperature* or *common cold*.

Negation detection

In case reports, it is common to not only discuss the existence of symptoms but also the non-existence. For example, a case report can explicitly state that a symptom is not present in the patient. Thus, a relation between disease and said missing symptom

should not be formed. For this reason, we used the default negation detection algorithm from MetaMap called *NegEx*.

Term processing

MetaMap was originally created to process full articles in the biomedical literature and chunks its input into phrases e.g., noun phrases, prepositional phrases etc. All phrases are analyzed separately. Term processing on the other hand disables this behavior, and the text will be processed as a single phrase. This is desirable when the inputs are short text snippets, e.g., "recurrent headaches". Case reports, however, may contain long and complex sentences. Therefore, we disabled term processing.

4.2.2 Symptom detection module

The purpose of this module is to detect where in the text symptoms experienced by a patient are mentioned. We represented this problem as a NER problem. This module of the model is a neural network based on the transformer architecture. As presented in Section 2.5.4, transformer based neural network architectures achieve state-of-the-art performance on many NLP related problems.

Figure 4.4 shows the architecture of the module and how it processes a sentence. The model receives a sentence as input. The sentence is then tokenized into subwords. The tokenization is based on a vocabulary that is learned during pretraining of the BERT model. If the word is not part of the vocabulary, it will be split into subwords that are part of the vocabulary. Each subword is then embedded and contextualized through the transformer encoder and decoder network. The last hidden state for each subword from the transformer network is $\vec{h_i}$, where $i \in \{1, \ldots, n\}$ and n is the number of subwords the input sentence was split into. The number of dimensions of $\vec{h_i}$ is chosen to be 768 based on the original BERT paper. The last hidden state $\vec{h_i}$ is then transformed by the classification module. The classification layer contains a dropout layer with a dropout rate of 10%, as well as a linear combination layer. Just like the hidden state size, the dropout rate is based on the BERT paper. The output of the classification layer is the activation for each subword $\vec{y_i}$. The number of dimensions of the activations are 2: one dimension for the existence of symptoms and the other dimension for the non-existence. It is possible to use a single dimension as well, as the activation of one class implies the opposite for the other class. However, the current design can be easily extended to more than just two opposite classes e.g., adding age and gender classes. Finally, the class the subword belongs to is defined by the dimension with the maximum activation.

The symptom detection module is a neural network, thus it requires training to set model parameters. The following subsections discuss how the neural network is trained.

Pretraining

Instead of training the large transformer neural network BERT ourselves, we used the pretrained SciBERT [BLC19]. As the name implies, SciBERT is based on BERT

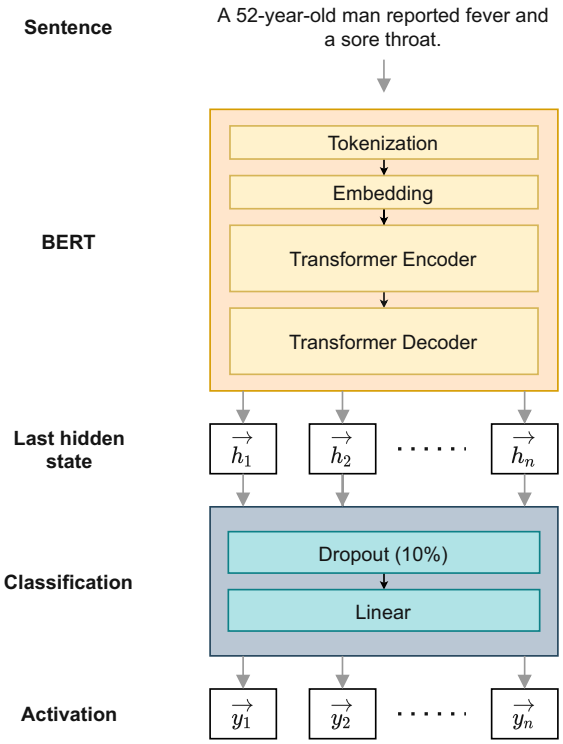

Figure 4.4: Neural network architecture of symptom detection module

[DCLT19]. The crucial difference between SciBERT and BERT is that it is pretrained with scientific literature to improve performance for downstream tasks in the scientific domain. Furthermore, we chose the uncased version of SciBERT instead of the version with a capitalized vocabulary.

Tokenization, collation and batch size

Neural networks based on BERT expect text to be tokenized into subwords. The dataset we use for training the neural network is labeled on a word level. This introduces a problem, since there are multiple ways to convert word level labels to subword level labels. To solve this problem we can assign labels corresponding to a word to all its subwords i.e., repeating the label. Another possible solution is to split each label into *head* and *tail* labels. For example, if the word "headache" was labeled with `symptom` and split into "head" and "ache", then the new labels could be `symptom_head` and `symptom_tail`. We chose the former solution to the problem because it requires fewer labels, and we found no evidence that one is preferable over the other.

BERT is not able to process text of arbitrary length. We chose to truncate sentences with more than 512 subwords or tokens. This is in line with the experiments from the original BERT paper. If sentences have less than 512 subwords, their representation will be padded with zeros. Furthermore, we chose to omit the sentence token `[CLS]`, as we

are only interested in making token level predictions and not sentence level predictions. We therefore have no use for a contextualized `[CLS]` token.

Due to the large size of BERT it quickly becomes computationally challenging to train with large batch sizes. One of the main purposes of larger batch sizes is to decrease training time. Given that our training dataset is relatively small we were able to achieve acceptable training times with a small batch size of 16.

Finetuning

This section describes how we train the symptom detection module.

The training of the model is often called finetuning because we are modifying already learned neural network parameters for our specific task. In line with the experiments from the authors of BERT and SciBERT we used the Adam optimizer [KB17] to optimize network parameters. To be exact, we used AdamW which decouples weight decay from the gradient update. This improves generalization performance and reinstates the original definition of weight decay regularization [LH18]. The loss function used for training is Cross-Entropy Loss.

The authors of BERT experimented with smaller learning rates for finetuning in the range of $[1e{-}5, 1e{-}4]$. Due to the rather small size of our training dataset a learning rate this small had no negative effect on training time. Furthermore, our initial experiments showed that larger learning rates e.g., $1e{-}3$, prevented the model to learn at all.

Due to the small size of the dataset we were able to train models for multiple epochs. We use an early stopping mechanism to determine the number of training epochs. If the validation performance of the model drops then the training is stopped and the second to last model is picked as the final model. The split between training and validation data is 80% and 20%. This training method was used to prevent the model from overfitting to the dataset, and generalize better to unseen data.

4.2.3 Relation extraction module

This module of the relation extraction model processes the results from the concept annotation module and the symptom detection module to extract relations. The main steps are shown in Algorithm 4.1.

The first step of this module is to remove concepts that are not symptoms or diseases. If a concept annotation from MetaMap appears in the title and is not in one of the MeSH disease subtrees then it is removed. If a concept annotation appears in the case report body, and it is not in one of the MeSH symptom subtrees, then it is removed. Table 4.2 shows which subtrees were used to determine whether a concept was a disease or a symptom.

Furthermore, if concepts are reported to be negated by MetaMap's NegEx algorithm, they are also removed. Finally, the remaining symptom concepts are matched against

Disease Patterns	Description
F03	Mental Disorders
C(?!22|23.888)	Diseases without Animal Diseases, Signs and Symptoms
Symptom Patterns	**Description**
C23.888	Signs and Symptoms
F01.145(?!.209.113)	Behavior without Communication, Animal Behavior
F01.470	Emotion

Table 4.2: RegEx patterns to match disease and symptom concepts based on their MeSH tree number

the symptom detection labels. If 50% or more of a concept overlaps with the symptom detection labels from the neural network, then they are considered to be symptoms. The overlap is the intersection of the character sequence from the concept annotation and the character sequence from the symptom labels. Finally, the cross product between remaining symptom and disease concepts are the disease-symptom relations. In other words, each remaining disease is paired with each remaining symptom. The set of all pairs are the disease-symptom relations we extracted. The reason we use the cross product, is because the title may contain more than one disease in some cases.

Algorithm 4.1: Extract relations from case report

Input:

- *TitleConcepts* found in case report by concept annotation module
- *BodyConcepts* found in case report by concept annotation module
- *SymptomLabels* from symptom detection module

Output: Disease-Symptom Relations from case report

```
1  Symptoms, Diseases, SymptomCandidates = ∅, ∅, ∅
2  forall concept ∈ TitleConcepts do
3  |  if concept ∈ MeSH disease subtrees ∧ concept not negated then
4  |  |  add concept to Diseases
5  |  end
6  end
7  forall concept ∈ BodyConcepts do
8  |  if concept ∈ MeSH symptom subtrees ∧ concept not negated then
9  |  |  add concept to SymptomCandidates
10 |  end
11 end
12 forall candidate ∈ SymptomCandidates do
13 |  if characterSequenceOverlap(candidate,SymptomLabels) > 50% then
14 |  |  add candidate to Symptoms
15 |  end
16 end
17 Relations = Diseases × Symptoms
18 return Relations
```

4.3 Summary

Our model consists of three main parts: concept annotation, symptom detection and relation extraction. The concept annotation module annotates case reports with MeSH concepts. The symptom detection module detects which parts of the text describe symptoms experienced by patients. The relation extraction module filters out all concepts from the case report body that are not symptoms experienced by patients, and all concepts from the title that are not diseases. We use the MeSH tree hierarchy to determine the

type of concept i.e., whether the concept is a symptom or another medical concept. Finally, all symptom and disease concepts that are left are connected as pairs to form disease-symptom relations.

Our model differs from the models we reviewed in Chapter 2 in the following ways. First, our model only requires a training dataset annotated with symptoms. Other approaches require symptoms and diseases to be annotated, as well as which pairs form a relation. The reason for this difference lies in the format of case reports. Case reports discuss single diseases which are often mentioned in the title. Thus, when the case report describes patients and their symptoms, they are related to the disease mentioned in the title.

Finally, our model goes further by also mapping relations to concepts in the medical vocabulary MeSH. The approaches we reviewed in Chapter 2 only concern themselves with detecting the type of relation and the position of the relation in the text. By outputting relations that are mapped to a shared medical vocabulary (e.g. MeSH), we can more easily integrate the relations into other medical information processing applications e.g., knowledge databases.

CHAPTER 5

Evaluation

This chapter evaluates the performance of our disease-symptom relation extraction model described in Chapter 4. We extract disease-symptom relations from more than 20,000 case reports (Section 3.3.2) using our model and compare the extracted relations with a ground truth created by human medical professionals. Furthermore, we also evaluate the hyperparameters of the symptom detection module. Both of these evaluations are described as experiments in Section 5.1.

5.1 Experiment design

We conduct two experiments for the evaluation in the following order

1. Symptom detection experiment
2. Relation extraction experiment

The first experiment's purpose to select model hyperparameters for the symptom detection module. The second experiment evaluates performance of the relation extraction model. The second experiment depends on the model hyperparameters selected in the first experiment, because the symptom detection module is part of the relation extraction model.

5.1.1 Symptom detection experiment

The goal of this experiment is to find the best performing hyperparameters for the model that will become the symptom detection module. While this experiment does not optimize for disease-symptom relation extraction directly, we expect that good performance in this experiment will translate to good performance in relation extraction or the second

experiment. The reason why the symptom detection module is evaluated separately in the first experiment, is because of the combinatorial explosion of hyperparameters combinations. Evaluating the entire model on the relation extraction experiment for each hyperparameter combination is computationally infeasible for this work.

In order to find the best performing hyperparameters, we train and test the model with different hyperparameter combinations on the *finetuning dataset* described in Section 3.3.1. The dataset consists of 2,447 sentences where words are labeled whether they describe symptoms experience by a patient or not. The task for the model in this experiment is to accurately detect which parts of the sentences describe symptoms. This task falls under the category of NER. To summarize, this experiment evaluates multiple hyperparameter combinations of the symptom detection model on a symptom detection task, in order to find the best set of hyperparameters.

Model hyperparameters & Experiment parameters

This section presents the model hyperparameters and experiment parameters we used in this experiment. All parameters are shown in Table 5.1 and evaluated as a grid search. The remainder of this section describes all the hyperparameters and experiment parameters.

We evaluate two domain specific pretrained models, SciBERT [BLC19] and ClinicalBERT [AMB+19], as well as the base BERT model [DCLT19]. Section 2.4.5 discusses the rationale of domain specific pretrained models.

The classification layer of the neural network can be either a linear layer or a Conditional Random Field (CRF). The neural network does not change otherwise, except for the loss function. The CRF layer uses a Negative Log-Likelihood loss function, instead of a Cross-Entropy loss function. This is due to implementation details.

We varied learning rate and bias correction for the gradient descent optimization with the AdamW algorithm. We only evaluate small learning rates like the authors of BERT [DCLT19]. The learning rates are 1e−5 and 3e−5. Furthermore, we evaluate training with enabled bias correction and without it. Adam uses the moving average of the gradients during training. Moving averages introduce a bias at the beginning of training which bias correction removes.

We evaluate each hyperparameter combination for three random train/test splits[1] (80% training, 20% testing) and three random seeds for the random generator[2] used in the neural network parameter optimization process. Therefore, each hyperparameter combination is evaluated $3 * 3 = 9$ times. The seeds are shown in Table 5.1. This approach reduces the chance of skewed test datasets or unfavorable optimization algorithm initialization, because we evaluate a distribution instead of a point estimate of the performance. Furthermore, we note that we use random subsets instead of k-fold cross-validation

[1] We use the `random_split` function from the PyTorch Python package `torch.utils.data`
[2] We use the `torch.manual_seed` function from the PyTorch Python library

because of the lower computational requirements. Due to the low amount of finetuning data we want to reserve more training data (80%) for the neural network. Cross-validation requires five instead of three evaluations if the training split is supposed to be 80%, thus more computation.

Lastly, the neural network receives the input in small batches of size 16. Larger batch sizes require more GPU memory than what is available to us.

Model Hyperparameter	Values
Pretrained model	BERT Base (uncased), SciBERT (uncased), ClinicalBERT (uncased)
Classification layer	Linear, CRF
Learning rate	1e−5, 3e−5
Bias correction	false, true
Experiment Parameter	**Values**
Dataset train/test split seed	3044, 21563, 11995
PyTorch seed	47026, 7206, 32277
Batch size	16
Dataset split	80% training and 20% test

Table 5.1: Hyperparameters of model and parameters of symptom detection experiment. The experiment is a grid search over all the parameters in this table. Some parameters have multiple values (e.g. learning rate), which are separated by commas.

Evaluation

This section describes how we evaluate the results of the experiment.

Since we represent the problem of symptom detection as a NER problem, we measure performance with classification metrics. We use the *f1-score* metric, because it provides a balanced measure of *precision* and *recall*. Also, we use macro averaging of the f1-score for the two classes, symptom and no symptom.

Finally, we evaluate the hyperparameters *pretrained model*, *classification layer*, *learning rate* and *bias correction*, to find the best values for the final symptom detection model. The experiment evaluates 24 hyperparameter combinations, along with 3 train/test splits and 3 model seeds. Thus, the experiment results are $24 * 3 * 3 = 216$ f1-scores achieved on the test splits of the finetuning dataset. To find the best value of each hyperparameter we compare the conditional distributions of f1-scores for each hyperparameter value.

To exemplify what this means, we explain how we find the best value for the hyperparameter *bias correction*. Bias correction has two possible values, *true* or *false*. Therefore, half of the 216 f1-scores stem from model runs with bias correction set to *true*, and the other half stems from model runs with bias correction set to *false*. The two sets

of f1-scores are conditional performance distributions, because bias correction is fixed while other parameters are variable. In order to find out whether *true* or *false* results in higher performance, we compare medians of both conditional performance distributions. We compare the two distributions using a box plot and a median hypothesis test. Box plots show us the median f1-score as well other quantiles of the performance distribution. Even though a box plot is sufficient to determine which sample median is higher, we also use the one-sided Wilcoxon signed-rank test to determine whether the median difference is significant. We consider a p-value less than 5% to be significant. If two hyperparameter values do not significantly differ in their median f1-score, we choose the value with the higher box plot upper fence. The upper box plot fence is computed using $Q_3 + 1.5(Q_3 - Q_1)$, where Q_1 and Q_3 are the first and third quartiles. The reason why we use the Wilcoxon signed-rank test is because we compare two dependent samples i.e., the two models we compare are trained on the same dataset. Lastly, we empirically analyze errors made by the model on the test sets.

To summarize, the final symptom detection model uses the hyperparameters that achieved the highest median f1-score on the test data splits in the experiment. If no significant difference is present, we choose the value with the higher box plot upper fence.

5.1.2 Relation extraction experiment

This experiment evaluates the performance of the relation extraction model described in Chapter 4. We extract disease-symptom relations from over 20,000 case reports described in Section 3.3.2. Then extracted relations are grouped per disease d. The result is that each disease d corresponds to a list of symptoms S_d. The order of the symptoms $s \in S_d$ is defined by how many case reports have the relation (d, s). In other words, each disease corresponds to list of symptoms ordered by often the symptoms co-occurred with the disease. We then measure the relevance of the symptoms recommended by our model i.e., the ranked list, by comparing it to a ground truth curated by medical doctors.

5.1.3 Evaluation

The ground truth is a ***D**isease **S**ymptom **R**elations Collection* (DSR-collection) created by two physicians [ZHRH20]. The medical doctors collected disease-symptom relations for 20 diseases and graded them by relevance. Primary symptoms receive the grade 2 and the other symptoms receive the grade 1. This allows us to partially order symptoms by relevance for all 20 diseases. The 20 diseases are shown in Table 5.2.

We measure the agreement between the model generated disease-symptom relations and the DSR-collection using *nDCG@k*, *precision@k* and *recall@k*, where k is the length of the ranked list. The ranked list contains the k most relevant disease-symptom relations. In order to be able to compare our model with the baselines provided with the DSR-collection, we set the constraint $k \in 5, 10$.

DSR-collection diseases	
Anorexia Nervosa	Appendicitis
Asthma	Bronchitis
Cholecystitis	COPD
Diabetes Mellitus	Epididymitis
Erysipelas	GERD
Influenza	Measles
Mental Depression	Migraine Disorders
Myocardial Infarction	Periodontitis
Pulmonary Embolism	Sleep Apnea Syndromes
Tonsillitis	Trigeminal Neuralgia

Table 5.2: List of 20 diseases from the DSR-collection [ZHRH20]

Baselines

The first baseline is a method proposed by Shah et al. [SLK+18] which we call Embedding. The method starts by replacing all occurrences of diseases and symptoms in the text corpus with unique names. The text corpus consists of 1.5 million full text PMC articles. They use MetaMap [Aro01] to detect the diseases and symptoms in the text. The next step is to train a word2vec model [MSC+] on the text corpus, such that each disease and symptom is mapped to a vector representation. The replacement in the first step ensures that synonyms of diseases and symptoms share the same embedding. The word2vec model uses a window size of 15 and 300 dimensions for the vector representation. Finally, ranked lists of symptoms are created for each disease. The order of the symptoms is defined by the cosine similarity between the disease embedding and the symptom embedding. Symptoms that are more similar rank higher in the list.

The second baseline is a method proposed by Zhou et al. [ZMBS14] and adapted by Zlabinger et al. [ZHRH20]. Zlabinger et al. named the baseline CoOccur-FullText which we refer to as CoOccur in this work. Using MetaMap, this method identifies all diseases and symptoms in 1.5 million PMC articles. Similarly to the first baseline, a list of symptoms is generated for each disease. Symptoms that co-occur more often with the disease rank higher in the list. A disease and a symptom co-occur if the disease appears in the title and the symptom appears in the body of a PMC article. Thus, co-occurrence is defined by the number of articles a disease and a symptom occur in together.

5.2 Symptom detection experiment results

This section presents the results of the symptom detection experiment. We present which hyperparameters outperformed others and which ones showed no significant difference. Using these insights, we select hyperparameters for our symptom detection module. Lastly, we empirically analyze errors made in the experiment by the model with the

selected hyperparameters.

5.2.1 Hyperparameter evaluation

The distributions of f1-scores for the test split per hyperparameter are shown in Figure 5.1.

The upper left box plot shows that SciBERT outperforms the other two pretrained BERT models. The median f1-score for models with SciBERT is 87%, whereas both BERT Base and ClinicalBERT models resulted in a f1-score of 85%. Furthermore, the box plot also suggests that the performance variance is largest for the Base BERT model, as the box plot whiskers cover almost 10%. The Wilcoxon signed-rank test statistic also shows significant median difference between SciBERT and the other two models, with p-values smaller than 1e−6. Bias correction also improves the test performance of the model significantly with a p-value of 2.5%.

The difference between using a linear and a CRF classification layer is insignificant. Neither the box plot nor the hypothesis test with a p-value of 69%, show a difference in the median. We also find no significant difference between the two learning rates with a p-value of 5%.

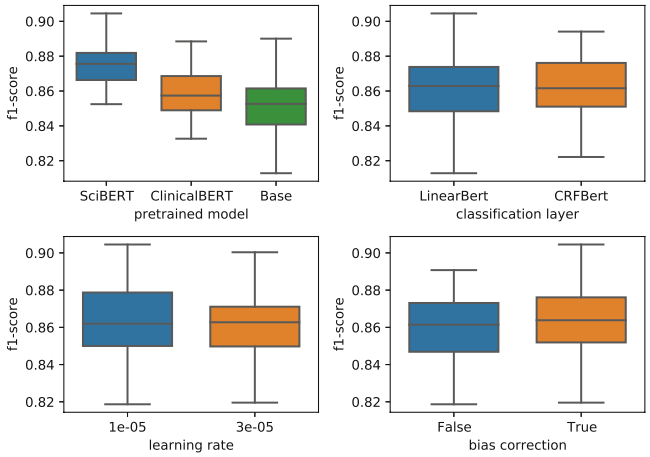

Figure 5.1: Conditional distributions of f1-scores per hyperparameter from the symptom detection experiment. Each of the four box plots contains the same 216 f1-scores.

5.2.2 Model selection

Using the results described in the previous section we select a model with SciBERT, linear classification layer, learning rate of 1e−5 and bias correction. We prefer the linear classification layer over the CRF layer because the upper fence of the performance distribution is higher. For the same reason, we choose the smaller learning rate 1e−5.

The average test performance in this experiment for the selected model is shown in Table 5.3.

Precision	Recall	F1-Score	Training Epochs
0.882	0.876	0.879	4

Table 5.3: Median performance of the best model in the symptom detection experiment from $3 * 3 = 9$ runs (3 dataset splits and 3 initial random seeds). The best model uses the hyperparameters SciBERT, linear classification layer, learning rate 1e−5 and bias correction.

5.2.3 Error analysis

The selected model misclassified 9.4% of the sentences from the test dataset fully i.e., none of the symptoms were found or the sentences did not contain any symptoms despite the model suggesting so. 16.3% of the sentences were partially misclassified i.e., at least one word was misclassified.

Our empirical analysis shows three main types of errors, which we describe in the following paragraphs

1. Inconsistent labeling of symptom qualifiers
2. Mistaking diseases for symptoms
3. Not finding any symptoms at all

Many symptom descriptions contain qualifiers such as duration, location or severity. For example *sudden pain* instead of simply *pain*. Failing to correctly label such qualifiers was a common type of error. However, we also find that the dataset was inconsistently labeled by the annotators. Figure 5.2 shows two errors the model made on the test dataset. The first error is due to a misclassification, the second is due to inconsistent labeling by the annotators.

Another source of errors was classifying diseases as symptoms. For example, the model predicted “acute renal failure” or “early onset of dementia” as symptoms in the test dataset. But, similar to the previous error type, this error was also found in the ground truth. The annotators were inconsistent with their distinction of symptoms and diseases.

The last type of error we noticed was failing to find any symptoms at all. We found that the model was very conservative or cautious with its predictions. This is reflected in the confusion matrix of the test classifications. Only 4% of words that do not describe symptoms, according to the ground truth, were labeled to be symptoms. In contrast, 10% of words that describe symptoms were predicted to not describe symptoms. In other words, the model failed to find symptoms more often than it predicted symptoms where there were none.

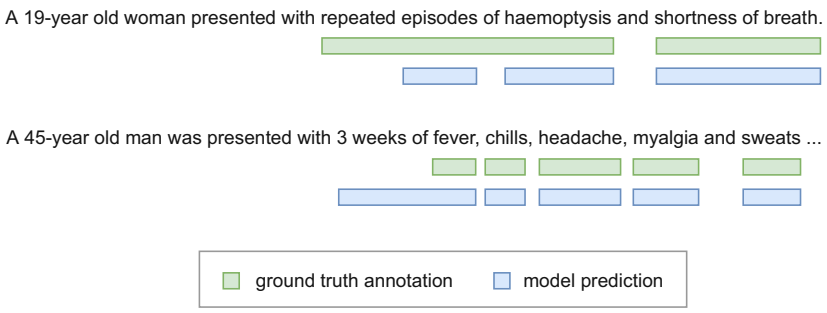

Figure 5.2: The first sentence shows a misclassification error on the model side, and the second sentence shows incorrect labeling of the ground truth on the annotators side.

5.3 Relation extraction experiment results

Our model improves upon the baselines in all metrics but recall@10 and precision@10, and the metrics that do not improve are close to the best baseline. The performance metrics for this experiment are shown in Table 5.4.

We noticed that our model extracted only very few relations for some diseases of the DSR collection. After searching for some of these diseases using the official BMJ Case Reports website, we concluded that some diseases have far few case reports about them. For example, there are only two case reports in BMJ Case Reports where erysipelas is declared as the diagnosis in the title. Thus, only very few relations can be extracted about these diseases. This skews the performance evaluation, as we evaluate the dataset more than the model. To mitigate this issue, we also measure the agreement for a subset of the 20 diseases. We select only diseases with relations from 10 or more case reports, which leaves us with 7 out of the 20 diseases. The performance of our model on this subset increases for all metrics. We observe the largest increase in the metrics precision@5, precision@10 and nDCG@5. Precision@5 increased by 24%, while precision@10 and nDCG@5 increased by 19%.

Model	nDCG@5	P@5	R@5	nDCG@10	P@10	R@10
Embedding	0.20	0.18	0.08	0.19	0.15	0.13
CoOccur	0.41	0.39	0.17	0.36	0.28	0.25
Our model	0.45	0.42	0.18	0.47	0.28	0.23
Our model (subset)	0.64	0.66	0.27	0.57	0.47	0.38

Table 5.4: Performance on the DSR-collection for baselines Embedding, CoOccur and our model. We also present performance of our model for a subset of diseases described in Section 5.3.

CHAPTER 6

Conclusion

This chapter presents conclusions we draw from the experiments we conducted and how to adapt our relation extraction model to sources other than case reports.

6.1 Conclusions

6.1.1 Disease-Symptom relations extracted by our model are more relevant

Our model outperforms the baselines in the relation extraction experiment in almost all metrics. If we restrict the evaluation to diseases with 10 or more case reports in our dataset, the agreement with the ground truth provided by the medical doctors further increases. We achieve up to 64% nDCG@5 and 66% precision@5. We conclude that our approach is an effective way to extract disease-symptom relations from case reports.

6.1.2 Our model requires only symptom annotations

An advantage of our approach is that it only requires symptom annotations for training. Other biomedical relation extraction models require disease, symptom and relation annotations. On the other hand, our model is not capable of extracting relations from sources other than case reports, at least not without modification. We discuss how to adapt our model for other types of text data in Section 6.2.

6.1.3 Our model values precision more than recall

The relation extraction experiment resulted in 42% precision@5, which is more than double the 18% recall@5 our model achieved. One of the motivations we presented in the beginning of this work was evidence based medicine. We believe that biomedical relation

extraction tools are more valuable in these types of tasks if they reduce the number of false positives. For this reason we believe that this model behavior is desirable.

6.1.4 MeSH vocabulary is too small for disease-symptom relation extraction

Upon further inspection of the DSR-collection we noticed that 36% of the symptoms were not part of MeSH. Consequently, 36% of the symptoms were impossible to predict for our model, as we are using MeSH to represent symptoms.

We also inspected which symptoms were part of DSR but not part of MeSH. Examples, are *sleep disturbance*, *testicular pain*, *dry cough* and *unable to concentrate*. We believe an effective disease-symptom relation extraction model should be able to represent these symptoms. Some of these symptoms can be represented using multiple MeSH concepts. However, doing so would require a change in the model architecture. Even after the change, not all symptoms can be represented. Therefore, we conclude that MeSH is too small for disease-symptom relation extraction.

6.1.5 Number of case reports has large impact on model performance

The fact that removing diseases with low number of case reports improves the performance in the relation extraction experiment, highlights the influence of the dataset. BMJ Case Reports, one of the largest case reports journals, provided slightly more than 20,000 case reports. Considering the vast number of different diseases, we cannot expect to have more than just a few case reports per disease.

6.1.6 Adding more case reports is not an easy way to improve model performance

We also considered collecting case reports from other journals. In 2015, at least 160 case report journals existed [Ake16]. However, most case report journals are smaller than BMJ Case Reports. Thus, increasing the number of sources does not increase the number of case reports proportionally. Furthermore, automating the collection of case reports is time-consuming, as access and licensing is handled differently by each journal. Therefore, we conclude that collecting more case reports is not an easy way to further increase performance of our model.

6.2 Future work

While our model is designed to extract disease-symptom relations from case reports, it can be easily adapted to other kinds of medical text, or even other types of relations e.g., drug-drug. The current limitation stems from the fact that we only have a training dataset with symptom annotations, but no disease annotations. Additionally, case reports

often mention the disease in the title, reducing the need for more complex models which are capable of extracting relations from a single body of text.

In its current state the symptom detection module is responsible for detecting symptom descriptions. However, the neural network that is the symptom detection module can be easily adapted to predict more than symptoms and diseases. We can achieve this by simply adding another dimension to the output of the neural network, because the symptom detection module solves a classification task.

In addition to the new disease dimension of the neural network output, we also need a new dataset. The new dataset must be annotated with the relations in mind. This means that annotators must annotate symptoms and diseases in the text only if the text states a relation. For example, the following sentence must not contain any symptom or disease labels:

> The woman, previously diagnosed with COVID-19, reported that her child showed fever symptoms.

Even though fever is a symptom of COVID-19, the meaning of the sentence does not imply this relation. Furthermore, it might be necessary to annotate at a paragraph level to accommodate for disease-symptom relation statements spanning multiple sentences. The current dataset is annotated at a sentence level.

The concept annotation module does not need to be modified, as it is already capable of annotating diseases. The relation extraction module, however, must be adapted to match diseases and symptoms together instead of separately from title and report body.

APPENDIX A

Appendix

A.1 Implementation

In this section we list major dependencies and implementation details of the model and the experiment. Table A.1 lists the software and data dependencies of the model. Table A.2 lists the hardware used to perform the experiments.

Name	Version
Python	3.8
Java	1.8
CUDA	10.2
MetaMap	2020
PyTorch	1.8
huggingface transformers	4.4
MetaMap Dataset	Base (2020AB)
MeSH thesaurus	2021

Table A.1: Most relevant software and data versions for the implementation of the relation extraction model

Component	Name
CPU	Intel i7 4790k
RAM	16GB
GPU	NVIDIA RTX 2060 Super (8GB VRAM)

Table A.2: Hardware used to perform the experiments

List of Figures

List of Tables

List of Algorithms

Bibliography

[ABV19] Alan Akbik, Tanja Bergmann, and Roland Vollgraf. Pooled Contextualized Embeddings for Named Entity Recognition. In *Proceedings of the 2019 Conference of the North*, pages 724–728, Minneapolis, Minnesota, 2019. Association for Computational Linguistics.

[Ake16] Katherine G. Akers. New journals for publishing medical case reports. *Journal of the Medical Library Association : JMLA*, 104(2):146–149, April 2016.

[AL10] Alan R Aronson and François-Michel Lang. An overview of MetaMap: Historical perspective and recent advances. *Journal of the American Medical Informatics Association*, 17(3):229–236, May 2010.

[AMB+19] Emily Alsentzer, John R. Murphy, Willie Boag, Wei-Hung Weng, Di Jin, Tristan Naumann, and Matthew B. A. McDermott. Publicly Available Clinical BERT Embeddings. *arXiv:1904.03323 [cs]*, June 2019.

[Aro01] A. R. Aronson. Effective mapping of biomedical text to the UMLS Metathesaurus: The MetaMap program. *Proceedings of the AMIA Symposium*, pages 17–21, 2001.

[BGJM17] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146, June 2017.

[BLC19] Iz Beltagy, Kyle Lo, and Arman Cohan. SciBERT: A Pretrained Language Model for Scientific Text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China, November 2019. Association for Computational Linguistics.

[BMR+20] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey

Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. *arXiv:2005.14165 [cs]*, July 2020.

[BMS+18] Bernd Bohnet, Ryan McDonald, Gonçalo Simões, Daniel Andor, Emily Pitler, and Joshua Maynez. Morphosyntactic Tagging with a Meta-BiLSTM Model over Context Sensitive Token Encodings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2642–2652, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[BZKA20] Nada Boudjellal, Huaping Zhang, Asif Khan, and Arshad Ahmad. Biomedical Relation Extraction Using Distant Supervision. https://www.hindawi.com/journals/sp/2020/8893749/, June 2020.

[CGCB14] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014.

[CZ19] Leyang Cui and Yue Zhang. Hierarchically-Refined Label Attention Network for Sequence Labeling. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4113–4126, 2019.

[DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[DLL14] Rezarta Islamaj Doğan, Robert Leaman, and Zhiyong Lu. NCBI disease corpus: A resource for disease name recognition and concept normalization. *Journal of Biomedical Informatics*, 47:1–10, February 2014.

[DS17] Rahul Dey and Fathi M. Salem. Gate-variants of Gated Recurrent Unit (GRU) neural networks. In *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 1597–1600, August 2017.

[Fle71] Joseph L Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378, 1971.

[GOMI18] Diana Galvan, Naoaki Okazaki, Koji Matsuda, and Kentaro Inui. Investigating the Challenges of Temporal Relation Extraction from Clinical Text. In *Proceedings of the Ninth International Workshop on Health Text Mining*

and Information Analysis, pages 55–64, Brussels, Belgium, October 2018. Association for Computational Linguistics.

[Hoc98] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

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

[HSMD13] María Herrero-Zazo, Isabel Segura-Bedmar, Paloma Martínez, and Thierry Declerck. The DDI corpus: An annotated corpus with pharmacological substances and drug–drug interactions. *Journal of Biomedical Informatics*, 46(5):914–920, October 2013.

[Hu20] Dichao Hu. An Introductory Survey on Attention Mechanisms in NLP Problems. In Yaxin Bi, Rahul Bhatia, and Supriya Kapoor, editors, *Intelligent Systems and Applications*, Advances in Intelligent Systems and Computing, pages 432–448, Cham, 2020. Springer International Publishing.

[JPS+16] Alistair E. W. Johnson, Tom J. Pollard, Lu Shen, Li-wei H. Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G. Mark. MIMIC-III, a freely accessible critical care database. *Scientific Data*, 3(1):160035, May 2016.

[KB17] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, January 2017.

[KOTT03] J.-D. Kim, T. Ohta, Y. Tateisi, and J. Tsujii. GENIA corpus–a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(Suppl 1):i180–i182, July 2003.

[LH18] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*, September 2018.

[Lip00] Carolyn E. Lipscomb. Medical Subject Headings (MeSH). *Bulletin of the Medical Library Association*, 88(3):265–266, July 2000.

[LSJ+16] Jiao Li, Yueping Sun, Robin J. Johnson, Daniela Sciaky, Chih-Hsuan Wei, Robert Leaman, Allan Peter Davis, Carolyn J. Mattingly, Thomas C. Wiegers, and Zhiyong Lu. BioCreative V CDR task corpus: A resource for chemical disease relation extraction. *Database*, 2016(baw068), January 2016.

[LWL+15] Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and Houfeng Wang. A Dependency-Based Neural Network for Relation Classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language*

Processing (Volume 2: Short Papers), pages 285–290, Beijing, China, July 2015. Association for Computational Linguistics.

[LYK+19] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. BioBERT: A pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, page btz682, September 2019.

[LZFJ17] Fei Li, Meishan Zhang, Guohong Fu, and Donghong Ji. A neural joint model for entity and relation extraction from biomedical text. *BMC Bioinformatics*, 18(1):198, March 2017.

[MB16] Makoto Miwa and Mohit Bansal. End-to-End Relation Extraction using LSTMs on Sequences and Tree Structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1116, Berlin, Germany, August 2016. Association for Computational Linguistics.

[MBSJ09] Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, Suntec, Singapore, August 2009. Association for Computational Linguistics.

[MDT+20] Khalil Mrini, Franck Dernoncourt, Quan Hung Tran, Trung Bui, Walter Chang, and Ndapa Nakashole. Rethinking Self-Attention: Towards Interpretability in Neural Parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 731–742, Online, November 2020. Association for Computational Linguistics.

[MSC+] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed Representations of Words and Phrases and their Compositionality. *NIPS 2013*, page 9.

[MSM93] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[NLØ19] Farhad Nooralahzadeh, Jan Tore Lønning, and Lilja Øvrelid. Reinforcement-based denoising of distantly supervised NER with partial annotation. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 225–233, Hong Kong, China, November 2019. Association for Computational Linguistics.

[NV19] Dat Quoc Nguyen and Karin Verspoor. From POS tagging to dependency parsing for biomedical event extraction. *BMC Bioinformatics*, 20(1):72, December 2019.

[PNI+18] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[PYL19] Yifan Peng, Shankai Yan, and Zhiyong Lu. Transfer learning in biomedical natural language processing: An evaluation of BERT and ELMo on ten benchmarking datasets. *BioNLP 2019*, page 58, 2019.

[SHTS93] P L Schuyler, W T Hole, M S Tuttle, and D D Sherertz. The UMLS Metathesaurus: Representing different views of biomedical concepts. *Bulletin of the Medical Library Association*, 81(2):217–222, April 1993.

[SLK+18] Setu Shah, Xiao Luo, Saravanan Kanakasabai, Ricardo Tuason, and Gregory Klopper. Neural networks for mining the associations between diseases and symptoms in clinical notes. *Health Information Science and Systems*, 7(1):1, November 2018.

[SSW15] Shiliang Sun, Honglei Shi, and Yuanbin Wu. A survey of multi-source domain adaptation. *Information Fusion*, 24:84–92, 2015.

[TK19] Tung Tran and Ramakanth Kavuluru. Distant Supervision for Treatment Relation Extraction by Leveraging MeSH Subheadings. *Artificial intelligence in medicine*, 98:18–26, July 2019.

[VF05] Michel Verleysen and Damien François. The Curse of Dimensionality in Data Mining and Time Series Prediction. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Dough Tygar, Moshe Y. Vardi, Gerhard Weikum, Joan Cabestany, Alberto Prieto, and Francisco Sandoval, editors, *Computational Intelligence and Bioinspired Systems*, volume 3512, pages 758–770. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all

you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 6000–6010, Red Hook, NY, USA, December 2017. Curran Associates Inc.

[WKL13] Chih-Hsuan Wei, Hung-Yu Kao, and Zhiyong Lu. PubTator: A web-based text mining tool for assisting biocuration. *Nucleic Acids Research*, 41(W1):W518–W522, July 2013.

[WLA+18] Yanshan Wang, Sijia Liu, Naveed Afzal, Majid Rastegar-Mojarad, Liwei Wang, Feichen Shen, Paul Kingsbury, and Hongfang Liu. A comparison of word embeddings for the biomedical natural language processing. *Journal of Biomedical Informatics*, 87:12–20, November 2018.

[YDY+19] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. XLNet: Generalized autoregressive pretraining for language understanding. *Advances in Neural Information Processing Systems*, 32:5753–5763, 2019.

[ZCY+19] Yijia Zhang, Qingyu Chen, Zhihao Yang, Hongfei Lin, and Zhiyong Lu. BioWordVec, improving biomedical word embeddings with subword information and MeSH. *Scientific Data*, 6(1):52, December 2019.

[ZHRH20] Markus Zlabinger, Sebastian Hofstätter, Navid Rekabsaz, and Allan Hanbury. DSR: A Collection for the Evaluation of Graded Disease-Symptom Relations. In Joemon M. Jose, Emine Yilmaz, João Magalhães, Pablo Castells, Nicola Ferro, Mário J. Silva, and Flávio Martins, editors, *Advances in Information Retrieval*, volume 12036, pages 433–440. Springer International Publishing, Cham, 2020.

[ZLL+17] Wei Zheng, Hongfei Lin, Ling Luo, Zhehuan Zhao, Zhengguang Li, Yijia Zhang, Zhihao Yang, and Jian Wang. An attention-based effective neural model for drug-drug interactions extraction. *BMC Bioinformatics*, 18(1):445, October 2017.

[ZLY+18] Yijia Zhang, Hongfei Lin, Zhihao Yang, Jian Wang, Shaowu Zhang, Yuanyuan Sun, and Liang Yang. A hybrid model based on neural networks for biomedical relation extraction. *Journal of Biomedical Informatics*, 81:83–92, May 2018.

[ZMBS14] XueZhong Zhou, Jörg Menche, Albert-László Barabási, and Amitabh Sharma. Human symptoms–disease network. *Nature Communications*, 5(1):4212, June 2014.

[ZZL+18] Yijia Zhang, Wei Zheng, Hongfei Lin, Jian Wang, Zhihao Yang, and Michel Dumontier. Drug–drug interaction extraction via hierarchical RNNs on sequence and shortest dependency paths. *Bioinformatics*, 34(5):828–835, March 2018.