Experiments with GloVe embeddings and Domain Adversarial Neural Networks for the Dutch Medical Domain

Master's Thesis

Student: H.W. Lokhorst
Student number: 4236319

First examiner: T. Deoskar
Second examiner: M. Fowlie
Daily supervisor: W. B. Veldhuis

Universiteit Utrecht

Artificial Intelligence
Utrecht University
Netherlands
November 5, 2021

Abstract

There is a lot of interest in developing automatic natural language processing tools in the Dutch language medical domain. Such tools would have great impact on automating hospital documentation procedures, as well as other benefits. However, most current NLP models and datasets exist only for English, and more specifically only the general domain of English. Domain-specific models are hard to come by even for English, let alone other languages which do not have the same amount of resources. The focus in this thesis is on developing models and resources that will be useful for the Dutch medical domain. This domain lacks annotated data and domain-specific models. In the fist part of the thesis, GloVe embeddings (Pennington et al., 2014) are developed. However, evaluating the quality of these embeddings is a challenge, given the lack of annotated resources for medical Dutch. The second part of the thesis presents experiments using a novel domain adaptation method, Domain Adversarial Neural Networks, which is getting attention for domain-adaptation problems in NLP. The network is trained on a Named Entity Recognition task and a Part-of-Speech tagging task, with and without (English) medical embeddings. Its performance and suitability for various domain-adaptation scenarios is evaluated.

Contents

Chapter 1

Introduction

Recent advances in natural language processing (NLP), for example via neural network models, have lead to a lot of interest in using language processing models for automatic processing of medical texts. Accurate and reliable language processing can have a great impact on many areas in the medical domain. Not only can automatic processing of documents enable medical specialists to spend less time on manual documentation for instance, it can also improve communication between divisions. This benefits physicians, patients and ultimately society as a whole.

The broad focus in this thesis is the development of natural language models and tools for the Dutch medical domain. This is done by investigating automated processing of radiology reports of the University Medical Center (UMC) in Utrecht. Automated language processing has been identified to have impact on various processes and workings in a medical institute, or radiology department. Examples of the most common research topics are:

1. Diagnostic Surveillance: scanning reports when new information on an illness is inserted in the system and extracting reports that might need to be reviewed based on this information.

2. Cohort Building: Patients are usually only scanned for a suspected problem. Reports and images can contain a lot of extra information. Storing the patient information together with certain observation could give insights for other studies. For example, one might get insights in 'large livers' from patients that had a belly scan.

3. Query-based Case Retrieval: Retrieve cases with non-defined conditions or outcomes, but specified by concepts and keywords. This usually requires a lexicon or ontology.

4. Quality assessment of radiology practice: Some procedures might show less important clinical findings; by comparing findings with subsequent action, procedures can be improved.

5. Clinical Support Services: aimed to improve the workflow of radiologists. For example a system suggesting the appropriate antibiotics based on the report findings.

The starting point for the Dutch medical domain is chosen to be word embeddings. Word embeddings are vector representations of words. Pretrained embeddings have not only shown to increase model performance, but can also significantly reduce training time for many NLP tasks. Another reason to make a start on this domain with word embeddings, is the difficulties that the domain poses.

First of all, while the English medical domain has some well-developed tools and models, there are no trained Dutch medical tools or models. Secondly, the medical data are noisy, as the reports are not manually transcribed. And finally, the Dutch medical domain is understudied: there are little to no NLP papers on the Dutch medical domain. The few papers that are available describe rule-based or hybrid models, with one specific application. Creating word embeddings will shed more light on those difficulties, what needs to be done to overcome them and it has value for future models.

1.1 Contributions of thesis

- Discuss main challenges in the field of Dutch medical NLP and the relevant approaches, mostly dependent on available (annotated) resources.
- Develop English medical embeddings based on radiology data and compare their performance with general embeddings on different datasets.
- Build a Domain Adversarial Neural Network (DANN) to be used as a downstream task architecture for word embeddings. The model can be deployed for any sequential task with existing data.
- Evaluate the English medical embeddings within the DANN.
- Suggests requirements for Dutch embeddings to be developed.
- Add useful resources and data in Appendix and Data section, as no integrated hub exists as of yet for medical NLP.

1.2 Structure of thesis

After this introduction, the next chapter describes the general challenges for Dutch medical NLP and discusses the approaches that can be taken. Chapter 3 describes the various datasets that are used in the thesis; both medical and non-medical. It also includes descriptions of relevant resources and datasets that were explored, but not ultimately used in the thesis. I hope that this inventory will serve as a useful resource for future work in this area. Finally, chapter 3 discusses preprocessing steps. Chapter 4 discusses the development of word-embeddings (GloVe) for the medical domain, and presents experiments to show the importance of different parameters and hardware requirements. Chapter 5 presents experiments on domain-adaptation using a Domain Adversarial Neural Network (DANN), after which the performance with and without word embeddings is discussed. The important factor here is that little to no annotated medical training data are required to train the DANN. Finally, Chapter 6 is the discussion, where results are interpreted, shortcomings are mentioned and suggestions for future research are given.

1.3 Restrictions during thesis

The idea for this thesis and its experiments were subject to restrictions due to the Covid lockdown and regulations. Due to the sensitive nature of medical data, the Dutch medical (UMC) data could

only be accessed from within the hospital. During the Covid period, access to the hospital was at best uncertain and mostly discouraged. Therefore, some experiments were modified to tackle general domain-adaptation problems, using online available English medical data, rather than Dutch data.

Chapter 2

Dutch medical NLP

2.1 General challenges for Dutch medical NLP

This section describes the general challenges for Dutch medical NLP. Tackling those challenges, in descending order of importance, should give Dutch medical NLP a boost. Next to this, the third and fourth problem only exist if one wants to train rule-based or hybrid models. Tackling the first two problems could overcome the need for rule-based and hybrid models altogether.

2.1.1 Lack of annotated training data

Even though many models are available, most current models are exclusively trained on English data. Where English data are widely available, domain-specific, non-English data are much scarcer. The problem becomes clearer when models become more complex and require more data. One example is GPT-3 (Brown et al., 2020), which used most 'clean' data that are available on the entire web (by using a web crawler). 92% of this data is English, giving an impression of the imbalance of English vs. non-English data. Apart from the data available to unsupervised models like GPT-3, English research also has an advantage when it comes to supervised models. Since there are a lot of annotated data available, training any kind of supervised NLP model is not as resource-consuming. When comparing this to the Dutch domain, the available resources are a lot scarcer.

2.1.2 Lack of (domain-specific) language models in Dutch

The second problem is the lack of domain-specific models. For English, many models are available in multiple different frameworks. The Huggingface Transformer framework (Wolf et al., 2019) offers a multitude of English Transformer models and variations of Bidirectional Encoder Representations from Transformer (BERT) (Devlin et al., 2018) models. The Flair framework (Akbik et al., 2019) has recently also moved their models to the Huggingface model hub. Flair offers a multitude of datasets with special biomedical support. In those frameworks, only one Dutch language model is present. This is the Dutch BERT model, BERTje (de Vries et al., 2019). This model was only trained on general Dutch data, resulting in a poor performance on biomedical data. In order to use a model like BERTje on the medical domain, the model would have to be trained on medical Dutch data. This approach

has the downside that it is not scalable: the same training would need to be done for any other language and for any other domain. Moreover, a language model only performs well when provided with enough training data. This is a tough requirement for many languages and domains: it requires resources. Even if such language model could be trained, the fine-tuning step would also require some annotated domain-specific data. Compared to language models, task-specific models would require less domain-specific, unannotated data, but more task-specific, annotated data. Annotated, English data are widely available, but for Dutch only a few datasets are available, let alone for Dutch medical data. It would require (a lot of) human effort to obtain the annotated data. This leads to the notion of focusing research on models that do not require much language-specific data.

2.1.3 Lack of ontologies

The third problem is the lack of ontologies. Ontologies are mappings from words to concepts. So, each medical term is mapped to one single concept. One example of such ontology is SNOMED-CT. Several English papers use this ontology in their medical NLP pipeline. A similar medical NLP pipeline for the Dutch medical domain has been theoretically proposed by (Cornet et al., 2012). The pipeline is based on the storage of medical concepts in a terminology system with a certain level of confidence. When Named Entities (NE) are recognized to a sufficient extent, the entities can be mapped to a concept. This requires either a Dutch ontology, or a translation tool after which an English ontology can be used. A thesis (Westerbeek, 2015) implemented the NER part to the extent of 60-70% accuracy. However, a translation tool for Dutch medical data has yet to be created and a Dutch ontology is also non-existent.

A more recent study (Nobel et al., 2020b) developed a rule-based algorithm to classify T-stage of pulmonary oncology from Dutch, free-text, radiology reports. This paper also mentions the problematic extraction score of Named Entities. To overcome this problem, they manually created synonym sets (comparable to ontology) and created one regular expression per set. Furthermore, this paper uses a structured report as input and recommends even tighter structuring of the report. For example, discussing only one concept per sentence would significantly reduce misinterpretation by models. Either way, an ontology or synonym set is used in multiple studies, but is not (freely) available for the Dutch medical domain.

2.1.4 Unstructured texts or reports

The final challenge is related to preprocessing. To understand the problem, an important concept in medical NLP should be discussed: structured reports. Structured reports are required – or at the least very useful – for most automated clinical decision support systems (Cornet et al., 2012), (Pons et al., 2016), (Nobel et al., 2020b)). To disambiguate standardized reports from structured reports, I would like to refer to Nobel, Kok, & Robben (Nobel et al., 2020a), who describe standardized reports as "aimed at improving the accuracy of the medical content of a radiological report", whereas a structured report is "aimed at reducing variability and enhancing the clinical utility of formal radiological interpretations". Based on those aims, they define structured reporting as "the use of an IT-based means of importing and arranging medical content in the radiological report". So, in other words, standardized reporting is about content, whereas structured reporting only modifies report structure. This modification of structure facilitates easier extraction of (named) entities, for both

rule-based and machine learning applications. Pons et al. (Pons et al., 2016) discuss in their review of NLP in (English) radiology, that almost all systems identify report sections and use segmentation steps; apart from also doing basic preprocessing and creating a domain-specific lexicon. The UMC reports are not structured and only partly standardized.

2.2 Approaches

Most problems described in the previous section can be solved by either (quality) data or good models. However, given the challenges, starting on this domain by developing a model does not seem reasonable. Approaches that boost future research on the Dutch medical domain and contribute to a better understanding of the domain seem most feasible. I start by describing the least feasible option, working towards the more interesting approaches. This chapter will end with a motivation for the chosen approach.

2.2.1 Automatic structuring of unstructured reports

Multiple options are available to create structured reports, also for non-English languages ((Pathak et al., 2019), (Nobel et al., 2020b)). By far the largest part of them are rule-based or supervised machine learning methods. Since this once again poses the problem of annotated data or creating a very task-specific architecture, an unsupervised structuring model would be preferred. Without domain-specific annotated datasets – which is the case for most non-English languages – one either has to annotate data oneself or train an unsupervised model.

Another point, closely related to structuring reports, is standardization of the data; see also 2.1.4. Standardization would provide a standard between hospitals, allowing for using resources together to create State-of-the-Art (SotA) medical models. Finally, standardization could improve model performance and would also make translation approaches more viable.

2.2.2 Translation of Dutch medical text into English

The second approach would be to translate named entities into English; which would allow the subsequent use of English models and tools, like ontologies. Previous research already tried doing this back in 2015 (Westerbeek, 2015). As described in section 2.1.3, this approach does not seem feasible: apart from the task loss, the translation loss also becomes an issue. Another reason not to use this approach, is that it does not encourage research to be independent of English. Research independent from the English domain could give more insights in different or universal language patterns and models. Furthermore, one could question to what extent English research makes use of existing tools because they exist, rather than actually needing those tools. In other words: is the availability of resources in English slowing down the advancement of NLP models? Focusing research on models that require less data could ultimately lead to more efficient models. Those models could initially be developed with English data, since the resource are in the end mostly available in English. However, by showing a decreased need for annotated data or model training requirements, the models become more appealing for AI world-wide. This research direction could therefore be a starting point for non-English domain-specific studies, even though it starts by utilizing English data.

2.2.3 Using unsupervised models

The third option is the implementation of an unsupervised model. This avoids the issue of requiring annotated data and would not only advance Dutch AI research, but could also help other languages in discovering new approaches. The progress of unsupervised models has taken a flight on English domains, with models like BERT and GPT-3. However, most NLP research on the medical domain uses language models combined with an ontology to map each named entity to a concept. As explained in the previous section, ontologies pose problems for non-English domains. Apart from the ontology that would have to be used, training a language model is resource-consuming. Therefore, this approach is not feasible for this thesis, even though it would be a State-of-the-Art approach. One other unsupervised model class is consists of models that are used to develop word embeddings. As this approach only uses free text data as input, to develop a useful tool, it will be taken in this thesis. However, as will be described in the next chapter, evaluation is not straightforward for such methods. Therefore, another model is required to evaluate the developed word embeddings.

2.2.4 Domain Adaptation

The final option, which is also taken in this thesis, is domain adaptation (DA). A default assumption in many machine learning algorithms is that the training and test data follow the same underlying distribution. If those do not match, one could speak of a domain shift. Due to this shift, the performance on the test set drops, confirming that a model is less able to generalize in case of a domain shift. Thus, domain shifts are related to a big issue in machine learning: generalization beyond the training distribution. Ideally, models would be robust to unknown domains. Most DA models have access to a small amount of target domain data. In this case, a supervised method can be used. One such supervised method is cross-domain adaptation. In this method, a domain-specific language model is trained, which is then fine-tuned with annotated data on a task-specific dataset (in a different domain). If applied to this thesis: training a Dutch medical language model and fine-tuning it on the Dutch CoNNL-2002 general dataset for Part-of-Speech (POS) tagging. This overcomes the problem of annotated data for a specific application, but still requires a domain-specific language model (Rietzler et al., 2019). Even though supervised models have a high performance, they are, as mentioned previously, too resource-heavy. So, this approach cannot be used in this thesis either. The final option is a combination between unsupervised models and domain adaptation: unsupervised DA (UDA) models. UDA models resemble the real-world better, as usually labeled data are absent, whereas unlabeled data are abundant (Ramponi and Plank, 2020). One such UDA model is the Domain Adversarial Neural Network (DANN), which will be used in this thesis.

2.3 Motivation of taken approach

I finally chose one unsupervised approach that can contribute a lot to many future medical NLP applications: word embeddings. Each NLP model makes use of word embeddings and good word embeddings usually lead to significant model improvements. The model I will use to develop the word embeddings is the GloVe model; this will be further explained in Chapter 4. Next to developing those embeddings, or at least investigating its possibilities, I will replicate a domain adaptation model: a DANN. This can be used as downstream task to evaluate the performance of the word embeddings.

I deem this approach the most feasible from the aforementioned four options. Those options can be divided into two categories. The first category includes annotating data oneself or having annotated data available. Subsequently, either a language model can be used, or a task-specific architecture can be developed. This first approach is therefore quite resource-heavy and tends to contribute less to the advance of AI in general. One of the least resource-consuming methods in this category is transfer learning. In transfer learning, an existing model is fine-tuned on new data. BERT models are considered one of the best when it comes to transfer learning. However, a BERT model does require a certain amount of annotated data to be fine-tuned on a task. Furthermore, if too many tokens in the data are out-of-vocabulary the BERT performance suffers.

The second category of approaches focuses on finding or developing a model that requires little to none annotated domain-specific data. This leads to the use of unsupervised models; or, in combination with domain adaptation, to unsupervised domain adaptation. Since the available Dutch (medical) annotated data are limited, the second category of approaches seems most feasible. Therefore, the approach of developing word embeddings as a starting point for the Dutch medical domain is chosen. By using a Domain Adversarial Neural Network as evaluation method, the path of domain adaptation is also explored. The DANN does not require any annotated medical data to train on. It only requires annotated test data: to evaluate the performance. As this can be a limited amount of data, this approach is one of the least resource-heavy approaches.

Chapter 3

Relevant datasets

This chapter describes the various English and Dutch datasets, both for the medical and non-medical domains, that were used for the domain-adaptation experiments. Additionally, some other relevant datasets that were explored in the initial phase of the thesis - but were not ultimately used - are also discussed here. This might be useful for future studies in this area. The second part of this chapter describes the preprocessing steps that were taken, before the data was used. This section also describes which dataset was used in which specific experiment.

3.1 Medical datasets

- GENIA (Kim et al., 2003) (Tsuruoka et al., 2005) (English): The GENIA corpus was developed in 2003. The corpus contains 1,999 Medline abstracts, selected using a PubMed query for the three MeSH terms "human", "blood cells", and "transcription factors". The abstracts contain 436,000 words of which about 96,000 are annotated for biological terms. The annotation has been done by two domain experts. The used ontology to annotate the abstracts is extensive. The top-level classifies into 'substance', 'source' and 'other'. Substance and source are both split into two underlying categories, and those kinds of splits finally result into 47 specific categories.

  Subsequently, in 2005, Tsuruoka annotated a part of the GENIA corpus for POS tag, while developing his biomedical POS tagger.

- MIMIC-CXR (Johnson et al., 2019): The MIMIC-CXR database is a large, publicly available, English dataset of chest radiographs. The data is stored in DICOM format, accompanied with their corresponding radiology reports. The dataset contains about $375,000$ images corresponding to $227,000$ radiographic studies. The studied are all performed at the Beth Israel Deaconess Medical Center in Boston, Massachussetts. Finally, all data is de-identified to satisfy the HIPAA Safe Harbor requirements and Protected Health Information (PHI) has been removed. Wherever PHI was removed, three underscores were inserted. The data in this database is not annotated for a task. Only the radiology reports were used for this thesis. It is useful to note that, despite the fact that the reports are structured, the data is still messy. This is because this data is publicly available and sensitive data is omitted, occasionally resulting in weird sentences.

- UMC data (Dutch): For this thesis, I got access to about 10,000 Dutch radiology reports from the University Medical Centre Utrecht. In consultation it was decided to restrict the reports to the topic of pulmonary embolisms. More data is available, but due to the data being highly confidential, the access is (initially) limited. The data is not annotated for a task.

3.2 Non-medical datasets

- CoNNL-2002 (Tjong Kim Sang, 2002): A Dutch dataset, annotated for four types of Named Entities: persons, locations, organizations and names of miscellaneous entities. The Dutch data consists of four editions of a Belgian newspaper of the year 2000. Annotation was done by the university of Antwerp, where annotators followed the MITRE and SAIC guidelines for NER (developed by Chinchor, 1999).

- CoNNL-2003 (Tjong Kim Sang and De Meulder, 2003): An English, publicly available dataset, developed for Named Entity Recognition. Specifically to recognize persons, locations, organizations and names of miscellaneous entities in free text. Apart from the NER labels, the data is also labelled with POS-tags and syntactic chunk tags. The data is a collection of newspaper articles from the Reuters corpus. It has been annotated by researchers from the university of Antwerp.

- MIT Movie: this corpus is a semantically tagged training and test corpus in BIO format. The corpus contains a part of simple queries and a part of more complex queries. From: https://groups.csail.mit.edu/sls/downloads/ .

- Twitter (Derczynski et al., 2016): An English Twitter dataset. It is annotated for NER.

- Westbury Lab Corpus (Shaoul, 2010): A corpus created from a snapshot of English wikipedia pages in 2010. It was downloaded using Wikipedia Dump and subsequently cleaned with the WikiExtractor tool. Finally, articles longer than 2000 words were omitted. The corpus contains about 1B words in approximately 2M documents.

- WNUT 2020 (Kulkarni et al., 2018) (Tabassum et al., 2020): "WNUT 2020 shared task is designed on the wet lab protocol data. Wet Lab protocols are basically a collection of steps from different lab procedures. They are noisy, dense, and domain-specific." The corpus consists of approximately 14,000 sentences and is annotated for the main classes Entities, Events and Relations. The total of the subclasses adds up to 76 classes, including an <UNK> token. Since this dataset is important for the GloVe embeddings, some examples will be included here:
  - "How to Make a 0.5M TCEP Stock Solution"
  - "Weigh 5.73 g of TCEP."
  - "Add 35 ml of cold molecular biology grade water to the vial, and dissolve the TCEP."
  - "This resulting solution is very acidic, with an approximate pH of 2.5."

3.3 Useful Datasets

- EMC Dutch Corpus (Afzal et al., 2014): a Dutch medical corpus consisting of different types of anonymized clinical documents. It contains entries from general practitioners (2000), specialist letters (2000), radiology reports (1500) and discharge letters (2000). The corpus is annotated for negation, temporality and experiencer properties. As the numbers between brackets indicate, this corpus consists of a total of 7500 documents. The data is not freely available: the Erasmus Medical Centre has to be contacted for access to the data.

- Mantra-GSC (Kors et al., 2015): this corpus was developed to create a multilingual gold-standard corpus for biomedical concept recognition. Different corpora were used to select English, French, German, Spanish and Dutch text units. The Dutch part consists of 100 MEDLINE abstracts, with a total of 922 words and 100 drug labels with a total of 2055 words. The text is annotated following a subset of the UMLS: MeSH, SNOMED-CT and MedDRA. This results in a total of 591,000 concepts, for 3,2M terms (over all five languages. The annotators were presented with automatically generated annotations. Usually about 2-3 relevant tags were presented, where the annotators had to disambiguate. In case disambiguation could not be resolved, both tags were kept.

- MedInfo-2013 (Chapman et al., 2013): multilingual lexicon, containing English, French, German and Swedish clinical free text. The lexicon is annotated for negation, whether a condition is historical, experienced by a family member and whether a concept is mentioned in a general, conditional or indicative context. Those annotations are done by extensions of NegEx in the ConText algorithm.

- NCBI (Doğan et al., 2014): the NCBI corpus is English, fully annotated at the mention and concept level to serve as a research resource for biomedical NLP. It consists of 793 PubMed abstracts, about 7,000 disease mentions, 790 unique disease concepts. Furthermore, the corpus is already divided into a training, validation and test set. Fourteen annotators worked on the creation of this corpus, with two annotators per article.

3.4 Preprocessing

General preprocessing

Preprocessing is required to improve the learning of machine learning models. Where humans understand that certain ambiguous character combinations, for example '.hello', 'Hello', 'HEllO', and 'h.el.lo', have the same meaning, artificial intelligent models do not understand this (without training). Therefore, all the data are transformed to avoid such ambiguities as much as possible. Resolving such ambiguities usually reduces the training time and improves model performance. The preprocessing steps mostly depend on the noisy datasets, as those contain most confusing symbols and words. The cleaner datasets will have to undergo the same preprocessing steps, to make sure all data are treated the same way. As mentioned in Chapter 3, the Dutch medical data are noisy. The data consist of sentences with random punctuation, and random capital letters. The punctuation was changed by making sure that each dot was followed by a space. After splitting sentences this way,

the punctuation, except for comma's, was entirely removed. Next to this, all words were lower-cased. The other used noisy dataset was the Twitter dataset. This data contains hyperlinks, symbols and non-Latin characters. In some cases those features could provide additional information, but since this is not the case in this thesis, I chose to remove those features. To summarize, all words were lower-cased, all punctuation except for comma's was removed and all symbols and non-Latin characters were omitted.

Mistakes

Some of the used datasets contain mistakes. Only mistakes resulting in file inconsistencies, which hamper the reading of the data file, were fixed. For example, newlines or spaces missing, or a missing letter of a tag. There have been no changes to the content of any of the datasets: if a tag seemed grammatically incorrect, this was not changed.

Tag sets

In the domain adaptation model, usually two different datasets are used. After all, one wants to adapt from one domain to the other. This does usually mean that the tag sets are not equal. In each case, I mapped the tags to the set of tags; i.e. tags that were present in both datasets. In some cases, I even removed a word-tag combination. This is when the word-tag type did not seem relevant for the task. Anything not mentioned in this section was kept as in the original dataset.

The CoNNL-2003 and Twitter tag sets were slightly changed. The CoNNL data are annotated with the inside-outside-beginning (IOB) tag set (Ramshaw and Marcus, 1999), whereas the Twitter data are annotated using the BIOES tag set. The BIOES tags have the 'End' and 'Single' annotation included, respectively marking ends of entities and one-word entities. Therefore, the Twitter data were adjusted by converting S-tags to B-tags and E-tags to I-tags.

The Part-of-Speech (POS) tags of the CoNNL-2003 dataset and the GENIA dataset were also slightly modified to make sure both datasets are annotated for exactly the same tags. The GENIA data lack the [$, NNPS and UH] tags. Therefore, the 'UH' tagged words were removed and the other tags were respectively remapped to 'SYM' and 'NNP'.

3.5 Overview of datasets used per experiment

Table 3.1 shows the datasets that were used in GloVe experiments; which are described in the next chapter. The data used for each domain adaptation experiment can be seen in table 3.2

	Exp. 1	Exp. 2	Exp. 3
Embedding dataset	MIMIC-CXR	MIMIC_CXR Stanford GloVe	MIMIC-CXR Stanford GloVe Westbury
Training dataset	WNUT-2020	MIT Movie	MIT Movie WNUT-2020
Test dataset	WNUT-2020	MIT Movie	MIT Movie WNUT-2020

Table 3.1: Datasets used per GloVe (part 1) experiment

	Exp. 0	Exp. 1	Exp. 2	Exp. 3*
Embedding dataset	-	Stanford GloVe	MIMIC-CXR Stanford GloVe Westbury	UMC Dutch Fasttext Dutch Wikipedia
Training dataset	CoNNL-2003	CoNNL-2003	CoNNL-2003	CoNNL-2002
Test dataset	CoNNL-2003	Broad Twitter Corpus	GENIA	UMC

Table 3.2: Datasets used per DANN (part 2) experiment. *See section 6.1.2.

Chapter 4

GloVe Embeddings

This chapter first explains the theory and model that are used to train word embeddings. Then the embeddings, created for the English medical domain, are described in detail. No Dutch embeddings are developed in this chapter, as there is no way of validating those (yet). One other point of clarification is that when I talk about 'word embeddings', I mean 'pretrained word embeddings'. Those are word embeddings learned in task A, to be used in task B. The other option, training the word embeddings during task B in an embedding layer, will be discussed in the next chapter.

4.1 Word embeddings

In order for machine learning models to 'read' natural language, the text is broken down into chunks or tokens. Those tokens subsequently need to be converted into (numerical) representations. One such representation is word embeddings: distributed vector representations of text. Word embeddings provide multiple benefits and advantages for NLP models. Word embeddings are used in many NLP tasks, improving the State-of-the-Art models' results (Wang et al., 2019). The two main reasons to use word embeddings are sparsity of training data and a large number of trainable parameters. Most datasets contain a huge number ($> 70\%$) of 'rare' words; as described by Zipf's Law. This means that estimating the vectors of those words during a task is quite hard, if not impossible. On top of that, learning the embeddings in the embedding layer makes for more trainable parameters. More trainable parameters results in a longer training duration. In summary, using word embeddings results in better model performance and faster model training. Multiple models have been developed over time to assign words to (vector) representations. In this thesis, I arbitrarily chose to use the GloVe model (Pennington et al., 2014).

4.2 GloVe

The main classes of methods that GloVe (Pennington et al., 2014) builds on, are local window-based methods (Mikolov et al., 2013) and global matrix factorization methods (Deerwester et al., 1990). Both individual methods were subject to some disadvantages. The main disadvantage of the local window-based methods is that they do not directly operate on the co-occurrence statistics of the corpus. This

prevents the models from taking advantage of repetitions and long-range dependencies in the data. Next to this, the disadvantage of the global matrix factorization methods, is that they does not do well on preserving linear regularities among words. GloVe combines both methods in a new theoretical framework. The authors start by analyzing the model properties, required for learning vector space representations, that capture fine-grained semantic and syntactic regularities. The authors argue that global log-bilinear regression models can overcome the aforementioned problems and produce linear directions of meaning among words. Global log-bilinear models use a weighted least squares model that trains on global word-word co-occurrences and therefore makes use of global statistics. They show that this is possible by proposing GloVe, a new weighted least squares regression model. In Glove, a word-word co-occurrence matrix is established. From this matrix, the probability can be derived, with which a word occurs in the context of another word. Those probabilities are subsequently used in a newly introduced weighted least squares regression model. This model calculates the similarity of two chosen words, when compared with a random third word. For example, if 'boy' and 'girl' are compared to 'car', 'boy' will probably co-occur more frequently with 'car'. So, its co-occurrence probability will be higher than that of 'girl' with 'car'. The model will therefore attempt to align the vector space in such a way - by updating the vector weights - that 'boy' is closer to 'car', than 'girl' is to 'car'. This is done for all words in the vocabulary. So, based on the relative co-occurrence probability of two words, compared to a random word, the vector space is estimated. In an ideal setting, the result of this fit is 0: the vector alignment corresponds perfectly to the (relative) co-occurrences in the data. This can be utilized by a cost function as objective, to estimate the optimal word vector space. So, based on a co-occurrence matrix, the GloVe model probes word vectors with respect to one another, to derive the optimal word vector space. The model is explained in more detail and with more examples in the original paper (Pennington et al., 2014).

The computational complexity in this (GloVe) model depends on the number of nonzero elements in the probability matrix. However, in this thesis I add a small constant to all matrix entries to avoid nonzero elements. This is because multiplication is used to derive the probability and one nonzero element in the co-occurrence matrix would result in a product (= probability) of zero. The original paper discards the non-zero entries for this same reason. This is not always useful, since the discarded co-occurrence might still occur in future data. This is especially the case in the small datasets that I will be using. Therefore, in my GloVe model, the size of the vocabulary (thus probability matrix) determines the computational complexity.

4.3 Word embedding evaluation methods

To evaluate the word embeddings, two main classes of evaluation methods exist: intrinsic and extrinsic methods. Intrinsic methods assess the (linear) relationships between words. Examples are the word analogy task, for example the Google dataset (Mikolov et al., 2013), and the word similarity task, for example SimVerb-3500 (Gerz et al., 2016). Apart from those well-known tasks, there is also concept categorization, outlier detection and the QVEC toolkit (Wang et al., 2019). Even though the general domain has quite a lot of such relationships, the question is whether this also holds for the medical domain. The similarity between 'Paris' and 'Berlin' is obvious; is the similarity between two diseases as obvious? What similarities does one expect models to capture in medical data? Apart from this question, no medical datasets exist to measure those similarities; especially when it comes to the

Dutch domain. Therefore, the second method, extrinsic evaluation, seems the more obvious approach. Extrinsic evaluation is done by using the pretrained word embeddings in a downstream task. Examples are Named Entity Recognition, Part-of-Speech tagging, negation detection and trigger event detection. This means that data are required, which in turn has to suffice several requirements. First of all, to ensure that one captures medical information in the word embeddings, the required data should also be medical data. Furthermore, in the future the task has to be performed in other languages, like the Dutch language. This means that a downstream task is required, for which annotated data on the English and Dutch medical domain is available. At the time of writing, no Dutch annotated medical dataset is available, which means the embeddings cannot be evaluated on a simple downstream task. To summarize, we are looking for a model that uses or could use word embeddings, while not requiring (much) annotated medical data. The solution for this lies in domain adaptation; as mentioned before, the approach in this thesis. The idea is to train a model on a domain where labels are available, and subsequently test the model on as little annotated English and Dutch medical data as possible. The next chapter (5) will describe the downstream task idea in detail.

4.4 GloVe experiments

Before the experiments, the types of embeddings that are experimented with, are defined as follows: medical and general embeddings result from training the GloVe model exclusively on a dataset from the respective domain. Mixed embeddings are the result of training the GloVe model on a dataset comprised of both medical and general data. The details of the embedding types are discussed later in this chapter. The following questions will be answered in the respective three experiments:

1. How do medical embeddings perform on a biomedical dataset?
2. How do medical embeddings generalise to general English? What is the effect of word embedding dimension?
3. Do mixed embeddings have an advantage compared to the medical embeddings?

4.4.1 Training GloVe embeddings

First, the GloVe model is trained on 15M tokens of English radiology data; the MIMIC-CXR data. After training the embeddings, the embeddings are evaluated in a downstream task in the Flair framework, using an LSTM-CRF. To get an impression of the performance of medical embeddings on biomedical data, the WNUT-2020 dataset is chosen. This is an English biomedical Named Entity dataset with lab protocols. The parameters of the LSTM-CRF are fixed: 256 hidden nodes, a batch size of 32, learning rate of 0.1 and a patience of 0.3. Next to that, for computation efficiency, the vocabulary size was set to 5000. The F1 scores on this dataset, as reported in the original WNUT paper (Tabassum et al., 2020), varied from 0.52 to 0.78. The results for this first experiment are shown in table 4.1. The results are slightly better than the worst results initially reported on the (WNUT) task; 0.55 vs. 0.52 respectively.

As performance clearly suffers when words with a count of 100.000+ are excluded, in following experiments a maximum word count of 1.000.000 is used. The same reasoning is used to select a window size of 10 and a minimum count of 5.

	Training F1	Test F1	Min. count	Max. count	Window size	Training time
1	0.61	0.54	5	1M	10	31:04
2	0.61	0.55	5	1M	5	33:23
3	0.60	0.54	10	1M	10	33:12
4	0.61	0.53	10	1M	5	29:36
5	0.52	0.46	5	100k	10	32:16
6	0.52	0.45	5	100k	5	31:06
7	0.52	0.46	10	100k	10	33:08
8	0.52	0.45	10	100k	5	33:16

Table 4.1: Tuning of hyperparameters on complex Named Entity dataset WNUT-2020.

4.4.2 Performance of medical embeddings on general dataset

The next question is to what extent the (same) medical embeddings capture general English language. This is tested on a dataset that tests both simple and complex general domain Named Entity Recognition (NER): the MIT Movie dataset. Next to testing the medical embeddings, the State-of-the-Art (SotA) general embeddings (Stanford GloVe) were tested. Since those are 300-dimensional, the medical embeddings were retrained for equal embedding size and tested once again. The results are shown respectively in table 4.2 and 4.3.

MIT movie	Embedding Type	Precision	Recall	F1
Simple	Medical	0.71	0.67	0.69
Complex	Medical	0.54	0.48	0.51

Table 4.2: Results on MIT movie dataset with medical, 100-dimensional embeddings.

MIT movie	Embedding Type	Precision	Recall	F1
Simple	General	0.87	0.86	0.86
Complex	General	0.72	0.71	0.72
Simple	Medical	0.72	0.67	0.70
Complex	Medical	0.54	0.42	0.52

Table 4.3: Results on MIT movie dataset with 300-dimensional embeddings. The general embeddings are the Stanford GloVe embeddings, whereas the medical embeddings are the ones trained on MIMIC-CXR.

As can be seen from the comparisons on the MIT Movie dataset, the embedding size has little to no influence on the performance. As expected, the SotA embeddings perform a lot better on this dataset, as MIT Movie is a general English dataset.

4.4.3 Comparison of mixed, medical and general embeddings

So far, the implementation and embeddings seem to work as expected. Medical data are domain-specific, but do use (parts of) the general language and its rules. Therefore, enriching the medical data with (some) general data might improve performance and likely improves the robustness of

the embeddings. The next experiment tests this hypothesis. As mentioned before, the medical GloVe embeddings are trained on 15.6M PhysioNet tokens. The mixed GloVe adds 17M tokens (32.5M total) from the Westbury corpus. The embedding size in this experiment is 100, as previous experiments have not shown a benefit of larger embedding size. The results of the mixed embeddings on the different datasets are shown in table 4.4. Those are also the embeddings that will be used in the Domain Adversarial Neural Network experiments, later on.

	Precision	Recall	Mixed F1
Simple Movie	0.79	0.79	0.79
Complex Movie	0.65	0.63	0.64
WNUT	0.59	0.49	0.54

Table 4.4: Results of 100-dimensional mixed embeddings on the different datasets: biomedical, simple general and complex general data.

As expected, mixed embeddings prove a lot more robust on general datasets. Both on the simple and complex MIT Movie dataset, the LSTM-CRF with mixed embeddings performs a lot better. However, on the specific WNUT dataset, the performance remained the same. To get a better overview of the combined results, the F1 score per dataset per embedding type is shown in table 4.5. This table does not show new results, but merely summarizes the previously recorded results. That is also the reason there is no value for general F1 on the WNUT (lab procedures) dataset.

	Medical F1	Mixed F1	General F1 (SotA)
Simple Movie	0.69	0.79	0.86
Complex Movie	0.51	0.64	0.72
WNUT	0.54	0.54	

Table 4.5: Comparison of 100-dimensional medical embeddings, 100-dimensional mixed embeddings and 300-dimensional State-of-the-Art embeddings on the different datasets.

The comparison shows that on the MIT Movie dataset, the performance of the mixed embeddings approaches that of the GloVe embeddings. This is interesting, because the SotA GloVe embeddings are 300-dimensional, compared to 100-dimension for the mixed embeddings. Furthermore, the SotA embeddings have a vocabulary of $400,000$ words and are trained on 6B (!) tokens. This shows that adding general data to the medical data, allows the word embeddings to capture more language-specific features. At the same time it indicates that word embeddings can perform well without a huge vocabulary and billions of training tokens. Finally, adding general data to embeddings does not automatically lead to better capturing of the specific domains, as indicated by the WNUT F1 scores. This can be explained two ways: the first option is that both datasets, used to create mixed embeddings, are too similar. E.g. both the 'medical' and 'general' embeddings differ in the same way from the 'lab protocol' (WNUT) data. Adding general data to the medical data would then not add new information. The second option is that adding general data is simply not beneficial. Either the general language rules can be learned from the domain-specific (medical) text, or the general language rules do not contribute to domain-specific task (WNUT) performance. Once there is a working embedding evaluation method and promising embeddings have been developed, this should

be reinvestigated.

4.4.4 GloVe summary

The GloVe experimenting shows that medical embeddings capture a good amount of general English. Furthermore, the SotA general embeddings have been trained on 6B tokens, compared to the 15M tokens that were used to train the medical embeddings. In the subsection, the performance of mixed, medical and GloVe embeddings is compared on all three datasets. In this experiment I confirm once again that the embeddings align with expectation: the mixed embedding performance lies between the medical and GloVe embeddings. The performance on a general task increases, when medical embeddings are enriched with general data. However, enriching medical embeddings with general data has not proven to improve the performance on a medical task. The performance on a domain-specific task does not increase with general language data added to the embeddings. The reason for this remains open: it could be due to the enrichment having no effect, but also due the fact that the 'medical' data differs in the same way from 'biomedical' data as 'general' data does. For example, the same words being out-of-vocabulary.

In any case, to assess the embedding quality, a way of evaluation is required. The experiments in this chapter merely give an idea of the embedding performance, but the available datasets in this Flair framework do not resemble medical data. Therefore, the next step is creating an evaluation method myself. One that does not require much annotated data.

Chapter 5

Domain Adversarial Neural Network (DANN)

The Domain Adversarial Neural Network was originally developed for image classification. However, not long after its first release, multiple papers also show its use in NLP. This chapter will first discuss the theoretical overview of the DANN, subsequently its use in different NLP tasks, my own implementation of the DANN and finally the DANN experiments. Also, when talking about 'the DANN' in this chapter, it means the concept of a Domain Adversarial Neural Network. The specific implementation differs per paper.

5.1 DANN background

The DANN is not the first model that is used for domain adaptation in NLP. The distinct approach that the DANN takes, is that the task classification and the domain adaptation happens in the same, single, neural architecture. The approach is motivated by a theory of Ben-David et al. (Ben-David et al., 2007). This theory shows that, to find a model with a small error on the target domain, the learning algorithm should minimize a trade-off between the source domain error and the $\mathcal{H}$-divergence. The $\mathcal{H}$-divergence quantifies the capacity of hypothesis class $\mathcal{H}$ to distinguish between samples generated by the source domain and the target domain (Ganin et al., 2016). So, to control the $\mathcal{H}$-divergence, one has to find a representation of the samples where both the source and target domain are as indistinguishable as possible. Under such representation, a hypothesis with a low source domain error should perform well on the target domain. In more practical words: the DANN focuses on learning features that combine discrimination and domain-invariance. This is done by optimizing both the underlying features and two discriminative classifiers operating on those features, at the same time. Those discriminative classifiers are a task classifier and a label predictor. The label predictor predicts the domain of each sentence in a batch during training. The task classifier predicts the class labels and is used during training and testing. The objective for the classifiers is to minimize error on the training set. However, the underlying feature mapping is optimized to minimize the loss of the label classifier and at the same time maximize the loss of the domain classifier. This maximization of loss works adversarially with respect to the domain classifier: the better the domain classifier predicts

the domain, the more domain-invariant the features should become. This process is achieved by using a so-called Gradient Reversal Layer (GRL): a layer that acts as the identity during forward pass, but during backward pass it scales the gradients by $-\lambda$ (Ganin et al., 2016).

Several papers made use of such version of a DANN and most of them extended upon it (Fu et al., 2017; Naik and Rosé, 2020; Peng et al., 2021). The key takeaway from these papers is that a DANN consists of a feature extractor, domain classifier, label decoder and GRL. The default component for the classifiers is usually an Multi-Layer Perceptron (MLP), which will also be the case in the described studies, unless otherwise specified. The specific feature extractor is usually chosen based on the task at hand. In this thesis, the DANN will be used for sequence classification. Therefore, the next subsections will describe classification and sequence labelling in greater detail. Apart from this, the DANN has also been used for dependency parsing (Sato et al., 2017) and relation extraction (Fu et al., 2017).

5.1.1 Domain Separation Network

One of the earliest and most important improvements on the DANN, was the Domain Separation Network (DSN) (Bousmalis et al., 2016). In short, Bousmalis et al. conclude that the DANN only makes use of the shared features of both domains. This is not optimal, since both domains will also have domain-specific features. Therefore, Bousmalis et al. attempt to create two different feature spaces for both the source and target domain. Their model, the DSN, uses three feature extractors: two private encoders for each domain and one shared encoder. Instead of focusing on shared features, the DSN attempts to learn domain-specific features for both domains. The shared encoder makes use of the adversarial loss function, whereas the private encoders are treated normally. During classification, based on the predicted domain, one of the private encoders is used, in combination with the shared encoder. The DSN performs slightly better (1-3 points F1-score) on most tasks. However, since the DSN is more complicated to implement and the DANN fulfills its job as a downstream task, I stick with the DANN for this thesis.

5.1.2 Classification

Two studies (Alam et al., 2018) (Naik and Rosé, 2020) used the DANN for trigger event classification. The second one (Naik and Rosé, 2020) is the most relevant for this thesis. In this study, trigger identification is regarded as a token-level classification task: for each token, it is predicted whether it is an event trigger. They experiment with several representation learners. They compare an LSTM, BiLSTM and BERT model with and without adversarial learning. Next to experimenting with the representation learners, they use an event classifier, being a single-layer MLP and a domain predictor, being a three-layer MLP. Unlike Ganin et al., they first train their domain predictor and subsequently they train the representation learner and event classifier. From the six resulting models, the adversarial BERT model appeared to perform best. The adversarial models had an average F1 score improvement of 3.9 to each of the models. Finally, they note an improvement from 51.5 to 67.2 in F1 score when training with 1% labeled target domain data, compared to using no labelled target domain data at all. This shows that the adversarial approach has value on low-resource domains and even more once some labeled data become available.

Another paper that used the DANN (principle), used it for multi-class text classification (Ding

et al., 2019). The DANN they used, had a Convolutional Neural Network (CNN) as feature extractor. Their model (called MDAnet) and the DANN were both tested, amongst other models, on the Consumption Intention Identification (CII) task (Ding et al., 2018) and a sentiment analysis task (Blitzer et al., 2007). MDAnet improved upon the DANN by a large margin on the CII task (30% - 40%) showing that there is a lot of room for improvement for the DANN. On the sentiment analysis task, the difference was about 10% in accuracy scores. At the same time, the DANN does perform comparable or better than multiple other models, showing its use as a baseline.

5.1.3 Sequence labelling

The DANN has also been used in Sentiment Analysis (SA) (Kim et al., 2017), by means of the DSN variant. Kim et al. (2017) investigate the use of the DSN in spoken language understanding (e.g. on Microsoft Cortana's input). More specifically, they use the DSN to adapt from Engineered domains (artificial data) to Live domains (user data). They use a BiLSTM with character embeddings and pretrained word embeddings as feature extractor. They report a significant increase in F1-score for adapting Engineered to Live data. Two interesting metrics they use to validate their performance and compare it with the DANN, include proxy $\mathcal{A}$-distance and the Jaccard coefficient of the different vocabularies. The two metrics respectively show the generalisation error in discriminating between the source and target dataset and the distance between both domains' vocabulary.

One recent paper builds upon the concept of the DANN, for the task of NER (Peng et al., 2021). They extend the DANN architecture by adding an entity-aware component, but they keep the DANN-specific part of the GRL below the discriminator. They use a BiLSTM as feature extractor, a Conditional Random Field (CRF) as label decoder and a sigmoid function as domain discriminator. Additionally, as extra input to their model, they add character embeddings to each word embedding. Finally, they regard BiLSTM-CRF as a widely-used standard, thus baseline model, for Named Entity Recognition. They also use DANN itself as a model to compare theirs to.

5.2 DANN implementation

The pseudocode for the original DANN architecture is given in figure 5.1 and the overview of the model architecture can be seen in figure 5.2. The BiLSTM-CRF architecture (Huang et al., 2015) is used as a baseline (and in the DANN itself) in multiple papers, including the only paper using DANN for NER. Therefore, the implementation in this study will also use the BiLSTM as feature extractor and the CRF as label decoder. Multiple papers use a Multi-Layer Perceptron (MLP), one of the simplest classifiers, as domain classifier. This is therefore also the choice for my implementation. Since the GloVe embeddings can (initially) be developed in a simple DANN architecture, I don't use additional options. For example, some papers also use character embeddings in addition to the extracted features. Next to this, manual feature selection would be possible. Those features would normally be added between the feature extractor and the classifiers. To summarize: as feature extractor a one-layer BiLSTM is used, as label decoder a CRF is used and as (domain) classifier a MLP was chosen.

Algorithm 1 Shallow DANN – Stochastic training update

Input:
— samples $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ and $T = \{\mathbf{x}_i\}_{i=1}^{n'}$,
— hidden layer size D,
— adaptation parameter λ,
— learning rate μ,
Output: neural network $\{\mathbf{W}, \mathbf{V}, \mathbf{b}, \mathbf{c}\}$

$\mathbf{W}, \mathbf{V} \leftarrow$ random_init(D)
$\mathbf{b}, \mathbf{c}, \mathbf{u}, d \leftarrow 0$
while stopping criterion is not met **do**
  for i from 1 to n **do**
    `# Forward propagation`
    $G_f(\mathbf{x}_i) \leftarrow \text{sigm}(\mathbf{b} + \mathbf{W}\mathbf{x}_i)$
    $G_y(G_f(\mathbf{x}_i)) \leftarrow \text{softmax}(\mathbf{c} + \mathbf{V}G_f(\mathbf{x}_i))$
    `# Backpropagation`
    $\Delta_{\mathbf{c}} \leftarrow -(\mathbf{e}(y_i) - G_y(G_f(\mathbf{x}_i)))$
    $\Delta_{\mathbf{V}} \leftarrow \Delta_{\mathbf{c}}\, G_f(\mathbf{x}_i)^\top$
    $\Delta_{\mathbf{b}} \leftarrow \left(\mathbf{V}^\top \Delta_{\mathbf{c}}\right) \odot G_f(\mathbf{x}_i) \odot (1 - G_f(\mathbf{x}_i))$
    $\Delta_{\mathbf{W}} \leftarrow \Delta_{\mathbf{b}} \cdot (\mathbf{x}_i)^\top$
    `# Domain adaptation regularizer...`
    `# ...from current domain`
    $G_d(G_f(\mathbf{x}_i)) \leftarrow \text{sigm}(d + \mathbf{u}^\top G_f(\mathbf{x}_i))$
    $\Delta_d \leftarrow \lambda(1 - G_d(G_f(\mathbf{x}_i)))$
    $\Delta_{\mathbf{u}} \leftarrow \lambda(1 - G_d(G_f(\mathbf{x}_i)))G_f(\mathbf{x}_i)$
    $\text{tmp} \leftarrow \lambda(1 - G_d(G_f(\mathbf{x}_i))) \times \mathbf{u} \odot G_f(\mathbf{x}_i) \odot (1 - G_f(\mathbf{x}_i))$
    $\Delta_{\mathbf{b}} \leftarrow \Delta_{\mathbf{b}} + \text{tmp}$
    $\Delta_{\mathbf{W}} \leftarrow \Delta_{\mathbf{W}} + \text{tmp} \cdot (\mathbf{x}_i)^\top$
    `# ...from other domain`
    $j \leftarrow$ uniform_integer$(1, \ldots, n')$
    $G_f(\mathbf{x}_j) \leftarrow \text{sigm}(\mathbf{b} + \mathbf{W}\mathbf{x}_j)$
    $G_d(G_f(\mathbf{x}_j)) \leftarrow \text{sigm}(d + \mathbf{u}^\top G_f(\mathbf{x}_j))$
    $\Delta_d \leftarrow \Delta_d - \lambda G_d(G_f(\mathbf{x}_j))$
    $\Delta_{\mathbf{u}} \leftarrow \Delta_{\mathbf{u}} - \lambda G_d(G_f(\mathbf{x}_j))G_f(\mathbf{x}_j)$
    $\text{tmp} \leftarrow -\lambda G_d(G_f(\mathbf{x}_j)) \times \mathbf{u} \odot G_f(\mathbf{x}_j) \odot (1 - G_f(\mathbf{x}_j))$
    $\Delta_{\mathbf{b}} \leftarrow \Delta_{\mathbf{b}} + \text{tmp}$
    $\Delta_{\mathbf{W}} \leftarrow \Delta_{\mathbf{W}} + \text{tmp} \cdot (\mathbf{x}_j)^\top$
    `# Update neural network parameters`
    $\mathbf{W} \leftarrow \mathbf{W} - \mu\Delta_{\mathbf{W}}$
    $\mathbf{V} \leftarrow \mathbf{V} - \mu\Delta_{\mathbf{V}}$
    $\mathbf{b} \leftarrow \mathbf{b} - \mu\Delta_{\mathbf{b}}$
    $\mathbf{c} \leftarrow \mathbf{c} - \mu\Delta_{\mathbf{c}}$
    `# Update domain classifier`
    $\mathbf{u} \leftarrow \mathbf{u} + \mu\Delta_{\mathbf{u}}$
    $d \leftarrow d + \mu\Delta_d$
  end for
end while

Note: In this pseudo-code, $\mathbf{e}(y)$ refers to a "one-hot" vector, consisting of all 0s except for a 1 at position y, and $\odot$ is the element-wise product.

Figure 5.1: Pseudocode for stochastic training update of the DANN. From (Ganin et al., 2016).

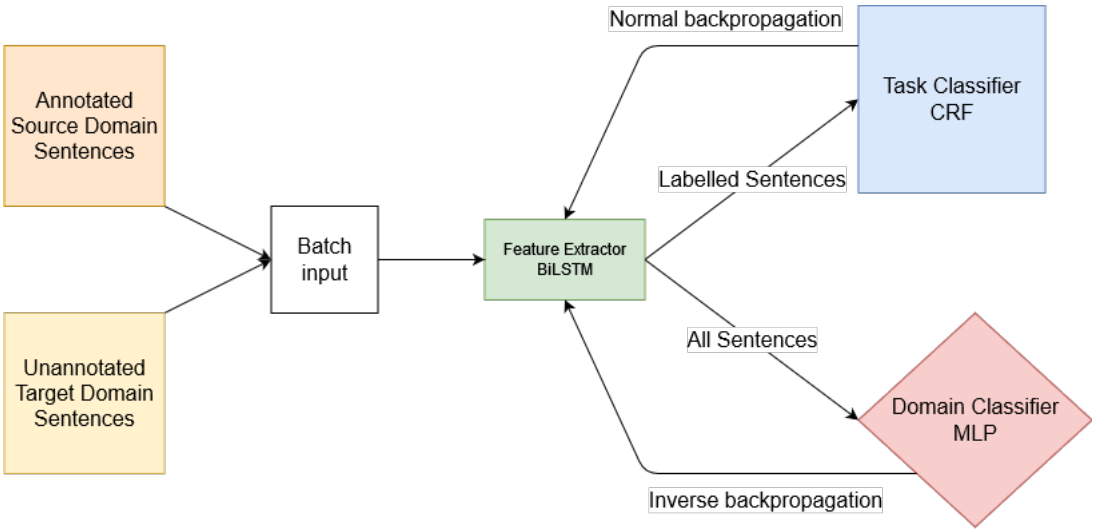


Figure 5.2: DANN Architecture. Half of the batch is source domain data annotated for a task, the other half is unannotated target domain data. The whole batch has domain annotations. The batch goes through the feature extractor, after which the features are sent to the task classifier and domain classifier. The unique aspect of this architecture is the inverse backpropagation from the domain classifier to the feature extractor.

5.2.1 BiLSTM-CRF

The BiLSTM-CRF was originally developed by Huang et al. (Huang et al., 2015). The idea behind it is to utilize input features in multiple ways. First, the BiLSTM captures past and future input features in both directions. This is done by having two LSTM layers, where one starts reading the sentence at the beginning and the other layer starts at the end. The two layers are then concatenated for the feature output. On top of that the CRF layer can exploit sentence-level tag information. The CRF operates on the probability that a word has a certain tag and the probability of a certain tag, given the previous tag. Based on those probabilities, it computes the most likely tag sequence. Huang et al. (2015) show that the BiLSTM-CRF is both more robust and less dependent on word embeddings than previous models. Even though the model has been around for some time, its results still prove to be a reliable baseline for many current studies. Huang et al. report an F1-score of 84.26 on the CoNLL-2003 dataset, with randomly initialised word embeddings and additional spelling and context features. Without the additional features, but with word embeddings, performance reaches up to 84.74. They further note that the CRF models' performance significantly decreases without the spelling and context features. A year after the original paper, Lample et al. (Lample et al., 2016) improved on the BiLSTM-CRF, since it uses so many hand-engineered features. They add a character-level component to the architecture, which removes the need of hand-crafted features. They use a hidden layer size of 25 (in both the forward and backward layer) and concatenate those as input to the CRF. They also use a dropout of 0.5 in the BiLSTM. To train the model, learning rate is set to 0.01, gradient clipping to 5.0, They report an F1 score of 90.20 for their model without character embeddings (but with pretrained word embeddings).

5.2.2 Domain Classifier

I use a two-layer MLP with one dropout layer in between the two fully-connected layers. The logsoftmax activation function is used, together with the negative log-likelihood loss function.

5.2.3 Embeddings as used in DANN

The embeddings that are used in the DANN, are the 100-dimensional embeddings described in 4.4.3. The only difference is that the medical embeddings were retrained for a vocabulary size of $10,000$. There are less medical data available, so the larger vocabulary size does not lead to memory issues. This means that the medical embeddings have a larger vocabulary, whereas the mixed embeddings are trained on double the number of tokens. They are both compared to SotA general embeddings: Stanford GloVe.

5.3 DANN Experiments

Initially, four DANN experiments were planned: one preliminary experiment and three DANN experiments. However, the last experiment became infeasible, which leaves us with the preliminary experiment and two DANN experiments. This will be further discussed in section 6.1.2. The preliminary experiment evaluates the BiLSTM-CRF performance on the CoNNL-2003 dataset (general data Named Entity Recognition). This experiment confirms the performance of the BiLSTM-CRF outside the DANN architecture and can function as a baseline to compare the DANN results to. The first DANN experiment will then perform adaptation from the general English domain (CoNNL-2003) to the Broad Twitter Corpus. This experiment resembles one experiment in (Peng et al., 2021), as they do the same for a different Twitter dataset. In this first DANN experiment, I also evaluate the effect of word embeddings and dropout in the DANN. The second DANN experiment will then compare different types of word embeddings when used in the DANN. This is where the embeddings that I trained myself in section 4.4.3 come in. The source domain in this experiment is CoNNL-2003 and the target domain is GENIA (biomedical, human blood cells). GENIA would be closest to an English annotated medical dataset. So, this second experiment measures the performance of mixed, medical and general embeddings when adapting from the general English to the medical English domain. Throughout the experiments, it is important to know that only the training set was used to create vocabularies. All 'new' words in the validation and test set were mapped to an <UNK> token. (So, the larger the domain discrepancy, the more words are mapped to the UNK token.) To summarize, the DANN experiments will answer the following questions:

- To what extent does hyperparameter selection affect the DANN performance? What is the effect of dropout in the DANN? And what is the effect of word embeddings in the DANN?
- To what extent is the DANN suited for sequence labelling tasks?
- How does the DANN perform when adapting from the general domain to Twitter data? And from the general to the biomedical domain?

5.4 Preliminary BiLSTM-CRF experiment

As mentioned this initial experiment aims to find optimal parameters for the BiLSTM-CRF (as used in the DANN), on the CoNNL-2003 dataset, to create a baseline. To find the (approximate) optimal value for the hidden nodes for the hidden layers, I experimented with [16, 32, 64] hidden nodes in combination with a batch size of [16 and 32]. 64 hidden nodes gives the best results, regardless of batch size. This is not surprising, as more nodes allows for more memory of the network. However, since I used early stopping based on a holdout validation set, this increase in memory should not have lead to overfitting. The other conclusion is that a batch size of 16 outperforms a batch size of 32 for an equal number of hidden nodes. To see whether even more hidden nodes allow for a better performance, I experimented with 96 and 128 hidden nodes. I compared this on the batch size of 32, as the training time with this batch size is lower. The effect of adding hidden nodes seems to fall off, when reaching 128 hidden nodes. Finally, I consider a batch size of 64 and 256 (close to 300) hidden nodes, as this was the reported number of hidden nodes in the original study of the BiLSTM (Huang et al., 2015). This does not yield better results. The results of this preliminarly experiment are shown in table 5.1.

	Batch	Hidden	Epochs*	Train F1	Test F1
1	16	16	33	0.61	0.52
2	16	32	21	0.73	0.57
3	16	64	26	0.80	0.59
4	32	16	38	0.54	0.43
5	32	32	45	0.67	0.56
6	32	64	30	0.75	0.58
7	32	96	26	0.78	0.61
8	32	128	23	0.80	0.61
9	64	256	45	0.74	0.58

Table 5.1: Performance for the BiLSTM-CRF (same as used in DANN), for different batch sizes and hidden nodes per hidden layer. *The number of epochs is not directly related to training duration, as one epoch takes longer for smaller batch sizes.

5.5 Experiment 1: NER on Twitter (English)

In this experiment I aim to reproduce results reported in a study by Peng et al. (Peng et al., 2021). This study compares its own model with other models, of which one is the DANN. The reported result for the DANN is among the few that are done with sequential DA models. Peng et al. use CoNNL-2003 (Named Entity) as source domain and a Twitter dataset (Lu et al., 2018) as target domain. However, there will be a few things different in this reproduction experiment. First of all, the used Twitter dataset cannot be accessed online. So, I choose to use the Broad Twitter Corpus (Derczynski et al., 2016) for the experiment. Next to that, they use pretrained word embeddings in the feature extractor of all models that are compared. The feature extractor they use in (most of) their comparison models is also a BiLSTM. I will initially use randomly initialised word embeddings

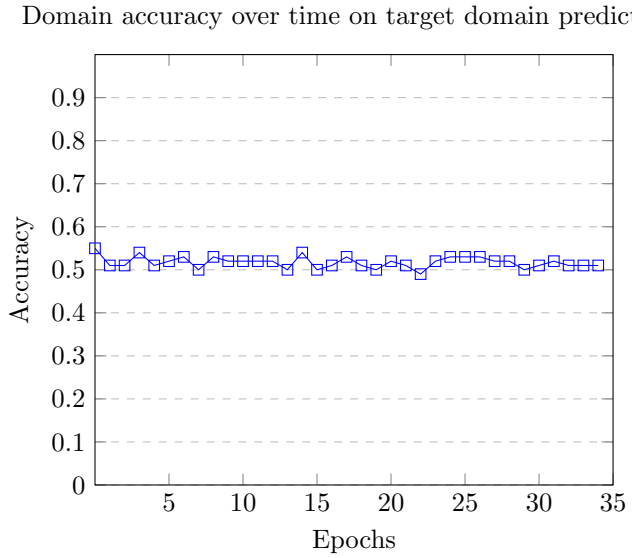
to test the DANN's performance. Finally, the data modifications can be found in section 3.4. Peng et al. do not mention any modifications in their study.

5.5.1 Experiment 1a: DANN performance

I start by experimenting with the DANN, without any additional features, except for two dropout layers. The same parameters are used, as in the preliminary experiment, to also be able to compare the DANN results to the baseline. The source domain data are CoNNL-2003 and the target domain data are the Broad Twitter Corpus. The results are shown in table 5.2. The source domain F1 score is based on the validation set, whereas the target F1 score is based on the same target data used to train the model. As a reminder: this target data are only used to train the domain classifier, not the task classifier. Additionally, I show that the DANN works as expected, by showing the domain accuracy over different epochs in the figure below. This stable line around 0.5 is the same for each training session. It shows that the domain classifier is confused and not able to distinguish the domains.

	Batch	Hidden	Source Val. F1	Target F1	Domain Acc
1	16	16	0.58	0.31	0.52
2	16	32	0.57	0.33	0.50
3	16	64	0.60	0.32	0.52
4	16	96	0.66	0.32	0.48
5	32	16	0.50	0.28	0.45
6	32	32	0.46	0.27	0.51
7	32	64	0.63	0.29	0.47
8	32	96	0.69	0.30	0.53

Table 5.2: Performance for the DANN for different combinations of batch size and hidden nodes. The F1 score is reported for the source and target domain used during training. During training, only the source domain task labels were used. The domain accuracy during training is shown.



5.5.2 Experiment 1b: Using dropout layers

As other studies used (multiple) dropout layers, I experimented with three dropout layers, either deactivated or active with 0.5 dropout. Even though Lample et al. (2016) Lample et al. (2016) mention that word embeddings tend to have a bigger impact on BiLSTM-CRF performance, I first test the effect of dropout. This allows for checking the effect of dropout in the embedding layer, as the embedding layer is frozen when pretrained embeddings are used. Based on (target domain) performance in the last experiment, a batch size of 16 and a hidden layer size of 64 is used for this experiment. Since an effect of dropout was already assumed, two dropout layers were used with a dropout of 0.5 in experiment 1a. The dropout was applied after the feature extractor (BiLSTM) and between the MLP layers of the domain classifier. The performance with different dropout layers is reported in table 5.3.

	Embeddings	LSTM	Classifier	Source Val. F1	Target F1
1	0	0	0	0.61	0.32
2	0	0	0.5	0.60	0.33
3	0	0.5	0	0.57	0.32
4*	0	0.5	0.5	0.60	0.32
5	0.5	0	0	0.61	0.33
6	0.5	0	0.5	0.60	0.33
7	0.5	0.5	0	0.59	0.31
8	0.5	0.5	0.5	0.61	0.33

Table 5.3: Performance of the DANN using different values of dropout in different parts of the model. * This setting was used in the first experiment.

5.5.3 Experiment 1c: Using word embeddings

In this third sub-experiment I investigated the effect of word embeddings in the DANN. As the main goal of the first experiment is aimed at replicating the result of Peng et al., I will use State-of-the-Art 100-dimensional GloVe embeddings. To make this final DANN sub-experiment as complete as possible, I experiment the batch sizes 8, 16 and 32 in combination with different numbers of hidden nodes: 32, 64 and 96. Those combinations seemed most promising, based on the experiments so far. Finally, I test once with a large number of hidden nodes, again because Peng et al. reported the best performance with 300 hidden nodes. Dropout setting 4 is used from the previous sub-experiment: no dropout on the word embeddings and one dropout layer after both the BiLSTM output and the first fully-connected layer in the domain classifier. Finally, another F1 score is added, according to the CoNNL-2000 evaluation script. This sentence F1 score over multiple tags, is the percentage of sentences that are completely correct. Since this F1 score was added in the final stages of the thesis, it is only included here; in the most complete DANN table. An overview of the results of this sub-experiment is shown in table 5.4.

	Batch	Hidden	Source Val. F1	Target F1	Sent. F1
1	8	32	0.70	0.42	0.44
2	8	64	0.71	0.44	0.44
3	8	96	0.71	0.44	0.44
4	16	32	0.70	0.43	0.42
5	16	64	0.73	0.43	0.43
6	16	96	0.71	0.45	0.43
7	32	32	0.72	0.41	0.35
8	32	64	0.71	0.42	0.36
9	32	96	0.72	0.42	0.39
10	32	256	0.73	0.43	0.38

Table 5.4: Performance of the DANN with different combinations of batch size and number of nodes in the hidden layer, while using word embeddings.

5.5.4 Experiment 1 overview

In this comparison the most-frequent-tag baseline is added, following Jurafsky et al. (Jurafsky, 2000). This baseline simply allocates the tag that most frequently appears for a given word, to that word. The baseline performance, the reported score of Peng et al. and the best DANN performance are shown in table 5.5. The DANN score that is reported here, corresponds to entry 6 of table 5.4, which was achieved using 100-dimensional Stanford GloVe embeddings. Both Peng et al. and our DANN perform significantly better than the most-frequent-tag baseline.

	Precision	Recall	F1
Baseline	0.17	0.10	0.13
Peng*	0.46	0.59	0.51
DANN	0.58	0.37	0.45

Table 5.5: Comparison of baseline (most frequent tag), Peng's reported DANN result, and our DANN result on the Twitter NER task.
*As mentioned, Peng used a different Twitter dataset.

5.6 Experiment 2: POS tagging on English medical domain

This second experiment will discuss the use of the different embedding types, that I trained in Chapter 4, in the DANN in detail. The embedding specifications for the English embeddings can be found in 4.4.3. I will compare the performance of mixed, medical and SotA general embeddings while adapting from a general to a medical dataset. An important change relative to the previous experiment, is that I now switch to Part-of-Speech tagging. This is the only option, as currently no general and medical dataset, using the same Named Entity tag set, exist. As mentioned in the preprocessing section (3.4), the POS dataset was adjusted for this task.

The DANN is used with a batch size of 16 and hidden layer size of 96. Recall, precision and F1-score are reported, next to the retrieval rate, in table 5.6. The retrieval rate shows how many words from the source domain vocabulary are found in the word embedding vocabulary. Words that are not found in the embedding file are mapped to the <UNK> token. When pretrained embeddings are used, the embedding layer is frozen; i.e. the weights are not updated during training. Since all non-retrieved words are mapped to the <UNK> token, and the embedding layer is frozen when using word embeddings, the performance is expected to drop with a lower retrieval rate. In this case, the retrieval is calculated on the source domain: the CoNNL-2003 data. The GENIA retrieval is reported to get more insight in the word embeddings, but this was not used in any way.

	CoNNL retrieval	GENIA retrieval	Precision	Recall	F1	Source F1
Baseline	-	-	0.02	0.03	0.02	-
No embeddings	-	-	0.77	0.67	0.72	0.77
Gen. embeddings	87.64%	51.98%	0.82	0.67	**0.74**	0.80
Med. embeddings	14.04%	18.11%	0.74	0.65	0.70	0.77
Mix. embeddings	30.17%	19.45%	0.77	0.65	0.70	0.78

Table 5.6: The baseline corresponds to the most-frequent-tag (for a given word). Precision, recall and F1 scores are reported for target domain. Additionally, source domain F1 score corresponds to the holdout validation set performance. Scores are reported for different English pretrained embeddings used in the DANN. Information on the embeddings can be found in section 4.4.3.

Chapter 6

Discussion

The main aim of this thesis was to create word embeddings for the Dutch medical domain. Additionally, the DANN was developed as a means of evaluation for the word embeddings. During the experiments, it became clear that training Dutch medical embeddings was not feasible during this thesis. The next subsections will discuss the results, the implications and shortcomings of this thesis and recommendations for future work.

6.1 Analysis

6.1.1 GloVe embeddings

I started the GloVe experiments with three questions: how do medical embeddings perform on a biomedical dataset? How do medical embeddings generalise to general English and do mixed embeddings have an advantage over medical embeddings? The main takeaway from the GloVe experiments, is that the hardware should allow for a larger vocabulary size. The vocabulary size in this thesis was a limiting factor. Therefore, the the effect of vocabulary size remains unclear. One thing is certain: a vocabulary size of 5000 leads to poor results. Apart from this, there are certain observations from the different embedding types; the general, medical and mixed embeddings. The medical embeddings seem to capture a decent amount of the general language, as the embeddings perform reasonably well, compared to the Stanford GloVe embeddings, on a general task. Finally, the comparison of mixed and medical embeddings does not prove that enriching the medical embeddings with general textual data improves the performance on a medical task. As expected, the mixed embeddings perform better than medical embeddings on a general task. However, the performance on the biomedical domain, WNUT-2020, did not increase with mixed embeddings. The reason for this remains open: it could be due to the chosen general data not being enriching, but also due to the dataset (WNUT-2020) not resembling medical data. If the first reason is the case, adding different or more general data could give an improvement. If the second reason is the case, the medical and general data leverage the same features for the WNUT task: both domains differ too much from the specific WNUT biomedical domain. Once SotA medical embeddings - or at least evaluated medical embeddings - have been developed, this should be investigated further.

6.1.2 DANN experiments evaluation

BiLSTM-CRF performance (Preliminary Experiment)

The preliminary experiment, with the BiLSTM-CRF, shed some light on the effect of the different hyperparameters. First of all, the F1 score on the CoNNL-2003 dataset seemed to be better for lower batch sizes, given an equal number of hidden nodes. Next to this, overfitting (on the source domain) did not occur, since the source domain performance did not decrease, until the test domain performance did. This is most likely due to the early stopping criterion. Underfitting due to too few hidden nodes did not seem to occur either, as I ran the model with a large number of (extra) hidden nodes and performance decreased. Finally, the reported performance for the BiLSTM-CRF is not even close to State-of-the-Art results. There are multiple reasons for this. The first reason is that the early stopping in combination with the learning rate could have been too strict. I did not experiment with the learning rate, apart from using a scheduler based on a patience of 5. Using different values for both patience factors (early stopping and scheduler), could improve the results. The second reason is that most - if not all - state-of-the-art results were achieved by extensive preprocessing. Finally, most BiLSTM-CRF's achieving good results used character embeddings, additional hand-crafted features (up to 20) (Huang et al., 2015) and (different) word embeddings.

DANN performance on NER task (Experiment 1)

DANN experiment 1a investigated the effects of hyperparameter selection, dropout and word embeddings for the DANN's performance. the domain classifier is confused by tracking the domain classifier performance over a full model run. The domain classifier consistently ends up around 50% accuracy, indicating successful confusion. Finally, when comparing the DANN performance on 32 batch size and 96 hidden nodes with the BiLSTM-CRF performance for the same batch size and hidden nodes, it shows that the DANN slightly improves on the source validation data (0.61 vs 0.69), which are the data that are used in both models. This indicates that the DANN learns additional information from using the domain classifier.

Experiment 1b was conducted with the best model from experiment 1a: a batch size of 16 and 64 hidden nodes. This model was used to compare the effect of dropout on. The observation in this experiment is that dropout does not seem to have an effect on the model performance. This could be due to early stopping already preventing the model from overfitting. It could also be due to the model reaching its maximum adaptation potential with the current features. In that case, the features or data need to be improved, before the model is able to learn more about the target domain.

The final experiment, 1c, tests the effect of word embeddings in the DANN. The first observation is that using word embeddings increase performance on the target domain a lot. In this section, an additional F1 score was added: the sentence accuracy F1 score. It is worthwhile to note that this sentence F1 is higher for a smaller batch size and seems to increase with more hidden nodes. This could indicate that a larger batch size favours classes that are more present in the data, resulting in more individual labels correctly classified, but less sentences completely correct.

The DANN experiment concludes with a comparison of the best obtained model to the most-frequent-tag baseline and the reported DANN score of Peng et al. (2021). The central question here is to what extent the DANN is suitable for sequence classification. The most frequent tag baseline performs reasonable, but a lot worse than both models. It shows that the DANN is able to

capture patterns in the text that predict whether a word is a Named Entity, even for text from a different domain. So, the DANN is suitable for sequence classification, especially given the room for improvement. It is likely that the DANN is still underfitting due to a lack of features. Suggestions to improve this are given later in this chapter.

Performance of medical embeddings in the DANN (Experiment 2)

The second DANN experiment investigated the performance of the medical embeddings that I trained in chapter 4. Next to this, it investigates to what extent the DANN is able to adapt from a general domain to a medical domain. The most-frequent-tag baseline achieved very poor results in this experiment. This is most likely due to the much higher number of tags. In experiment 1 8 NER tags were used, relative to 41 POS tags in experiment 2. The next point of interest is the retrieval rate. In the current implementation, only the source domain vocabulary was used to initialise the embeddings. The medical advantage of the embeddings is therefore minimal. This issue is discussed further in section 6.2.1. Next to this, the importance of a large vocabulary is once again shown by the Stanford GloVe embeddings. The vocabulary size of $400,000$ gives a retrieval rate of 52% on the biomedical GENIA dataset.

Apart from the general observations, I also put detailed overviews of the performance per tag in the Appendix (see 7.3). The most relevant are the Common Noun (NN), Proper Noun (NNP) and Common Noun Plural (NNS). For more Part-of-Speech tag explanations, I would like to refer to the Brown Corpus manual (Francis and Kucera, 1979). The detailed tag analysis shows that the DANN assigns NNP a lot more, when less word embeddings are randomly initialized. When the Stanford GloVe embeddings are used, 85573 NNP's are assigned, compared to 17184 times when not using pretrained embeddings. The opposite effect can be observed for NN's. This is especially interesting because the training data only have 361 true NNP's versus 107445 NN's. The model without pretrained embeddings also assigns NN and NNP to true NNS's. It is unclear why this effect appears. Recognizing plurals should be easier for the model when using additional features, like word endings, or by using character embeddings in the model. With the model settings as is, learning the embeddings during training seems to give the best results.

Experiment 3 (infeasible)

The plan originally included a third experiment, however this experiment was infeasible. The idea of the experiment was to evaluate Dutch GloVe embeddings the same way as the English embeddings in DANN experiment 2. However, as the randomly initialized embeddings performed better than the embeddings I trained, this third experiment was not reasonable anymore. Instead, the English embeddings should first be evaluated and improved. This was already discussed in section 6.1.1.

6.2 Limitations and shortcomings

The first two limitations and shortcomings described here, explain two factors that should influence the interpretation of the results in this thesis. The other limitations and shortcomings mostly explain why the results in this thesis did not get close to SotA model results.

6.2.1 Model implementation

One limitation is an implementation choice which, I realized later, limited the model severely. As I mentioned throughout the thesis, I loaded the word embeddings based on the source domain. I regarded the target domain as the test domain, which is incorrect. The target domain is also used during training and should therefore also be used to load embeddings. Both the source domain training data and the target domain training data should be used to retrieve word embeddings from an embedding file. This should (greatly) reduce the number of words that are not recognized during the validation and testing phase.

6.2.2 Hardware limitations

The other limitation is the hardware that was used. Due to memory limits I was restricted to a vocabulary of $10,000$ words for the medical embeddings. Stanford GloVe has $400,000$ embeddings in their standard file. This allows for a very high retrieval rate. Since it is common practice to freeze the embedding layer when using pretrained word embeddings, a high retrieval rate directly results in better model performance. Words in the training vocabulary, that are not in the pretrained word embeddings, are randomly initialized and remain so. So, the less words that are recognized from the training vocabulary, the more randomly initialized frozen embeddings are used. Those frozen embeddings are not informative for the model: in this case, randomly initialised embeddings that are updated (learned) during training, perform better. This was the case in my second DANN experiment.

6.2.3 Word embedding data selection

Apart from task-specific annotated data, the general unannotated data used to train GloVe on can also be improved. One way to do this, is by more careful selection of input data. For English, this data curation can be done through PubMed. For Dutch, such biomedical data are not as easily available, but one could think of using Wikipedia pages on a large number of medical terms. In the future this could be done within hospitals itself, when the reports are structured. Quality reports would allow for relatively easy sentence segmentation, making much more data available for models like GloVe. Things that should be kept in mind here are the minimum count of words, the vocabulary of the data (i.e. the range of words), the writing style; does it reflect the domain?

6.2.4 Additional features (character embeddings)

Additional features could be used, like capital letters, the number of numbers in a word, the length of a word, etc. Another type of additional feature is the use of character embeddings. Many state-of-the-art models use character embeddings next to word embeddings in NLP models. Those additional features are usually represented in manually given numbers and concatenated with the feature extractor output. Especially for out-of-vocabulary words, this could add a lot of value. For example, in the biomedical domain, it is not uncommon for special characters to be in words. CD29, p95vav, aml1 and many more such tokens exist in the GENIA (biomedical) dataset. The combination of characters could be quite indicative for the POS type here, even if words are out-of-vocabulary.

6.2.5 Mapping of (unknown) words

The last point of discussion is the mapping of unknown tokens. In language models and task-specific architectures it is normal to map unknown tokens from the target domain to an 'unknown' token with its own embedding. Each unknown token in the test data is mapped to the same vector representing the <UNK> token. Reducing the number of unknown tokens, by careful preprocessing or by creating general classes of words, could improve model performance. For example, considering I used Twitter data, it would have been interesting to compare performance when converting internet links to an <URL> token and @-mentions to a <Mention> token.

Furthermore, it appeared that 3% of the vocabulary consisted of numbers; one could investigate the effect of mapping all numbers to a single number in the vocabulary. Or the effect of simply removing all numbers. Both measures would result in an increase of the retrieval rate (since the vocabulary becomes smaller).

6.3 Future work

Future work should narrow the scope of research. Developing word embeddings and an evaluation method in one research or thesis is too broad of a scope. Since there was not much work in this domain before this thesis, it was the only option. I suggest specific options for both the word embedding and domain adaptation paths. I also give a recommendation for physicians.

6.3.1 Developing Dutch medical embeddings

In the later stages of my thesis I learned about a Dutch medical dataset: the Erasmus Medical Centre (EMC) corpus. With this Dutch medical dataset available, it is possible to develop word embeddings on a much simpler downstream task. An existing classification or tagging model can be used, in combination with word embeddings, to obtain scores on this EMC dataset. If I would have known about this dataset earlier and had access to it, using the DANN as evaluation method would not have been necessary. A future project that aims to develop word embeddings for the Dutch medical (radiology) domain, could use this dataset.

While developing Dutch word embeddings, one should look into ways of cleaning large bulks of medical reports. Once there is a way to clean them, and obtain clean Dutch medical sentences, the importance of data input size to the word embedding algorithm can be made clearer. It would answer the question: do medical embeddings, with a higher retrieval rate than general SotA word embeddings, outperform the Stanford GloVe embeddings on the medical domain?

6.3.2 Using the DANN on the medical domain

Future research could also further investigate the DANN. It is important to mention the EMC dataset again here: instead of adapting from the general to the medical domain, one could adapt from the EMC data to the UMC data. Usually the language domain differs to such extent between hospitals, that language models are not interoperable between hospitals. Therefore, domain adaptation between hospitals would also be useful. The EMC data is annotated for negation, amongst others, which makes it even more relevant: negation labels would be very useful to obtain, according to UMC physicians.

The concept of the DANN has already been improved on multiple times. The current SotA in the field of adversarial domain adaptation is promising. One example of a first addition could be the use of marginalized stacked denoising autoencoders (mSDA) (Chen et al., 2012), or one of its more recent improvements. This feature learns a new feature representation of the samples. It assumes noisy data entries and tries to reconstruct the denoised data entry. So, where the DANN attempts to find the underlying domain representation of two domains from noisy samples, this mSDA attempts to remove the noise from samples from both domains. Of course, many other (small) variations of the DANN could be looked into. In this category, of looking into the domain adversarial techniques itself, one could also experiment with the (negative) multiplication factor in the Gradient Reversal Layer.

Another path that could be looked into, is the use of additional features and a DANN with character encodings. Most SotA models use the latter: character encodings. This path is more data-focused and would include looking into preprocessing and data selection as well.

6.3.3 Annotation standards

The main recommendation for physicians, is to develop one or a few annotation standards. What does the medical world expect from Natural Language Processing? When selecting datasets for this thesis, it became clear how many different annotation styles there are and how little agreement there is between studies. A dataset annotated for negation can not be used to develop a model that should extract disease names and vice versa. Not only would standards make it easier to develop task-specific models, it would also make it easier to fine-tune existing language models in frameworks like Huggingface Transformers, SpaCy and Flair. Currently, those frameworks need to develop a new piece of code for every different tag set and file format. With datasets that are more alike, datasets can be combined, modified and used more easily. Even if the resources would be available now, there is no agreement amongst hospitals, let alone physicians on what annotations and file format should be used.

I will not go into detail on structured reports, as this is something the UMC is already working on. Next to this, once (better) data (sets) are available, the need for structured reports becomes less important.

6.4 Conclusion

This thesis made one of the first attempts to obtain useful information from noisy Dutch medical (domain-specific) reports. The thesis also discusses multiple challenges and approaches. Two challenges form a paradigm problem in Dutch medical NLP. To obtain or extract useful data from medical reports, either a solid annotated dataset for the specific task is required, or an advanced (language model) on the domain is required. Without one of them, it is hard to create the other, and neither was available for this thesis. Therefore, as a first step, GloVe embeddings are developed for the English medical domain. The second step is to show its performance when adapting from the general to the medical domain, by using a DANN. For this reason, a DANN for sequence labelling was developed. Once this working is confirmed, the same approach could be used for the Dutch medical domain. The DANN performance was best when using State-of-the-Art embeddings. This is most likely due to the low retrieval rate of the medical embeddings. The general English embeddings have a

vocabulary of 400.000 embeddings, whereas the mixed and medical embeddings only contained 5.000 to 10.000 embeddings. The DANN even performed better learning the embeddings during training, than when using the pretrained medical and mixed embeddings. However, given the large discrepancy in retrieval rate, the mixed and medical embeddings could still prove useful with a larger vocabulary size. Furthermore, with comparable performance, pretrained embeddings are still useful due to a lower training time. Future development of word embeddings should make use of better hardware. Finally, this development could make use of a simpler model than the DANN, by making use of the EMC dataset annotated for negation.

In summary, this thesis shows the current state of the Dutch medical NLP and gives a clear outline of the remaining challenges. Based on the acquired information in this thesis, two main research paths open up. The first being the development of Dutch GloVe embeddings, which should take into account the considerations of this thesis. The second research path could look into domain adaptation (using the DANN), between different medical domains.

Bibliography

Z. Afzal, E. Pons, N. Kang, M. C. Sturkenboom, M. J. Schuemie, and J. A. Kors. Contextd: an algorithm to identify contextual properties of medical terms in a dutch clinical corpus. *BMC bioinformatics*, 15(1):1–12, 2014.

A. Akbik, T. Bergmann, D. Blythe, K. Rasul, S. Schweter, and R. Vollgraf. Flair: An easy-to-use framework for state-of-the-art nlp. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59, 2019.

F. Alam, S. Joty, and M. Imran. Domain adaptation with adversarial training and graph embeddings. *arXiv preprint arXiv:1805.05151*, 2018.

S. Ben-David, J. Blitzer, K. Crammer, F. Pereira, et al. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19:137, 2007.

J. Blitzer, M. Dredze, and F. Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pages 440–447, 2007.

K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan. Domain separation networks. *Advances in neural information processing systems*, 29:343–351, 2016.

T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

W. W. Chapman, D. Hilert, S. Velupillai, M. Kvist, M. Skeppstedt, B. E. Chapman, M. Conway, M. Tharp, D. L. Mowery, and L. Deleger. Extending the negex lexicon for multiple languages. *Studies in health technology and informatics*, 192:677, 2013.

M. Chen, Z. Xu, K. Weinberger, and F. Sha. Marginalized denoising autoencoders for domain adaptation. *arXiv preprint arXiv:1206.4683*, 2012.

R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, 2011.

R. Cornet, A. Van Eldik, and N. De Keizer. Inventory of tools for dutch clinical language processing. In *MIE*, pages 245–249, 2012.

W. de Vries, A. van Cranenburgh, A. Bisazza, T. Caselli, G. van Noord, and M. Nissim. Bertje: A dutch bert model. *arXiv preprint arXiv:1912.09582*, 2019.

S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407, 1990.

L. Derczynski, K. Bontcheva, and I. Roberts. Broad twitter corpus: A diverse named entity recognition resource. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1169–1179, 2016.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

X. Ding, B. Cai, T. Liu, and Q. Shi. Domain adaptation via tree kernel based maximum mean discrepancy for user consumption intention identification. In *IJCAI*, pages 4026–4032, 2018.

X. Ding, Q. Shi, B. Cai, T. Liu, Y. Zhao, and Q. Ye. Learning multi-domain adversarial neural networks for text classification. *IEEE Access*, 7:40323–40332, 2019.

R. I. Doğan, R. Leaman, and Z. Lu. Ncbi disease corpus: a resource for disease name recognition and concept normalization. *Journal of biomedical informatics*, 47:1–10, 2014.

W. N. Francis and H. Kucera. Brown corpus manual. *Letters to the Editor*, 5(2):7, 1979.

L. Fu, T. H. Nguyen, B. Min, and R. Grishman. Domain adaptation for relation extraction with domain adversarial neural network. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 425–429, 2017.

Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

M. Gardner, J. Grus, M. Neumann, O. Tafjord, P. Dasigi, N. F. Liu, M. Peters, M. Schmitz, and L. S. Zettlemoyer. Allennlp: A deep semantic natural language processing platform. 2017.

D. Gerz, I. Vulić, F. Hill, R. Reichart, and A. Korhonen. Simverb-3500: A large-scale evaluation set of verb similarity. *arXiv preprint arXiv:1608.00869*, 2016.

Z. Huang, W. Xu, and K. Yu. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*, 2015.

A. E. Johnson, T. J. Pollard, S. J. Berkowitz, N. R. Greenbaum, M. P. Lungren, C.-y. Deng, R. G. Mark, and S. Horng. Mimic-cxr, a de-identified publicly available database of chest radiographs with free-text reports. *Scientific data*, 6(1):1–8, 2019.

D. Jurafsky. *Speech & language processing*. Pearson Education India, 2000.

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, 2003.

Y.-B. Kim, K. Stratos, and D. Kim. Adversarial adaptation of synthetic or stale data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1297–1307, 2017.

J. A. Kors, S. Clematide, S. A. Akhondi, E. M. Van Mulligen, and D. Rebholz-Schuhmann. A multilingual gold-standard corpus for biomedical concept recognition: the mantra gsc. *Journal of the American Medical Informatics Association*, 22(5):948–956, 2015.

C. Kulkarni, W. Xu, A. Ritter, and R. Machiraju. An annotated corpus for machine reading of instructions in wet lab protocols. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, 2018.

G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*, 2016.

D. Lu, L. Neves, V. Carvalho, N. Zhang, and H. Ji. Visual attention model for name tagging in multimodal social media. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1990–1999, 2018.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

A. Naik and C. Rosé. Towards open domain event trigger identification using adversarial domain adaptation. *arXiv preprint arXiv:2005.11355*, 2020.

J. M. Nobel, E. M. Kok, and S. G. Robben. Redefining the structure of structured reporting in radiology. *Insights into imaging*, 11(1):10, 2020a.

J. M. Nobel, S. Puts, F. C. Bakers, S. G. Robben, and A. L. Dekker. Natural language processing in dutch free text radiology reports: Challenges in a small language area staging pulmonary oncology. *Journal of digital imaging*, pages 1–7, 2020b.

S. Pathak, J. van Rossen, O. Vijlbrief, J. Geerdink, C. Seifert, and M. van Keulen. Post-structuring radiology reports of breast cancer patients for clinical quality assurance. *IEEE/ACM transactions on computational biology and bioinformatics*, 17(6):1883–1894, 2019.

Q. Peng, C. Zheng, Y. Cai, T. Wang, H. Xie, and Q. Li. Unsupervised cross-domain named entity recognition using entity-aware adversarial training. *Neural Networks*, 138:68–77, 2021.

J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

E. Pons, L. M. Braun, M. M. Hunink, and J. A. Kors. Natural language processing in radiology: a systematic review. *Radiology*, 279(2):329–343, 2016.

A. Ramponi and B. Plank. Neural unsupervised domain adaptation in nlp—a survey. *arXiv preprint arXiv:2006.00632*, 2020.

L. A. Ramshaw and M. P. Marcus. Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer, 1999.

A. Rietzler, S. Stabinger, P. Opitz, and S. Engl. Adapt or get left behind: Domain adaptation through bert language model finetuning for aspect-target sentiment classification. *arXiv preprint arXiv:1908.11860*, 2019.

M. Sato, H. Manabe, H. Noji, and Y. Matsumoto. Adversarial training for cross-domain universal dependency parsing. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 71–79, 2017.

C. Shaoul. The westbury lab wikipedia corpus. *Edmonton, AB: University of Alberta*, 131, 2010.

J. Tabassum, S. Lee, W. Xu, and A. Ritter. WNUT-2020 Task 1 Overview: Extracting Entities and Relations from Wet Lab Protocols. In *Proceedings of EMNLP 2020 Workshop on Noisy User-generated Text (WNUT)*, 2020.

E. F. Tjong Kim Sang. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*, 2002. URL `https://www.aclweb.org/anthology/W02-2024`.

E. F. Tjong Kim Sang and F. De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147, 2003. URL `https://www.aclweb.org/anthology/W03-0419`.

Y. Tsuruoka, Y. Tateishi, J.-D. Kim, T. Ohta, J. McNaught, S. Ananiadou, and J. Tsujii. Developing a robust part-of-speech tagger for biomedical text. In *Panhellenic conference on informatics*, pages 382–392. Springer, 2005.

B. Wang, A. Wang, F. Chen, Y. Wang, and C.-C. J. Kuo. Evaluating word embedding models: Methods and experimental results. *APSIPA transactions on signal and information processing*, 8, 2019.

Westerbeek. Natural language processing for dutch medical language. 2015.

T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Chapter 7

Appendix

7.1 Implementation Details

7.1.1 GloVe implementation

The implementation of GloVe used for those experiments can be found on GitHub (reference to be added). Part of the code was written by teachers of the Utrecht University as assignment for the course Natural Language Processing. I completed the assignment and elaborated on it. The implementation is written in Python, using the Pytorch library. Since the medical domain would benefit a great deal from NLP tools that can extract Named Entities (diseases, body parts, etc.), the experiments are done on the Named Entity task. This task is done using a BiLSTM-CRF in the Flair framework. Some parameters were found to have no influence on the results and were therefore fixed during (most of) the experiments. The fixed parameters are the following: constant C (avoid sparsity): 0.001, x_max: 100 and α: 0.75; following the original paper, epochs: 25; for the sake of training time, embedding dimension: 100 and learning rate: 0.05.

One important shortcoming of the implementation is that it is not adjusted to accept batch size input. When building a larger GloVe vocabulary, (> 10.000), this is practically mandatory.

7.1.2 DANN implementation

For the implementation itself, several pieces of code were copied or used as building block. Initially the advanced Pytorch BiLSTM-CRF guide helped a lot with the basics. However, this code is written for single instance input. Especially the CRF part becomes a lot more complicated for batch input. Therefore I ended up using 'pytorch-crf', which is largely borrowed from the CRF module from AllenNLP (Gardner et al., 2017). The BiLSTM code is relatively standard PyTorch code, from the torch library (Collobert et al., 2011). The training for each experiment was done with some specific parameters. First of all, early stopping was used with a patience of 10, unless otherwise specified. This means that the validation loss was allowed to worsen for 10 epochs, after which the training was stopped. Each time the model obtained a better validation score, the model was saved. This optimal model is used for testing and obtaining the inference scores. An SGD optimizer was used with a learning rate scheduler. The scheduler had a patience of 5 and used validation loss in its step

function. Next to this, the value of the Gradient Reversal Layer was fixed on -1. Furthermore, the loss function for both the task and domain classification was negative log likelihood. During training, a confusion matrix is constructed, giving access to metrics like recall, precision and F1-score. The reported recall, precision and F1-score are macro-averages, since there exists a class imbalance in all datasets. The 'benefit' of this approach is that all classes contribute equally to the reported scores. However, it also means that some classes can have very poor performances despite the relatively high macro score.

7.2 Useful resources (hyperlinks)

1. Flair framework: contains models and datasets
2. Huggingface Transformers: model hub, containing hundreds of models. Huggingface.co
3. GENIA POS: Github GENIA link.
4. PubMed: allows for downloaded up to $10,000$ abstracts at once. This is very useful for training word embeddings on (English) medical data. It is even possible to select based on keywords. With minimal preprocessing it is possible to obtain the plain text files: first split the data based on newlines, then filter the entries with over 100 words. Finally, remove the entries starting with 'Author', since some of those are also above 100 words.
5. Niderhoff: Github link.
6. WikiExtractor: PyPI link
7. Word senses & dictionaries: as of December 2020, this website provides word senses and dictionaries for many languages. My search for NLP tools for this thesis ended around December 2020, which is why I have not used or mentioned this. If this contains medical terms as well, it could prove very useful. Kaikki.org.
8. AI FAQ: in case you forgot anything on related to machine learning: faqs.org
9. Microsoft Research Open Data: msropendata.com

7.3 Additional analyses

(See next pages)

POS-tag	Precision	Recall	Number of labels	Times assigned
NNP	0.01	0.44	361	17184
VBZ	0.89	0.63	6647	4700
JJ	0.66	0.35	36266	19333
NN	0.65	0.86	107445	142658
TO	1.00	0.96	6382	6112
VB	0.60	0.84	4711	6572
.	1.00	1.00	14058	14050
CD	0.77	0.63	6062	4921
DT	0.93	0.88	29904	28428
VBD	0.82	0.75	9475	8579
IN	0.97	0.94	49448	47786
PRP	0.96	0.59	2863	1746
NNS	0.84	0.35	25818	10845
VBP	0.93	0.49	5571	2952
MD	0.97	0.94	1708	1662
VBN	0.79	0.57	11806	8552
POS	0.78	1.00	121	155
JJR	0.72	0.52	630	459
”	0.96	1.00	106	110
RB	0.94	0.54	10684	6173
,	1.00	1.00	14802	14803
FW	0.38	0.00	1509	8
CC	1.00	0.91	13861	12667
WDT	0.95	0.77	2083	1684
(	1.00	0.99	4123	4071
)	1.00	0.99	4140	4091
:	0.99	0.99	352	352
PRP$	0.97	0.81	1181	977
RBR	0.88	0.27	243	74
VBG	0.44	0.49	4072	4565
EX	0.99	0.70	141	99
WP	0.98	0.93	42	40
WRB	1.00	0.71	427	306
SYM	0.01	1.00	1	117
RP	0.52	0.64	39	48
RBS	0.95	0.46	82	40
PDT	1.00	0.58	36	21
”	1.00	0.05	38	2
LS	-	0.00	70	0
JJS	0.31	0.84	218	591
WP$	1.00	0.89	56	50

Table 7.1: Precision and recall scores on the target set (GENIA POS) experiment 2 with randomly initialized word embeddings.

POS-tag	Precision	Recall	True Labels	Assigned
NNP	0.01	0.52	361	30983
VBZ	0.91	0.61	6647	4463
JJ	0.60	0.34	36266	20325
NN	0.65	0.79	107445	131279
TO	1.00	0.96	6382	6106
VB	0.63	0.81	4711	6097
.	1.00	1.00	14058	14050
CD	0.79	0.60	6062	4610
DT	0.91	0.88	29904	28934
VBD	0.77	0.75	9475	9263
IN	0.95	0.93	49448	48090
PRP	0.96	0.59	2863	1743
NNS	0.81	0.27	25818	8649
VBP	0.96	0.44	5571	2590
MD	0.96	0.90	1708	1599
VBN	0.73	0.58	11806	9348
POS	0.80	1.00	121	151
JJR	0.66	0.41	630	390
”	0.98	1.00	106	108
RB	0.93	0.52	10684	5989
,	1.00	1.00	14802	14803
FW	0.00	0.00	1509	1
CC	1.00	0.90	13861	12545
WDT	0.96	0.78	2083	1703
(	1.00	0.99	4124	4069
)	1.00	0.99	4140	4084
:	0.99	0.99	352	351
PRP$	0.99	0.80	1181	957
RBR	0.83	0.20	243	58
VBG	0.52	0.42	4072	3343
EX	0.99	0.70	141	99
WP	1.00	0.93	42	39
WRB	1.00	0.70	427	300
SYM	0.01	1.00	1	71
RP	0.53	0.59	39	43
RBS	0.88	0.51	82	48
PDT	0.96	0.64	36	24
”	-	0.00	38	0
LS	-	0.00	70	0
JJS	0.59	0.67	218	247
WP$	1.00	0.55	56	31

Table 7.2: Precision and recall scores on the target set (GENIA POS) experiment 2 with medical word embeddings.

POS-tag	Precision	Recall	True	Assigned
NNP	0.00	0.97	361	85573
VBZ	0.91	0.71	6647	5158
JJ	0.58	0.56	36266	35199
NN	0.81	0.42	107445	56249
TO	1.00	0.96	6382	6110
VB	0.80	0.79	4711	4665
.	1.00	1.00	14058	14045
CD	0.57	0.72	6062	7590
DT	0.96	0.87	29904	27332
VBD	0.77	0.83	9475	10181
IN	0.97	0.94	49448	48083
PRP	1.00	0.59	2863	1688
NNS	0.95	0.48	25818	13024
VBP	0.95	0.63	5571	3689
MD	0.99	0.94	1708	1611
VBN	0.81	0.63	11806	9143
POS	0.78	1.00	121	156
JJR	0.93	0.28	630	188
”	1.00	1.00	106	106
RB	0.96	0.57	10684	6372
,	1.00	1.00	14802	14803
FW	1.00	0.00	1509	1
CC	1.00	0.90	13861	12553
WDT	0.98	0.83	2083	1762
(	1.00	1.00	4124	4116
)	1.00	0.99	4140	4083
:	0.99	0.99	352	353
PRP$	0.99	0.81	1181	967
RBR	0.69	0.54	243	189
VBG	0.93	0.40	4072	1728
EX	0.99	0.70	141	99
WP	0.97	0.88	42	38
WRB	1.00	0.68	427	292
SYM	0.01	1.00	1	134
RP	0.69	0.62	39	35
RBS	0.91	0.49	82	44
PDT	1.00	0.31	36	11
”	1.00	0.03	38	1
LS	-	0.00	70	0
JJS	0.85	0.71	218	183
WP$	1.00	0.52	56	29

Table 7.3: Precision and recall scores on the target set (GENIA POS) experiment 2 with SotA GloVe word embeddings.