An evaluation of BERT for a Span-based Approach for Jointly Predicting Entities, Coreference Clusters and Relations Between Entities

ULME WENNBERG
An evaluation of BERT for a Span-based Approach for Jointly Predicting Entities, Coreference Clusters and Relations Between Entities

ULME WENNBERG

Master in Computer Science
Date: October 7, 2019
Supervisor: Gustav Eje Henter
Examiner: Jonas Beskow
School of Electrical Engineering and Computer Science
Host company: Paul G. Allen School of Computer Science, University of Washington
Swedish title: En utvärdering av BERT för en span-baserad metod för att samtidigt identifiera och klassificera entiteter, koreferenskluster samt relationer i text
Abstract

This degree project examines and evaluates the performance of various ways of improving contextualization of text span representations within a general multi-task learning framework for named entity recognition, coreference resolution and relation extraction. A span-based approach is used in which all possible text spans are enumerated, iteratively refined and finally scored. This work examines which ways of contextualizing the span representations are beneficial when using the text embedder BERT. Furthermore, I evaluate to what degree graph propagations can be used together with BERT to enhance performance further, and observe F1-score improvements over previous work. The architecture sets new state-of-the-art results on four datasets from different domains - SciERC, ACE2005, GENIA and WLPC. Qualitative examples are provided to highlight model behaviour and reasons for the improvements are discussed.
Sammanfattning

Foreword

This degree project was carried out at the Paul G. Allen School of Computer Science, in Hannaneh Hajishirzi’s lab at University of Washington, as part of my Master of Science degree in Machine Learning at KTH Royal Institute of Technology. I was supervised by Gustav Eje Henter from KTH Royal Institute of Technology, with whom I had weekly meetings over Skype. I also had weekly meetings with Hannaneh Hajishirzi and other PhD students in the lab. This degree project is strongly connected to ongoing research in the field, and I am incredibly grateful to have had one supervisor from each university who have provided me with insights and valuable feedback.
Acknowledgements

This degree report constitutes my final assignment here at KTH Royal Institute of Technology. Looking back, it has been a fun and incredibly interesting journey that I cannot fathom is already about to end. There are many people who have helped me along the way, who I would like to express my gratitude towards and my appreciation for. Some of them are mentioned in the following.

Firstly, I am extremely grateful for having been supervised by Gustav Henter (KTH Royal Institute of Technology), who has been incredibly helpful, thoughtful and most accommodating throughout the process with everything from giving insightful hands-on feedback to more fundamentally helping me understand and see concepts and aspects of machine learning from new perspectives.

I would also like to thank my supervisor at the University of Washington, Hannaneh Hajishirzi, as well as the PhD student David Wadden (University of Washington) and recently graduated PhD Yi Luan (Google AI Language) with whom I have had many fruitful discussions. These brilliant people have provided me with not only helpful advice and valuable feedback but also the encouragement needed to take me through my journey.

Furthermore, I would like to thank Daniel del Castillo Iglesias for the thoughtful opposition along with the kind and helpful feedback.

I would also like to thank everyone that has helped me learn what I know today - I would like to especially thank everyone in my team from KTH Fantom with whom I participated in the Conversational AI competition Amazon Alexa Prize 2018. I want to thank you all for making my encounter with NLP research such an interesting and joyful experience. It would not have
been the same without you.
Contents

1 Introduction .................................. 1
   1.1 Introduction .................................. 1
   1.2 Research Question ............................... 3
   1.3 Objective ..................................... 4
   1.4 Specified Problem Definition .................... 4
   1.5 Examination Method .............................. 5
   1.6 Initial Hypothesis ............................... 5
   1.7 Evaluation ................................... 5
   1.8 Societal Aspects ................................. 6
   1.9 Sustainability and Ethics ....................... 6
       1.9.1 Ethical Implications ....................... 6
       1.9.2 Sustainability ............................. 7

2 Background ..................................... 8
   2.1 Artificial Intelligence (AI) ....................... 8
       2.1.1 Narrow Artificial Intelligence ............... 9
       2.1.2 Artificial General Intelligence ............. 9
   2.2 Machine Learning ................................ 10
       2.2.1 Artificial Neural Networks .................. 11
       2.2.2 Feedforward Neural Network ................. 11
       2.2.3 Recurrent Neural Network ................... 12
       2.2.4 Long Short-Term Memory (LSTM) Neural Network 12
       2.2.5 Training an Artificial Neural Network ......... 13
   2.3 Natural Language Processing (NLP) ............... 16
       2.3.1 Language Model .............................. 18
       2.3.2 Statistical Language Model .................. 18
       2.3.3 Word Embeddings ........................... 19
3 Method

3.1 Model

3.1.1 Span Embedding Module

3.1.2 Coreference Propagation

3.1.3 Relation Propagation

3.1.4 Named Entity Recognition (NER)

3.1.5 Coreference Resolution

3.1.6 Relation Extraction

3.1.7 Loss Function

3.1.8 Pruning

3.2 Implementation Details

3.2.1 Finetuned BERT

3.2.2 BERT + LSTM

3.3 Datasets

3.3.1 SciERC

3.3.2 ACE 2005

3.3.3 GENIA

3.3.4 WLPC
Chapter 1

Introduction

In this chapter I provide a brief introduction to the motivation behind tackling the different problems of named entity recognition, coreference resolution and relation extraction. I further state my research question that the degree project is based around. I then move on to discuss the objective of the report - to provide concrete learnings that can be shared with the scientific community - and discuss this further. Afterwards, I move on to dive deeper into what challenges this project entails, how it will be examined, and what was my initial hypothesis before starting the project. I end the chapter by briefly stating the evaluation criteria for the experiments, as well as discussing societal aspects as well as sustainability and ethics relating to the project.

This chapter is meant to provide the reader with an overview of what to expect to find in the project report. This chapter assumes that the reader has some familiarity with key concepts. For a more thorough explanation of many of the key concepts I advise the reader to move on to chapter 2, which provides more background information and an overview of the field.

1.1 Introduction

This section provides an introduction to information extraction as well as the motivation behind tackling the different problems of named entity recogni-
tion, coreference resolution and relation extraction together in the way presented in this report.

**Information Extraction** Information Extraction (IE) is the task of extracting structured information from unstructured text documents. This is typically achieved through detecting and classifying text spans, which are continuous sequences of words belonging to the text document.

**Multi-Task Learning** Three types of information extraction task are: entity recognition - the task of assigning a label to a span, coreference resolution - the task of group words referring to the same entity to the same coreference cluster, and relation extraction - the task of classifying the directed relation between two text spans. Since these three methods all depend on and excel with improved quality of the semantic representations of spans, there is reason to believe that predicting these tasks jointly may boost performance in each of the tasks.

**Named Entity Recognition** Named entity recognition (NER) has previously mainly been handled using sequential BIO-based models. While these models are successful in many areas, they suffer from the fact that they cannot assign a word to more than one named entity. This means that these kinds of architectures perform poorly on overlapping entities. A way to tackle this limitation is to use span based approaches, such as the one presented in this project report. These types of approaches considers all possible spans and therefore does not suffer from the aforementioned drawback.

**Coreference Resolution** Coreference resolution was first tackled using an end-to-end neural model by Lee et al. [1], where the authors generate span embeddings for all possible spans in the text document and use aggressive pruning to improve computational complexity. I use a similar approach for representing the spans and for generating the coreference resolution predictions.
**Relation Extraction** The relation extraction task is formulated as a multi-class classification task in order to predict the directed relation between each pair of text spans. Pruning of unpromising spans is used for computational efficiency.

**Pipelined Approaches and Cascading Errors** Pipeline style approaches have been commonly used when attempting the above tasks in the past. These approaches typically work by first predicting NER-tags and then choosing the most probable entities as inputs to the prediction of coreference resolution and relation extraction prediction modules. A positive aspect of pipelined approaches is that they make it possible to see what NER predictions were made, to better understand the relations that were predicted based on this. However, while pipelined approaches have the benefit of generating interpretable intermediate results, they suffer from the drawback that there is significant information loss in reducing the probability distribution over the NER-label predictions to a one-hot encoding of its most likely entry. This introduces so called cascading errors, where the errors in one module are cascaded throughout the network and contaminate all downstream results, which is one of the main motivations for using a neural architecture for joint end-to-end multi-task prediction. Newer models in this area have focused on joint inference of these tasks and tend often to rely on a shared LSTM layer that aims to generate more generalizable representations that are used in all tasks. The framework used in this project report is completely neural and it does not rely on any external syntactical tools or linguistic features, effectively avoiding cascading errors.

### 1.2 Research Question

In this report, I work on a general framework for incorporating information about named entities, relation extraction and coreference resolution through shared span representations, in order to improve performance across four datasets from different domains. I use text embeddings from BERT in order to get deep contextualized word representations that make use of global context. I evaluate using BERT and other way of contextualizing span embeddings in order to generate better representations. I also examine how this
affects performance through providing and analyzing quantitative and qualitative results. The research question addressed in this thesis work is:

"How can contextualized text embeddings by BERT be used in multi-task joint prediction of NER, relation extraction, as well as coreference resolution as an auxiliary task?"

### 1.3 Objective

The objective of this degree project is to pursue research in the field of information extraction and multi-task learning more generally while gaining a better understanding of how BERT and transformer-architectures work, and how usage of them can enhance performance. The project will focus on accumulating understanding of how adding BERT affects the tasks named entity recognition (NER) and relation extraction (RE), while using coreference resolution (CR) as an auxiliary task, and sharing this with the research community. The aim and ultimate goal of this degree project is to write down concrete learnings that can be shared with the scientific community.

### 1.4 Specified Problem Definition

The projects entail challenges such as what the best way is to use the encoded BERT-features to maximize the performance of the network, how one should go about finetuning BERT, if an LSTM is needed to gather contextualized information from the word embeddings or if finetuning BERT remove this need, if BERTs embeddings make the use of advanced propagation methods such as coreference propagation and relation propagation redundant, as well as which tasks are helped the most and the least by switching to BERT embeddings. It will also be interesting to see if any tasks will experience a decline in F1-score when moving over from ELMo to BERT.
1.5 Examination Method

The research question in Section 1.2 is examined by implementing the relevant models, training these using the default splits of training, validation and test data for each of the different datasets, and then using standard machine learning approaches for optimizing the loss functions on the training dataset (explicitly) and validation dataset (implicitly). I will thereafter use the model configuration that performs best on the validation dataset, and report its performance on the test dataset in order to compare to the performance of previous state-of-the-art approaches. This procedure is in accordance with the procedure that was used to achieve the previous state-of-the-art results on each of these datasets. I also do ablation studies where I investigate the effects of using different model configurations together with BERT and how this affects the performance on the validation dataset.

1.6 Initial Hypothesis

I hypothesize that using BERT will improve the results on most tasks, and I believe that it is quite likely that I will set new state-of-the-art (SOTA) results on some of these tasks. I also hypothesize that the performance will improve to different degrees on different tasks, and that the rate of the improvements might be possible to relate to the way in which the text embedders BERT and ELMo are trained. It is also possible that using BERT makes the span-based multi-task approaches obsolete, as it is quite possible that the BERT embeddings are so good at capturing and representing complex multi-word dependencies that there is no real improvement from jointly predicting these features, as they might already be implicitly encoded in the BERT embeddings.

1.7 Evaluation

I evaluate the models using F1-scores on several well-studied datasets that are respected by the scientific community. These include ACE2005, Genia, Wet Lab Protocol Corpus, and SciERC.
1.8 Societal Aspects

Developing advanced machine learning algorithms for this type of information extraction tasks could potentially make it feasible to build systems for performing mass-scale information extraction that are based on vastly more data than any human could realistically process. Successfully accomplishing this would significantly impact the process of organizing and structuring text in the future. A long term vision would be to organize the text on the internet and convert it into a structured representation, which could in turn be used in order to more efficiently answer the questions we have about the world around us.

1.9 Sustainability and Ethics

In this section I discuss ethical implications of the work in this degree project and of the field of artificial intelligence in general, as well as the potential implications on sustainability and the environment.

1.9.1 Ethical Implications

Historically, the field of natural language processing and information extraction have required close human supervision due to available hardware not meeting the computation requirements and available software not being sufficiently efficient at pattern recognition and reasoning with uncertainty. I believe that computers today are on the brink of reaching sufficient performance to start out-competing humans in many of the more repetitive and monotonous tasks in this area, which may have huge consequences for the future of jobs and introduce many ethical considerations. As an example, natural language processing-based algorithms are currently being introduced in law firms to read through old cases and automatically pick out relevant prior rulings. The impact of improved artificial intelligence algorithms has been voiced more in recent years, with advancements in this field making it possible to accomplish things that were previously impossible to achieve without human intervention. Given how fast this transition is happening, it is very hard to predict what effects this will have on the future
work situation. While some predict that this will lead to a mass eradication of jobs, others argue that it will free up people to work on more fulfilling tasks with the help of these tools. I am currently a proponent of the first argument, as I believe that the number of people that will experience layoffs in current jobs will significantly outnumber the minority with hireable skills that will thrive in the economy of the future. I believe that this will be a very difficult problem for society to solve.

1.9.2 Sustainability

The models and algorithms presented in this degree project are currently very computationally expensive to train. However, inference is relatively computationally cheap. This means that usage of these algorithms have a high fixed-cost during train-time, but potentially with a low relative operational cost if they are to be adopted in society to a sufficient extent. Low operational costs mean that the average cost and hereby the environmental impact of using these technologies could still end up being relatively low, if sufficient economies of scale are utilized. More research in the field of computationally efficient natural language processing-based information extraction systems is an interesting field that I hope will get increased funding and interest in the future.
Chapter 2

Background

This chapter introduces and explains relevant background concepts that are useful in order to understand the method and the context for the report. I start off by explaining bigger-picture concepts such as Artificial Intelligence and Machine Learning and then move into Natural Language Processing in order to scope this degree project and put it into relation to common approaches in current research.

2.1 Artificial Intelligence (AI)

Artificial Intelligence (AI) is commonly used to refer to machines that can mimic human intelligence processes, as well as the pursuit of building computer systems with these capabilities. The human processes to mimic includes learning about systems as well as being able to reason about them and update one's beliefs. Some applications-based subfields of Artificial Intelligence include Computer Vision, Natural Language Processing and Speech Recognition. Artificial Intelligence is commonly divided into two major branches - Narrow Artificial Intelligence and Artificial General Intelligence.
2.1.1 Narrow Artificial Intelligence

Narrow Artificial Intelligence, (Narrow AI), is a subset of artificial intelligence that deals with developing methods for solving a certain task or problem. While oftentimes excelling at a predefined task, the proposed algorithms are often handcrafted in certain ways for the task and domain at hand, which makes this type of artificial intelligence hard to extend to other domains. One example of this is the fact that while excelling at playing the game of chess, Deep Blue (Murray Campbell and Hsu [2]), the first computer to achieve superhuman performance in the game of chess, still lacked capabilities to outperform a completely randomized algorithm in four-in-a-row. This is because core elements of the intelligence are being coded into the program, without enough flexibility to be able to adapt to new situations and domains. This project primarily deals with Narrow AI.

Multi-Task Learning

One way of working to generalize slightly from more narrow intelligence is through training a model on more and more tasks - ideally from different domains - while keeping the model architecture as simple and as general as possible. By using this approach, the model designer is incented to build a framework that generates more generalizable representations which are in a better position to generalize to previously unseen samples as well as to new tasks.

2.1.2 Artificial General Intelligence

Artificial General Intelligence, in contrast to Narrow Artificial Intelligence, deals with how to write and design programs that are transferable not only between domains, but also between completely different tasks requiring different capabilities. This pursuit aims to solve intelligence on a more fundamental level, to the point where an Artificial General Intelligence system would be capable enough to find optimal behaviour in new domains without human guidance. While Artificial General Intelligence at the level of human performance is a long-term research goal for many people in the field, it is not currently present in any current machines or algorithms.
2.2 Machine Learning

Machine learning is the scientific discipline of studying statistical models and algorithms that learn to detect trends in data and can estimate complex functions that explain the patterns. Machine learning is typically viewed as a subset of artificial intelligence, nearly exclusively Narrow Artificial Intelligence, where a function is iteratively refined through update formulas based on minimizing the discrepancy between its predictions and the labels in the data. Machine learning models explain rules in the training data, and use various techniques in order to improve generalizability of the model to unseen data points. Such techniques include adding noise to data samples through dropout (Srivastava et al. 2014), putting soft constraints on network parameters through regularization, and using a validation set of data not seen by the model to continually estimate the model’s performance on. These techniques improve generalizability by incenting the model to learn smoother functions in input space that generalize well over a broader range of input data samples.

Machine learning is closely related to statistics, where statistical methods are also used in order to provide insights from the patterns in the data. The biggest difference between machine learning and statistics is that statistics builds on concrete assumptions about the variables - assumptions that allow the analyst to calculate uncertainty bounds under these assumptions. In machine learning, however, these assumptions are treated as less important than the actual performance of the model. This means that harder-to-analyze (often strongly non-linear) models are accepted - as long as they offer superior performance at explaining new unseen data. This stands in stark contrast to the rigid assumptions about normal distributed variables and independence that constitute crucial underlying assumptions in many statistical models.

Due to the nature of the problem formulation, machine learning systems usually formulate a loss function - a differentiable proxy for the error of the predictions - and optimizes it using non-linear optimization techniques such as iterative update rules like gradient descent.
2.2.1 Artificial Neural Networks

Artificial Neural Networks are a class of models that aim to process and analyze complex data and perform function approximation as well as multi-class classification. In multi-class classification problems, these models can learn to perform tasks such as classifying what fruit is in an image or what word class a certain word in a sentence belongs to. Artificial neural networks have been adopted for a wide range of different tasks in different fields - including language understanding, image recognition, speech recognition and fraud detection just to mention a few. First an artificial neural network architecture is chosen for the problem at hand, after which the network’s trainable parameters are updated iteratively until convergence.

In recent years, artificial neural networks have gained wide-spread recognition for their capabilities in solving more complex problems of pattern recognition that were previously unsolvable for computers.

Artificial neural networks are loosely based on biological neural networks, using some of the underlying principles in order to achieve desired behaviours; artificial neural networks were initially proposed as a way to mimic the way the human brain tackles problems, by introducing small computing units - neurons - that perform simple computations that are optimized in a way as to create emergent phenomena that gives rise to intelligent behaviour. Perhaps the most notable strategy is the use of threshold logic, using multiple nodes to sum their contributions together, and then using a threshold to decide whether to pass the signal on. This idea clearly stems from the biological neural networks and the integration of presynaptic input. Beyond this, neural network researchers have split into two camps: the one that focused on enhancing the understanding of the biological processes that take place in the brain, and the one that aims solely to perfect information processing.

2.2.2 Feedforward Neural Network

Feedforward neural networks are a class of artificial neural network architectures, that deals with processing an input vector and predicting a desired output. Feedforward neural networks are the simplest form of artificial neu-
eral networks. The simplest feedforward neural network layer work by applying a linear transform of the input vector, followed by a non-linear transformation (the so-called activation function) of the output. In feedforward neural networks, the information always moves in one direction - from the input vector to the predicted output.

2.2.3 Recurrent Neural Network

Recurrent Neural Networks are a subclass of Artificial Neural Networks, that deals with sequence problems by sequentially looping through the data. Recurrent Neural Networks differ from Feedforward Neural Networks in that Recurrent Neural Networks have recurrent connections, which means that they keep track of an internal state that gets passed along as input together with the next sample in the sequence. This allows the same structure to iteratively loop through the data and keep track of context through the internal state that gets updated as time passes.

2.2.4 Long Short-Term Memory (LSTM) Neural Network

The long short-term memory network is a type of recurrent neural network that was first proposed by Hochreiter and Schmidhuber [3], as a way to deal with some of the challenges of remembering context over different time scales that the conventional recurrent neural network faces. The long short-term memory neural networks improve performance of conventional recurrent neural networks by introducing additional parameters for distinguishing what parts of the current state should be kept in memory and what should be updated. It does this by introducing three distinct gates into each LSTM unit - these are the input gate, the output gate and the forget gate. These gates help with the vanishing gradient problem in different ways, allowing the long short-term memory network unit to adjust its output depending on previous input over longer time scales and learn long time-dependencies easier. The LSTM units are also typically more stable to optimize than vanilla RNNs. The long short-term memory networks are Turing complete (Siegelmann and Sontag [4]) meaning that they can in theory learn to model any arbitrary signal function over time. Long short-term memory networks
are one of the most commonly network architectures - due to its expressiveness and due to it being easy to use. The LSTM-cell is governed by the following equations.

\[
f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \tag{2.1}
\]

\[
i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \tag{2.2}
\]

\[
o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \tag{2.3}
\]

\[
c_t = f_t \odot c_{t-1} + i_t \odot \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \tag{2.4}
\]

\[
h_t = o_t \odot \sigma_h(c_t) \tag{2.5}
\]

### 2.2.5 Training an Artificial Neural Network

Artificial neural networks are designed to learn arbitrary mappings from an input to an output vector. They achieve this through using inductive bias - by mathematically formulating rules for inducing the most suitable explanatory models from data. This is typically achieved by introducing a loss function, that quantifies a heuristic for the performance of the model on the training dataset, that is then minimized using non-linear optimization approaches. This is done iteratively for several epochs - iterations of going through every sample in the training data and updating the model accordingly. At set intervals, the trained model is evaluated on the validation dataset which is also sometimes referred to as the development dataset. As far as possible, the performance is here measured on the actual metric that one seeks to optimize. It is often hard to optimize directly on the metrics as these often times are not differentiable, which is why heuristics are necessary.
Loss Function

The loss function, $L$, is introduced as a differentiable heuristic for the metric that one seeks to optimize. The loss function needs to be differentiable in order for the model to be able to use gradient descent methods.

**Mean Squared Error Loss**  The most commonly used loss function in the regression case is the *mean squared error* loss, $L_{\text{MSE}}$, that is calculated as the mean of the squared errors between the predicted values and the true values. This is commonly used for real-valued regressor functions. In this example, I show the mean squared error-loss for $N$ samples, where $y_i$ is the true value of the $i$:th example, and $\hat{y}(x_i)$ is the predicted value based on the network input $x_i$.

$$L_{\text{MSE}} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}(x_i))^2 \quad (2.6)$$

**Categorical Cross Entropy Loss**  The most commonly used loss function in the classification case is the *categorical cross entropy* loss, $L_{\text{CE}}$, that is calculated as the negative sum of the probabilities of the logarithm of the probability mass that the model assigns to the true labels. This is used for classification problems. In this example, I show the categorical cross entropy-loss for $N$ samples, where $y_{\text{obs},i}$ is the true value of the $i$:th sample, and $\hat{y}_{\text{obs}}(x_i)$ is the predicted probability mass distribution based on the network input $x_i$. Here $I(y_{\text{obs},i} = k)$ is an indicator function that is equal to one if the $i$:th sample is observed to belong to the $k$:th class.

$$L_{\text{CE}} = - \sum_{i=1}^{N} \sum_{k=1}^{K} I(y_{\text{obs},i} = k) \log(\hat{y}_{\text{obs}}(x_i)) \quad (2.7)$$

Note that this is the same as maximizing the product of the probability mass assigned to the correct classes.

$$\arg \min_{\text{weights}} L_{\text{CE}} = \arg \max_{\text{weights}} \prod_{i=1}^{M} \hat{y}_{\text{obs}}(x_i) \quad (2.8)$$
This is the same as the maximum likelihood estimate if samples are assumed to be independent.

**Gradients**

After the appropriate loss function is chosen, each batch of data samples is predicted, using the so called *forward pass* of the artificial neural network. After this, the loss is computed and differentiated with respect to each of the weights in the neural network. This differentiation proceeds layer by layer, starting with the weights in the layer closest to the output. For the \( n \):th layer, the \( k \):th weight’s gradient is computed through the following equation, which is computed recursively by using the chain rule, where \( 1 \leq k \leq K_n \). This step is called the *backward pass* and its introduction was one of the catalyzing events in the field of machine learning, when it was first proposed by Werbos [5].

\[
\frac{\partial L}{\partial W^n_k} = \sum_{i=1}^{K_{n+1}} \left( \frac{\partial W^{n+1}_{i}}{\partial W^n_k} \cdot \frac{\partial L}{\partial W^{n+1}_{i}} \right) \tag{2.9}
\]

**Optimizer**

After all the gradient calculations have been performed, the weights are updated (adjusted slightly) to move closer to a local minimum of the loss, preferably as quickly as possible. This procedure varies slightly depending on the kind of optimizer that is being used.

**Stochastic Gradient Descent**  The simplest optimizer - stochastic gradient descent, or SGD for short - generally updated the weight parameters using the following equation, where the learning rate \( l_{\text{rate}} \) controls the magnitude of the gradient updates and typically might vary with the layer number \( n \), and the number of epochs \( e \) that have been complete. That is \( l_{\text{rate}} = l_{\text{rate}}(n, e) = \hat{l}_{\text{rate}}(n) \cdot \tilde{l}_{\text{rate}}(e) \).

\[
W^n_k = W^n_k - l_{\text{rate}} \cdot \frac{\partial L}{\partial W^n_k} \tag{2.10}
\]
The learning rate is tuned in such a way as to reach the local minimum of the loss function as soon as possible. If the learning rate is too high, however, this scenario causes too big updates for each weight which leads to chaotic behaviour (Lau [6], Zulkifli [7]).

**Adam**  Adam is a more complex optimizer first proposed by Diederik P. Kingma [8]. It involves estimating the first and second momentum of the gradients through saving and averaging consecutive estimates of these. These estimates are then used in the iterative update formulas in order to improve stability of the model while enabling rapid convergence on a local optimum. This is the optimizer that will be used in the report. Here, $0 < \beta_1 < 1$ and $0 < \beta_2 < 1$ are factors that indicate how much of the running averages should be used in the calculations, in relation to the new estimates. Also, $\epsilon > 0$ is a small non-zero constant that is introduced in order to avoid division by zero.

\[
m_{W}^{t+1} = \beta_1 m_{W}^{t} + (1 - \beta_1) \nabla_{W} \text{Loss}(t) \quad (2.11)
\]

\[
v_{W}^{t+1} = \beta_2 v_{W}^{t} + (1 - \beta_2)(\nabla_{W} \text{Loss}(t))^2 \quad (2.12)
\]

\[
\hat{m}_{W} = \frac{m_{W}^{t+1}}{1 - (\beta_1)^{t+1}} \quad (2.13)
\]

\[
\hat{v}_{W} = \frac{v_{W}^{t+1}}{1 - (\beta_2)^{t+1}} \quad (2.14)
\]

\[
W^{t+1} = W^{t} - \eta \frac{\hat{m}_{W}}{\sqrt{\hat{v}_{W}} + \epsilon} \quad (2.15)
\]

### 2.3 Natural Language Processing (NLP)

Natural language processing (NLP) is a subfield of artificial intelligence that comprises using computational engineering techniques for analysis and generation of natural language. While computers are excelling in certain
areas of machine learning, there are still significant unsolved challenges in understanding natural language.

NLP is commonly modelled as a sequence problem - given a sequence of words or characters, build a model that predicts the next word, or the sentiment of the sentence, or the named entities in the sentence or similar.

**Comparison to Computer Vision**  Another subfield of artificial intelligence where machine learning algorithms are currently excelling is in the area of computer vision. In the following paragraph, I will argue for what properties NLP has that makes it increasingly difficult to model effectively.

One aspect that makes natural language processing more difficult is the fact that natural language is a form of communication - and as such - is governed by the needs of individuals to share information and learn from each others. Cutlip and Center [9] present the seven C’s of communication: Completeness, Conciseness, Consideration, Concreteness, Courtesy, Clearness and Correctness. While principles of clearness and correctness encourage individuals to focus on the most important things, increasing signal over noise, there are also difficulties; the principle of conciseness imply that information that does not immediately help the point to be made - or that is obvious to both parts communicating - should not be mentioned. While leaving out information that is shared by both parties tends to be beneficial for the flow of the conversation at hand, however, it is not always the case that this information is known by an outside observer. This makes it hard for a natural language processing model that does not know facts about the world, to understand the full extent of what is happening in a text - without possessing necessary background information.

These principles for text to convey as much information as possible with the least amount of effort for the involved parties has certain drawbacks; the sentences below exemplify why this makes learning natural language hard - while the first sentence includes more information, it is more appropriate in most cases to go with the second option.

*I went out for a run in my running shoes.*

*I went out for a run.*

Because of this, there are difficulties associated with learning the fact that
most people wear shoes when they are out running, unless implicitly building a more complex model of the world in which it learns that most people wear shoes unless otherwise specified.

This is not the case in computer vision. In computer vision, the model is typically fed an image as input, which in the case of photographs shows a two-dimensional projection of the three-dimensional world. Due to this fact, in a photo, the challenge is quite different in that there often is an abundance of information that the model is faced with and the challenge is instead to train up the ability to attend to the right information. These projections and challenges in computer vision, however, seem way simpler to express in mathematical notation that the rules that project their NLP-counterparts - the rules that project the three-dimensional world into a sentence.

2.3.1 Language Model

A language model refers to a probability distribution over all possible word sequences. This is one of the most important concepts within natural language processing, as this ability of a model is crucial for many applications such as detecting spelling mistakes, inducing grammar and generating language.

2.3.2 Statistical Language Model

Since the 1950s, statistical language models have gone from being criticized by Chomsky [10] and others, to being a fundamental aspect of every language modelling system.

An implicit aspect that makes language modeling an important task when using artificial neural network-based models is the fact that in order to correctly model the probability of a word sequence, the network is incented to learn useful representations of semantics and context awareness in order to successfully classify that "He eats pasta." is more likely to occur in a sentence than "He eats shoes.", even though both are grammatically correct.
These are often times calculated using sequential models - that is for each word \( w_i \) for \( i \geq 1 \), I calculate \( p(w_1, ..., w_i) \) recursively using:

\[
p(w_1, ..., w_i) = p(w_i | w_1, ..., w_{i-1}) p(w_1, ..., w_{i-1})
\]

(2.17)

Modern approaches commonly use a neural network to estimate \( p(w_i | w_1, ..., w_{i-1}) \). Earlier so-called \( n \)-gram approaches rely on simplifying independence assumptions, that assume that only the previous \( n-1 \) closest words contribute to the probability. Using this assumption, a strong baseline is achieved through the formula:

\[
p(w_i | w_{i-1}, w_{i-2}, ..., w_{i-(n-1)}) = \frac{\text{count}(w_i, w_{i-1}, w_{i-2}, ..., w_{i-(n-1)})}{\text{count}(w_{i-1}, w_{i-2}, ..., w_{i-(n-1)})}
\]

(2.18)

Here, the function count is used to count the number of occurrences of a certain \( n \)-gram or \((n-1)\)-gram in the training dataset, where minor adjustments are commonly made to avoid division by zero. This is the maximum likelihood-estimate if we assume the word choices to be categorically distributed.

### 2.3.3 Word Embeddings

The realization that statistical language models need to capture semantics to effectively model the probability distribution was used by Mikolov et al. [11], who proposed training \( d \)-dimensional \( (d = 300 \text{ is most common}) \) word vectors in a language modeling task in order to gain semantic representations for each word in the English language, by using a two-layer neural network, and use each word in a large corpus as input to predict their nearby words. The authors use the hidden representations in the first layer as the vector of values representing the word itself in semantic space. By formulating this training objective and representing the vectors in this way, two words
that occur interchangeably should be trained to predict similar words around them - and therefore have similar hidden representations when training converges. The model hereby trains up a mapping from each word to a vector in d-dimensional space, that is trained to give a semantic representation of the word’s meaning.

### 2.3.4 Contextualized Word Embeddings

In the last years, contextualized word embeddings have been proposed as a way to enable modelling the fact that many words have different meanings depending on in which context they occur. Recent models achieve this by letting each word embedding be a function not only of the word itself, but of the word conditioned on its context (Peters et al. [12], Jacob Devlin and Toutanova [13]). This is motivated by the fact that almost every word can have slightly different meanings depending on its context; the word *elephant* means different things in the phrases *the elephant at the zoo* and *the elephant in the room*. There are also the case that a certain word can have completely different meanings - *bank* refers to completely different things when referring to the *river bank* and in the phrase *the bank was robbed*.

### 2.3.5 Embeddings from Language Models (ELMo)

Peters et al. [12] introduce the system ELMo, where the authors propose using deep contextualized representations for embedding the words in the document. ELMo word representations are the first to condition word representations on the entire input sequence. This allows each word to be represented depending on its context, and improves the state-of-the-art on a range of significant NLP benchmarks.

ELMo reaches a performance boost by contextualizing each word representation by conditioning on within-sentence context. The original ELMo model was trained on 5.5B tokens dataset, of which 1.9B come from Wikipedia and the rest come from the monolingual news crawl data from WMT 2008-2012. ELMo was trained on randomly shuffled sentences. Because of this, ELMo learns not to propagate context over sentence boundaries. This means that even though the word representations can in theory be contextualized
over sentence-boundaries, this does not happen in practice and so there is no point in contextualizing the word representations conditioned on more than one sentence.

2.3.6 Transformer

In the paper "Attention Is All You Need" by Vaswani et al. [14], the authors propose a new architecture for contextually embedding words, the Transformer, which achieves similar performance to state-of-the-art RNN architectures. The Transformer is a neural architecture that alternately uses various versions of self-attention between all words in a sequence and feedforward neural networks. The combination of these two paradigms allows for obtaining contextually dependent representations for each individual word token in a text. This differs from LSTMs and other RNN architectures that instead encode the entire context up to each point in a fixed-size vector. Furthermore, the transformers differs from previous RNN approaches in that it is highly parallelizable due to its calculations being non-sequential in nature. This is beneficial for modern machine learning with graphical processing unit (GPU)-based calculations, that are made to perform many computations in parallel. Because of this, the transformer can encode the entire sentence at once. Due to this fact, it is called bidirectional. It is, however, more accurate to call it nondirectional.

2.3.7 Bidirectional Encoder Representations (BERT)

Jacob Devlin and Toutanova [13] further improve on the Transformer architecture by using it as a building block when introducing the neural architecture: Bidirectional Encoder Representations (BERT). In the original paper, the authors show that BERT outperforms state-of-the-art on 11 highly competitive major NLP tasks. BERT reaches this performance through introducing novel training objectives that the model trains on, as well as a novel model architecture in order to generate these embeddings.

BERT introduces the following novel training objectives in order to capture more rich semantics and meaning within the sentences:
• **Masked Language Model** During training, 15% of the words in the sentence are masked out. The model then gets the task of predicting the missing words. Doing this requires a high-quality understanding of both semantics and syntactic details.

• **Next Sentence Prediction** Another task that BERT is trained to perform is the following: given two sentences, predict whether the second sentence follows directly after the first in the text document that they occur. The training data is acquired in the following way. For each sentence, use the next sentence as the true sample (with 50% probability), or randomly pick a sentence from the document (with 50% probability) and use this as the negative sample. By doing this, the authors introduce a binary classification task where the model aims to correctly predict whether one sentence follows immediately after another.

### 2.3.8 SciBERT

SciBERT is a BERT-model that was proposed by Iz Beltagy and Lo [15] in order to provide a language model specifically trained for scientific domains. It is trained on scientific papers from different domains in the semanticscholar.org-corpus, which has 1.14M papers and 3.1B tokens. By doing this, the authors show enhanced performance on a number of natural language processing tasks on scientific text.

### 2.4 Information Extraction

In the last few years there has been a growing interest in developing methods for automatically extracting *structured information* from *unstructured text documents*, with the desire for improved performance of look-ups as well as enabling a wider range of search queries.
2.5 Beginning-Inside–Outside-Tagging

Beginning-Inside–Outside-Tagging (BIO-Tagging) is a commonly used tagging format for tagging tokens, that was proposed by Ramshaw and Marcus [16]. Beginning, Inside and Outside constitute three distinct states for each of the tags (which could be entity classes) and so the task becomes predicting the correct state sequence. Ergo, named entity recognition (Section 1.1) and similar tasks have historically mainly been viewed as sequence problems. While this is satisfactory in many normal domains where the entities consist of contiguous tokens that do not overlap with each other, this way of treating the entity recognition generally has the issue that overlapping spans (explained in Section 1.1) cannot be extracted. There have been recent efforts to tackle this by using more complex neural architectures and applying hypergraph-based representations on top of sequence labelling systems (Wang and Lu [17]).

2.6 Span Based Methods

The model in this report addresses the challenge with overlapping spans in an alternative way by considering all possible spans as candidate entities independent of other overlapping entities and thereby avoiding the problem altogether. Span-based methods typically extract all possible text spans - all unbroken subsequences of the words in the sequence - and then process and classify them independently. This means that the sentence *He eats pasta* would be split into 1: *He*, 2: *eats*, 3: *pasta*, 4: *He eats*, 5: *eats pasta*, 6: *He eats pasta*. This makes it possible to extract features from overlapping spans. This is especially useful in domains where overlapping entities are common and constitute a significant portion of the total number of spans. This approach allows *Seattle Seahawks* to be classified as a sports team, while simultaneously predicting that *Seattle* is a city. These examples of overlapping and nested entities cannot be efficiently modeled using the BIO-tagging paradigm, since these cases break the assumption that one word is only part of at most one entity. In span based-methods, all the words in each span are typically combined and represented as one joint embedding.
2.7 Graph Propagations

One way to improve the span representations is through propagation of global information from other spans within the document. This is achieved through supervised methods by training propagation modules such as relation propagation and coreference propagation, as described below. These types of graph propagations are then used to update span embeddings using information from nearby span embeddings within the same document in a dynamically constructed knowledge graph. Using the graph propagations allows for hierarchical contextualization. It works by first contextualizing the word embeddings using BERT, followed by creating the span embeddings based on this, in order to finally contextualize the spans based on all other spans in the document.

2.7.1 Coreference Propagation

Coreference Propagation was first introduced by Lee, He, and Zettlemoyer [18]. In this paper, the authors propose propagating information between different entities through an attention mechanism over each word’s antecedents - the previous words in the document refering to the same entity. The idea behind coreference propagation is that spans belonging to the same coreference cluster - meaning that they refer to the same entity - should share certain features and therefore get more similar representations in certain dimensions. This is enforced by having the coreference propagation module propagate information between spans depending on how likely the model estimates it is that the spans are coreferences to each others - meaning that they refer to the the same entity. This improves performance by allowing information to flow between entities on a document-level, hereby conditioning the predictions not only on the current span, but on all the spans in the document.

\[
u_C^t(i) = \sum_{j \in B_C(i)} P_C^t(i,j) g_j^t
\]  

\[
P_C^t(i,j) = \frac{\exp(V_C^t(i,j))}{\sum_{j' \in B_C(i)} \exp(V_C^t(i,j'))}
\]  

(2.19)  

(2.20)
Here $P_C(i, j)$ constitutes a softmax distribution computation, where the update embedding $u_C(i)$ is computed as a weighted mean of the spans, where each weight corresponds to the probability of the $j$:th span being predicted to be an antecedent of the $i$:th span.

### 2.7.2 Relation Propagation

Relation propagation was first introduced by Luan et al. [19], as a way to allow the relation extraction module to help propagating information between the entities that are related to each others in different ways. The update rule is described in the following, where $V_R^t(i, j)$ is a vector of the predicted score-distribution for the relation between the $i$:th and the $j$:th span, $A_R$ is a linear transformation matrix and $\odot$ refers to element-wise multiplication. The function $f$ is the ReLU-function that is used in order to remove the effect of spans that are unlikely to be related to each others.

The idea behind relation propagation is that spans that are related to each other in a certain way are likely to share certain attributes. By propagating relation information to a span about what other spans it is related to, this can help generate a better representation of the span which leads to more accurate predictions. This improves performance by allowing information to flow between different entities on a document-level, hereby conditioning the predictions not only on the current span, but on all the spans in the document.

$$u_R^t(i) = \sum_{j \in B_R(i)} f(V_R^t(i, j)) A_R \odot g_j^t$$  \hspace{1cm} (2.21)

### 2.7.3 Propagation Procedure

By propagating relation and coreference information, the model performance can be improved by conditioning the output on global context. This propagation procedure works by alternately performing the following two update steps.

- First, compute an update embedding $u_X^t(i)$, for the $i$:th span, where $X \in \{C, R\}$ refers to coreference propagation or relation propaga-
tion. This is done by summing up the pair-wise contributions for all the other spans.

- Second, compute a gating function that determines to what extent the information in the span embedding should be updated. This is achieved through performing a linear transform of the update embedding $u^t_X(i)$ and the original vector $g^t_i$, followed by an element-wise sigmoid function $\sigma(x)$. The new embedding is then computed as a weighted average of the update embedding and the span embedding, according to the weights in the gating vector.

\[
\sigma(x) = \frac{1}{1 + e^{-x}} \quad (2.22)
\]

\[
f^t_X(i) = \sigma(W_X(g^t_i, u^t_X(i))) \quad (2.23)
\]

\[
g^{t+1}_i = f^t_X(i) \odot g^t_i + (1 - f^t_X(i)) \odot u^t_X(i) \quad (2.24)
\]

This procedure is performed for $T$ steps, where $1 \leq t \leq T$. By propagating the graph information in this way, each span embedding becomes contextualized conditioned on the predictions of other spans in the document, which enables the model to generate more informative intermediate representations that ultimately improve model performance.

### 2.8 Metrics

In statistical analysis with classification problems, the F1-score is often used to measure a model’s performance. The F1-score is useful because it considers both the precision - the fraction of predictions that actually belong to the correct class, and recall - the fraction of samples in a certain class that are correctly captured by the model. The F1-score is calculated as the product of precision and recall, divided by the arithmetic mean of the two. The F1-score therefore lies in the interval between 0 and 1. However, it is often implicitly reported as a percentage, that lies in the interval between 0 and 100.
For each class, the precision and recall get calculated based on the true positives, false positives, true negatives and false negatives. True positives, false positives, true negatives and false negatives are calculated as follows for class i, where N is the number of samples. Here \( I(y_{\text{pred}}^k = i, x_{\text{true}}^k = j) \) is an indicator function that is equal to one if the \( k \):th sample is predicted to belong to class \( i \), and it belongs to class \( j \). This function is equal to zero for all other configurations of \( y_{\text{pred}}^k \) and \( x_{\text{true}}^k \).

\[
\text{true positives}_{i} = \sum_{k=1}^{N} I(y_{\text{pred}}^k = i, y_{\text{true}}^k = i) \quad (2.25)
\]

\[
\text{false positives}_{i} = \sum_{k=1}^{N} I(y_{\text{pred}}^k = i, y_{\text{true}}^k \neq i) \quad (2.26)
\]

\[
\text{true negatives}_{i} = \sum_{k=1}^{N} I(y_{\text{pred}}^k \neq i, y_{\text{true}}^k \neq i) \quad (2.27)
\]

\[
\text{false negatives}_{i} = \sum_{k=1}^{N} I(y_{\text{pred}}^k \neq i, y_{\text{true}}^k = i) \quad (2.28)
\]

\[
\text{recall}_{i} = \frac{\text{true positives}_{i}}{\text{true positives}_{i} + \text{false negatives}_{i}} \quad (2.29)
\]

\[
\text{precision}_{i} = \frac{\text{true positives}_{i}}{\text{true positives}_{i} + \text{false positives}_{i}} \quad (2.30)
\]

\[
F1_{i} = 2 \cdot \frac{\text{recall}_{i} \cdot \text{precision}_{i}}{\text{recall}_{i} + \text{precision}_{i}} \quad (2.31)
\]

When calculating F1-scores for the multi-class setting, there are two ways of combining these scores for each of the classes into a general score. These are macro-averaging and micro-averaging. In macro-averaging, the F1-score is calculated as the arithmetic mean of the individual F1-scores for each of the K classes.
\[
F1_{\text{macro}} = \frac{1}{K} \sum_{j=1}^{K} F1_j \quad (2.32)
\]

\[
\text{recall}_{\text{micro}} = \frac{\sum_{j}^{K} \text{true positives}_j}{\sum_{i=1}^{K} (\text{true positives}_i + \text{false negatives}_i)} \quad (2.33)
\]

\[
\text{precision}_{\text{micro}} = \frac{\sum_{j=1}^{K} \text{true positives}_j}{\sum_{i=1}^{K} (\text{true positives}_i + \text{false positives}_i)} \quad (2.34)
\]

\[
F1_{\text{micro}} = 2 \cdot \frac{\text{recall}_{\text{micro}} \cdot \text{precision}_{\text{micro}}}{\text{recall}_{\text{micro}} + \text{precision}_{\text{micro}}} \quad (2.35)
\]

In this report, I will be using micro-averaged F1-scores, which is what is most commonly used within the domain. There are, however, some works that instead report macro-averaged F1-scores, which makes their results noncomparable.

### 2.9 Software

The code for this report was written in AllenNLP (Gardner et al. [20]), which is an open-source NLP research library, built on PyTorch (Paszke et al. [21]). I had access to a code-base of previous work in TensorFlow (Martin Abadi et al. [22]), that was re-implemented by me and another student in AllenNLP. This was done in parallel with the literature review in order to perform active learning of the material. We aim to release our model in the AllenNLP framework. The code that was run on Graphics Processing Units, (GPUs), using CUDA - a parallel computing platform by NVIDIA for general computing on GPUs.

#### 2.9.1 AllenNLP

AllenNLP is based on using config files in order to specify the hyperparameters of each module as well as which module to use. This then constructs
PyTorch objects of the specific kinds. AllenNLP further implements specific modules and architectures with pretrained weights that are commonly used in NLP. While AllenNLP had support for many of the functionalities that I needed throughout the project, there were some functionalities that we needed to implement. Some of these changes have now been pushed to the AllenNLP open-source library.

2.10 Hardware

I used 3-15 GPUs in parallel throughout the project, where each experiment typically used one GPU for about 5-10 hours. The majority of the computations for these experiments were performed using a EVGA GeForce GTX TITAN X with 3 GPUs at the Paul G. Allen School of Computer Science. For the more intense period when many configurations needed to be trained and tested in parallel, I also used 3 Virtual Machines with 4 NVIDIA Tesla K80 GPUs each on Google Cloud.

2.11 Limitations

While I had good access to computational resources, I would still argue that computing times and the 8 hours that often were needed to get feedback for each experiments was a bottle-neck. To combat this, I worked on optimizing the code in several directions. From using tensors to parallelizing computations when possible to using different kinds of pruning to avoid unnecessary computations for unpromising spans. I also experimented with using newly proposed optimizers such as AdaBound (Luo et al. [23]), that is known for combining fast training with better generalizability from the training set to the validation set and the test set.

2.12 Pipelined Approaches

the Pipeline style systems have commonly been used for coreference resolution and relation extraction tasks in the past. These consider NER-tags
being known during training, which means that these tags either have to be annotated beforehand by human annotators, or by relying on the predictions of third-party NER-task classifiers. While relying on the former is unfeasible in many domains due to annotation costs, relying on the latter introduces cascading errors from the errors in the entity predictions that propagate through-out the model. This behaviour is undesirable and motivates the use of end-to-end neural models. Lee et al. [1] proposed the first end-to-end neural model for coreference. Luan et al. [24] built on this by incorporating this model as a module in their multi-task learning framework, predicting entities, coreference resolution and relations in a joint end-to-end fashion, relying on a shared LSTM layer that generates shared span embeddings that are used in all three tasks.
Chapter 3

Method

This chapter explains the methodology used in the project. I start by describing the model used in the project and then move on to lay out the different experiments that were performed. I also examine the datasets and explain the format of the data further.

3.1 Model

In this section I introduce the model. Given an input document, the model generates a span embedding for each span in the text, upon which the output of all modules are based. The output of the model is threefold; the main module branches out into one prediction module for each of its three tasks - named entity recognition, coreference resolution and relation extraction. Each of these tasks are described in detail in the following.
3.1.1 Span Embedding Module

Given an input document $D$, with $T$ words, all spans up to span width $k$ are extracted ($T$ spans of length 1, $T - 1$ spans of length 2, all the way up to $T - k + 1$ spans of length $k$), which gives us $N = \sum_{i=\max(T-k+1,1)}^{T} i$ unique spans. The span width indicates the number of words in the words sequence that constitute the span. Each span is embedded using an embedder, BERT, which is followed by a bidirectional-LSTM in some of the experiments in order to generate the span embeddings.

Span Representations

In order to correctly perform the prediction tasks in this project report, there are two general types of information that need to be extracted from the text. These are the vector representations of the context surrounding each span as well as the internal structure within it. These both types of information have been found in previous work to be well-encoded with the help of a bi-directional LSTM layer. Using such a layer, it has also been shown that concatenating the LSTM-computed embeddings of the first and last word in the span gives a robust embedding that is useful for downstream tasks. Intuitively, these endpoint embeddings contain information of surrounding context on both sides of the span as well as the span interior.
3.1.2 Coreference Propagation

After enumerating and representing each span, each span that survived the coreference module’s pruning is iteratively refined through the coreference propagation procedure. This procedure is described in detail in section 2.7.1 in the background, and generally works by propagating information between the span embeddings through an attention mechanism for the likely antecedents of each span.

3.1.3 Relation Propagation

After the coreference propagation is completed for the involved spans, each span that survived the relation extraction module’s pruning stage is iteratively refined through the use of relation propagation. This procedure is described in detail in section 2.7.2 in the background.

3.1.4 Named Entity Recognition (NER)

The named entity recognition task is treated as a multi-class prediction task, where each span is assigned to one of the NER-classes.

The categorical cross entropy loss function is computed over the annotated ground-truth entities for each span. This task feeds each of the span embeddings through a feedforward neural network, which is finally fed through a softmax layer in order to get a probability distribution over the different classes. No pruning is used in this model. I calculate the probability under the model of a certain span representing \( s_i \) a certain entity \( e \) in the NER-task using the following function, where each \( f_e \) is calculated using a feed-forward neural network.

\[
\Phi_e(e, s_i) = f_e(s_i)
\] (3.1)
3.1.5 Coreference Resolution

The model is based on the approach in Lee et al. [1]. The loss function used is the negative log marginal likelihood of annotated ground-truth antecedent spans for each span pair. Span mention scores $f_c$ are calculated for each span. The highest-scoring spans (that survive all the general and the coreference resolution pruning stages) are then combined in all pairwise combinations with order of appearance maintained. Each combination is then fed as a span pair to $g_c$. I calculate the score distribution for the coreference resolution-task using the following functions, where each $f_c$ and $g_c$ are calculated using feed-forward neural networks. $\Phi_c(s_i, s_j)$ approximates the score for the span $s_i$ constituting an antecedent to the span $s_j$.

$$\Phi_c(s_i, s_j) = f_c(s_i) + f_c(s_j) + g_c(s_i, s_j)$$  \hspace{1cm} (3.2)

3.1.6 Relation Extraction

The relation extraction task is treated as a multi-class prediction task, where each span pair is assigned to one of the relation-type classes $r'$. The loss function used is categorical cross entropy over the annotated ground-truth relations for each of the span pairs. Span mention scores $f_r$ are calculated for each span and the best spans (that survive all the general and the relation extraction pruning stages) are matched together pair-wise and fed as a span pair to $g_r$. I calculate the score distribution for the relation extraction-task using the following functions, where each $f_r$ and $g_r$ are calculated using feed-forward neural networks to estimate the score for each relation-type class $r'$.

$$\Phi_r(r', s_i, s_j) = f_r(r', s_i) + f_r(r', s_j) + g_r(r', s_i, s_j)$$  \hspace{1cm} (3.3)

3.1.7 Loss Function

The loss function in the multi-task setting is calculated by computing a weighted sum of the loss functions of the individual models, where the weights of the different loss terms constitute hyper-parameters of the model.
that are tuned for each specific task. This means that the weights are set to control the relative importance of each task, and that we can get single-task models as a special case of the multi-task model by setting the loss weights for two of the tasks to zero. This also means that the multi task model is a generalization of the single task model. The total loss $L_{\text{tot}}$ is given by the following equation, where $w_x$ and $L_x$ denotes the weight and the calculated loss for task $x$.

$$L_{\text{tot}} = w_e L_e + w_c L_c + w_r L_r$$  \hspace{1cm} (3.4)

### 3.1.8 Pruning

A difficulty with the approach to consider all possible spans is the fact that it is computationally expensive. In a document of length $T$ there are $\mathcal{O}(T^2)$ different spans, where each is then combined with every other span, generating a total of $\mathcal{O}(T^4)$ span pairs for both the coreference resolution task and the relation extraction task. Because of this, aggressive pruning is used in those two stages, to reduce complexity to $\mathcal{O}(T^2)$ and thereby speed up computations.

#### Maximum Span Width $k$

The first pruning strategy is to set a maximum span width $k$ - the number of words in the sequence that the span represents. This maximum width is set in order to improve computational complexity, using the fact that most entities only contain a few words. According to previous research by Luan et al. [24], only a negligible amount of entities are missed out when setting the maximum span width to $k = 8$, while reducing the number of individual spans to $\mathcal{O}(Tk)$.

#### Mention Score Pruner

There is also a second pruning strategy that is used in the coreference resolution as well as the relation extraction modules in order to minimize the
computational load and memory usage. Coreference resolution and relation extraction considers span pairs in order to predict whether or not the spans belong to the same coreference cluster and what relationship exists between them. This means that for a document of $O(Tk)$ spans, we will need to process $O(T^2k^2)$ span pairs. Because of this, the number of spans to be considered is further pruned to $\lambda T$, where $0 < \lambda < 1$, which reduces the total number of pairs to be processed to $O(\lambda^2T^2)$. The mention score pruner is trained jointly with the end-to-end model on optimizing the end-task. This means that the span-pruning module is trained together with the rest of the model and updated for each new batch.

3.2 Implementation Details

In this section, I provide implementation details for the different variations of the model considered in the experiments. There are two main variations of the model that I will discuss further - these are pretrained BERT, followed by contextualization through an LSTM, as well as finetuned BERT without an LSTM. Both variations are implemented in AllenNLP. Each of the feedforward neural networks in the task-specific layers of the network has 150 hidden dimensions, two hidden layers, uses 40% dropout during training and uses ReLU activation functions. I increase contextual information to the BERT embeddings by passing in a sliding window of the $L-1$ neighbouring sentences to the actual sentence. This means that for $L = 1$, I contextualize only the current sentence, whereas for $L = 3$, the sentence to the left and to the right of the target sentence are included when embedding it with BERT. $L$ is a hyper-parameter that is chosen independently for each problem. I do a grid-search over the hyper-parameters $L \in \{1, 3, 5\}$, as well as the number of times to perform the coreference propagation and the relation propagation. For these parameters, I do a grid-search over the values $c_p \in \{0, 1, 2\}$ and $r_p \in \{0, 1, 2\}$. All hyper-parameter tuning is performed on the validation dataset. The model with the hyper-parameter configuration that achieves the highest F1-score is then saved and evaluated once on the test dataset.
3.2.1 Finetuned BERT

In this variation, I finetune the entire architecture (including BERT) jointly on the end task. When doing this, I omit the LSTM layer and let the task specific layers of each module follow immediately after BERT. I use the Adam optimizer (Diederik P. Kingma [8]) with different learning rates for BERT and for the task specific layers - $5.0 \cdot 10^{-5}$ for BERT and $1.0 \cdot 10^{-3}$ for the task specific layers. For most of the datasets, I train the model for 200 000 batches. I perform linear warmup for the task specific layers for the first 20 000 batches, followed by linear decay for 180 000 batches, as well as linear warmup for the task specific layers for the first 40 000 batches, followed by linear decay for the following 160 000. Using a longer warmup period for the BERT parameters than for the rest of the network allows the task specific to change in the first epochs while affecting the BERT-embeddings minimally. I found that using different learning rates for the different parts of the network gave significant improvements over using the same learning rate for the entire network. I note that this scheme gave us an improvement of around 5 percentage-units absolute F1-score over finetuning everything jointly with one learning rate from the start. I also use the decay-on-plateau learning rate scheduler as well as early stopping, based on the F1-score on the validation dataset.

3.2.2 BERT + LSTM

In this variation, I use pretrained BERT for contextualizing the word representations. After this, the word embeddings are contextualized further by a bi-directional LSTM-layer. BERT is kept frozen during training and the LSTM is trained together with the task specific layers. This reduces the number of trainable parameters in the model from $\approx 115,000,000$ to $\approx 5,000,000$.

3.3 Datasets

I test the architectures on the four datasets SciERC, ACE 2005, GENIA, and WLPC. These are from different domains - the SciERC dataset contains sci-
Scientific research articles from the field of artificial intelligence, ACE 2005 contains a range of common domains such as news stories and online forums, GENIA contains a collection of biomedical abstracts and Wet Lab Protocol Corpus (WLPC) contains a collection of wet lab protocols. The class labels for each of the datasets are provided in the appendix.

<table>
<thead>
<tr>
<th></th>
<th>Domain</th>
<th>Documents</th>
<th>Entities</th>
<th>Relations</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACE 2005</td>
<td>News</td>
<td>511</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>SciERC</td>
<td>AI</td>
<td>500</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>GENIA</td>
<td>Biomed</td>
<td>1999</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>WLPC</td>
<td>Bio lab</td>
<td>622</td>
<td>18</td>
<td>13</td>
</tr>
</tbody>
</table>

### 3.3.1 SciERC

The SciERC corpus provides coverage of 500 artificial intelligence article abstracts and involves many scientific terms as well as scientifically oriented named entities and relations. SciERC has a small fraction of overlapping spans (words taking part in two or more spans that each constitute an entity), constituting less than 3% of all spans in the dataset. This dataset includes annotated entities, coreference resolution and relations which makes it suitable for the multi-task training. Please note that the coreference resolution task does not use category labels.

### 3.3.2 ACE 2005

The ACE 2005 corpus contains a range of common domains such as news stories and online forums and provides coverage of a wide range of events and activities. ACE 2005 also has a small fraction of overlapping spans, constituting less than 3% of all spans in the dataset. ACE 2005 does not include any coreference resolution annotations. Because of this, I train the coreference module on the Ontonotes dataset, in order to facilitate the use of coreference propagation.
3.3.3 GENIA

The GENIA corpus (Kim et al., 2003) provides coverage of 1999 biomedical abstracts. This corpus also includes many scientific terms and biological substances. GENIA has a substantial 24% of its named entity spans overlapping with another named entity span. Because of this, this is a dataset in which span-based approaches significantly outperform BIO-based approaches. The dataset includes entity annotations as well as coreferences between different entities. It does, however, not have any relation extraction annotations. Because of this, I did not use the relation extraction module for this task. Please note that the coreference resolution task does not use category labels.

3.3.4 WLPC

The Wet Lab Protocol Corpus provides coverage of 622 wet lab protocols. This dataset includes annotated entities and relations, but has no coreference resolution annotations. Because of this, I did not use the coreference resolution module for this task.

3.4 Experiments

In this section I present and describe the experiments I have run in more detail.

3.4.1 Ablation Studies

I evaluate the effects of the different kinds of propagations for each dataset by ablating (selectively removing) the different types of propagations and studying the effect on the validation dataset F1-score. I also compare the variations of the model - BERT+LSTM vs. finetuned BERT. I also explore the effect of conditioning each span embedding on more than one sentence of context through BERT, by varying, $L$, the number of sentences that are contextualized through BERT together.
3.4.2 In-Domain Pretraining

I evaluate the importance of in-domain pretraining for the datasets SciERC and GENIA. This is done by performing additional ablation studies for these datasets, where I compare the performance of both BERT and SciBERT with and without pretraining on data from the domain being studied.
Chapter 4

Results

In this chapter I present the results from the experiments. I have organized this chapter by dividing the results into two sections - quantitative results and qualitative results. In the quantitative results section, I compare model performance to current state-of-the-art. I also perform ablation studies in this section, to analyze what modules help and whether there are synergy effects occurring between them. In the same section, I also evaluate the impact of using in-domain pretraining for language models. In the qualitative results section, I highlight some specific examples and use them to provide deeper explanations for how the model and its modules work.

4.1 Quantitative Results

In this section I present the results for each of the numerical experiments. Each table presents the F1-scores - on the test set for the comparisons to other models, and on the validation set for the ablation studies. For each test, the best performing system’s result is marked in bold. The best system is then used for comparing against current state-of-the-art.
4.1.1 Comparison to Current State-of-the-Art

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Task</th>
<th>SOTA</th>
<th>This Model</th>
<th>Δ%</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACE 2005</td>
<td>Entity</td>
<td>88.4</td>
<td><strong>88.6</strong></td>
<td>1.7</td>
</tr>
<tr>
<td>ACE 2005</td>
<td>Relation</td>
<td>63.2</td>
<td><strong>63.4</strong></td>
<td>0.5</td>
</tr>
<tr>
<td>SciERC</td>
<td>Entity</td>
<td>65.2</td>
<td><strong>67.5</strong></td>
<td>6.6</td>
</tr>
<tr>
<td>SciERC</td>
<td>Relation</td>
<td>41.6</td>
<td><strong>48.4</strong></td>
<td>11.6</td>
</tr>
<tr>
<td>GENIA</td>
<td>Entity</td>
<td>76.2</td>
<td><strong>77.9</strong></td>
<td>7.1</td>
</tr>
<tr>
<td>WLPC</td>
<td>Entity</td>
<td>79.5</td>
<td><strong>79.7</strong></td>
<td>1.0</td>
</tr>
<tr>
<td>WLPC</td>
<td>Relation</td>
<td>64.1</td>
<td><strong>65.9</strong></td>
<td>5.0</td>
</tr>
</tbody>
</table>

Table 4.1: F1-scores on the test dataset for each of the tasks in the report, as well as comparison to and relative improvement Δ over current state-of-the-art.

The new model gives a big impact on the performance in the relation extraction task on SciERC; it outperforms previous state-of-the-art by 6.8 F1-points which constitutes an 11.6% relative improvement. This is partly due to the in-domain language model using SciBERT, that helps (See 4.5 for more details on this). The results also show that the propagations are beneficial even when moving over to using BERT.

It is also interesting to note that the performance boost that BERT gives is highly dependent on the domain of the document; while using pretrained BERT seems to have a limited effect on the named entity recognition task on the ACE 2005 dataset (0.2 F1-score improvement compared to current state-of-the-art), it has a much bigger impact on the named entity recognition task on SciERC. This might in part be due to the fact that the ACE 2005 dataset is more similar to the text domain in which ELMo and BERT are trained on, but might also be explainable with system theory and the fact that it is harder to improve an already well-functioning system - it is relatively harder to improve one F1-point at 80 F1-score than what it is to do it at 60 F1-score. It is also harder to improve the performance on a dataset such as ACE2005 where one faces a lot of other competing models. This is the reason why I report the relative improvement similar to how this is done by Peters et al. [12], calculated as follows:
\[ \Delta \%_{ACE\ 2005} = 1 - \frac{100 - 88.6}{100 - 88.4} \approx 1.7\% \]  

(4.1)

### 4.1.2 Ablation Studies

<table>
<thead>
<tr>
<th></th>
<th>ACE 2005</th>
<th>SciERC</th>
<th>GENIA</th>
<th>WLPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT+LSTM</td>
<td>85.8</td>
<td>69.9</td>
<td>78.4</td>
<td>78.9</td>
</tr>
<tr>
<td>+RelProp</td>
<td>85.7</td>
<td>70.5</td>
<td>-</td>
<td>78.7</td>
</tr>
<tr>
<td>+CorefProp</td>
<td>86.3</td>
<td><strong>72.0</strong></td>
<td>78.3</td>
<td>-</td>
</tr>
<tr>
<td>Finetuned BERT</td>
<td>87.3</td>
<td>70.5</td>
<td>78.3</td>
<td>78.5</td>
</tr>
<tr>
<td>+RelProp</td>
<td>86.7</td>
<td>71.1</td>
<td>-</td>
<td>78.8</td>
</tr>
<tr>
<td>+CorefProp</td>
<td><strong>87.5</strong></td>
<td>71.1</td>
<td><strong>79.5</strong></td>
<td>-</td>
</tr>
</tbody>
</table>

Table 4.2: F1-scores on the validation dataset for the NER-task for each of the datasets. Ablation studies for different hyper-parameter configurations.

<table>
<thead>
<tr>
<th></th>
<th>ACE 2005</th>
<th>SciERC</th>
<th>WLPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT+LSTM</td>
<td>60.6</td>
<td>40.3</td>
<td>65.1</td>
</tr>
<tr>
<td>+RelProp</td>
<td>61.9</td>
<td>41.1</td>
<td>65.3</td>
</tr>
<tr>
<td>+CorefProp</td>
<td>59.7</td>
<td>42.6</td>
<td>-</td>
</tr>
<tr>
<td>Finetuned BERT</td>
<td><strong>62.1</strong></td>
<td>44.3</td>
<td>65.4</td>
</tr>
<tr>
<td>+RelProp</td>
<td>62.0</td>
<td>43.0</td>
<td><strong>65.5</strong></td>
</tr>
<tr>
<td>+CorefProp</td>
<td>60.0</td>
<td><strong>45.3</strong></td>
<td>-</td>
</tr>
</tbody>
</table>

Table 4.3: F1-scores on the validation dataset for the Relation-task for each of the datasets. Ablation studies for different hyper-parameter configurations.
Table 4.4: F1-scores on the validation dataset for the ACE 2005 dataset Relation-task. Ablation studies for different hyper-parameter configurations.

Table 4.4 shows that adding more sentences of context helps the model to reach higher F1-scores. The results further indicate that contextualizing $L = 3$ sentences through BERT together tends to reach the best performance.

### 4.1.3 In-Domain Pretraining

<table>
<thead>
<tr>
<th></th>
<th>SciERC Entity</th>
<th>SciERC Relation</th>
<th>GENIA Entity</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT+LSTM</td>
<td>69.1</td>
<td>38.8</td>
<td>74.7</td>
</tr>
<tr>
<td>+RelProp</td>
<td>69.8</td>
<td>37.2</td>
<td>-</td>
</tr>
<tr>
<td>+CorefProp</td>
<td>69.8</td>
<td>40.6</td>
<td>75.4</td>
</tr>
<tr>
<td>Finetuned BERT</td>
<td>68.2</td>
<td>37.2</td>
<td>77.5</td>
</tr>
<tr>
<td>+RelProp</td>
<td>69.5</td>
<td>40.0</td>
<td>-</td>
</tr>
<tr>
<td>+CorefProp</td>
<td>69.1</td>
<td>41.9</td>
<td>78.4</td>
</tr>
<tr>
<td>SciBERT+LSTM</td>
<td>69.9</td>
<td>40.3</td>
<td>78.4</td>
</tr>
<tr>
<td>+RelProp</td>
<td>70.5</td>
<td>41.3</td>
<td>-</td>
</tr>
<tr>
<td>+CorefProp</td>
<td><strong>72.0</strong></td>
<td>42.6</td>
<td>78.1</td>
</tr>
<tr>
<td>Finetuned SciBERT</td>
<td>70.5</td>
<td>44.3</td>
<td>78.3</td>
</tr>
<tr>
<td>+RelProp</td>
<td>71.1</td>
<td>43.0</td>
<td>-</td>
</tr>
<tr>
<td>+CorefProp</td>
<td>71.1</td>
<td><strong>45.3</strong></td>
<td><strong>79.5</strong></td>
</tr>
</tbody>
</table>

Table 4.5: F1-scores on the validation dataset. Evaluating the effect of in-domain pretraining: SciBERT vs. BERT.
In this table, SciBERT consistently outperforms BERT for the scientific domain datasets SciERC and GENIA. There is no clear winner between the different model versions: BERT+LSTM and finetuned BERT. Instead, each variation of the model outperforms the other in certain situations.

4.2 Qualitative Results

In this section I highlight model behaviour by visualizing two examples of how coreference propagation leads to more global awareness. I use two documents from the dataset GENIA.

4.2.1 Coreference Propagation Leading to Correct Entity Mention Prediction

I visualize how the coreference propagation module propagates information to a span, leading to the correct entity prediction. For this, I am using a document from the dataset GENIA. This is shown in figure 4.1 and figure 4.2.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td><strong>v-erbA</strong> overexpression is required to extinguish c-erbA function...</td>
</tr>
<tr>
<td>2:</td>
<td>The <strong>v-erbA oncoprotein</strong> represents a retrovirus-transduced...</td>
</tr>
<tr>
<td>3:</td>
<td>It contributes to virus-induced erythroleukemia...</td>
</tr>
<tr>
<td>4:</td>
<td>Here, we show that <strong>v-erbA</strong> and c-erbA bind directly to sequences...</td>
</tr>
</tbody>
</table>

Figure 4.1: When classifying the entity mention for the span **v-erbA** (green) in the fourth sentence, the model can share information from its predicted coreference antecedents **v-erbA** and **v-erbA oncoprotein** (red).
Figure 4.2: Coreference propagation attention weights for the likely antecedents to the span v-erbA. The biggest update comes from the span v-erbA oncoprotein.

By correctly propagating information from the representation of the span v-erbA oncoprotein in the second sentence, the model can use this information to update the representation of v-erbA in sentence 4, so as to include the necessary information to predict it as the entity mention "protein". This propagation is done on a document level, enabling the model to propagate context over arbitrary large distances.

### 4.2.2 Propagating Contextual Information by Coreference Propagation

I visualize how the coreference propagation module propagates information to a span, giving rise to more context awareness. For this, I use a document from the GENIA dataset. This is shown in figure 4.3.
Human T cell transcription factor GATA-3

A family of transcriptional activating proteins, the GATA factors, has been shown to bind to a consensus motif through a highly conserved C4 zinc finger DNA binding domain. One member of this multigene family, GATA-3, is most abundantly expressed in T lymphocytes, a cellular target for human immunodeficiency virus type 1 (HIV-1) infection and replication.

I visualize the antecedent distribution for the coreference module in figure 4.3. This shows how the coreference propagation module helps to propagate information between the span representations in the document, depending on how probable it is that two spans belong to the same coreference cluster as each others. This is done for each span that has survived the pruning, and each span is updated with information from all other surviving spans.
Chapter 5

Discussion

In this chapter I analyze the results presented in the chapter above and discuss these results. Lastly, I suggest future work in the area that I believe is likely to help the field move forward.

5.1 Discussion

The experiments verify the finding from Luan et al. [24] and Luan et al. [19] that using multi-task learning and propagating global context can help generate better representations that improve the performance on each individual task. Furthermore, I establish that BERT’s contextual word embeddings outperforms ELMo’s on these datasets, by comparing to and improving the current state-of-the-art across four different datasets from different domains.

When comparing results to previous state-of-the-art systems, it is interesting to note that BERT can effectively carry embedding context between sentences. Even though ELMo and BERT are both trained to give deep contextualized word embeddings, they are trained in different ways; while ELMo is trained on shuffled sentences which incents it not to carry context across sentences, BERT is trained on coherent text documents which incents it to model and propagate relevant context. This means that in the following sentence - "General Electric said the Postal Service contacted the company. It..." - it is unlikely that ELMo will be able to predict that It refers
back to *the company - General Electric* - which is why the load of keeping this context is instead put onto the LSTM. BERT, on the other hand, handles this propagation information between word embeddings over sentence boundaries automatically and therefore facilitates further specialization of the LSTM or the finetuning to instead focus on capturing more complex linguistic phenomena.

Coreference propagation proves to be helpful for the performance in the entity recognition task and adding it show improvements across all datasets for this task.

Relation propagation improves relation extraction performance for the BERT + LSTM, but does not boost the performance of the finetuned model. I hypothesize that this is because the self-attention mechanism in BERT (when trained in a supervised way on the end task) can learn to effectively capture most of the information in the relation extraction labels when finetuning it. This could cause the positive effect of then adding more explanatory power to the model to be out-weighted by the negative effect of adding more trainable parameters, which ultimately leads to the relation propagation module in its current form not enhancing performance.

The ablations of the contextualization hyperparameter $L$ shows that the BERT-model benefits from wider context windows in most cases. The model typically achieves the best performance with an $L = 3$ sentence window. It is interesting to note that more context does not seem to improve performance beyond this point. This is interesting since it intuitively seems that more context should always give rise to a superior performance. One reason for why this is not the case might be due to that all annotated relations between entities are within-sentence. This means that providing additional surrounding context might be confusing the model by diluting the within-sentence information that is represented in the contextualized word-representations, based on which the spans are constructed.

I also verify and quantify previous findings that in-domain language models significantly outperform more general language models across the board, by comparing model performance using the text embedders BERT and SciBERT on the datasets SciERC and GENIA.

Furthermore, I show that adding coreference propagation and relation propagation to the BERT+LSTM variation of the model reaches competitive per-
formance to finetuned BERT; this variant of the model performs competi-
tively with the finetuned BERT-model - and in some cases outperforms it. I note that this offers a lighter-weight alternative to finetuning BERT, using almost two orders of magnitude fewer trainable network parameters.

5.2 Future Work

While the empirical results are strong, important future work is what new ar-
chitectures are made possible by using BERT as the embedder. The [CLS]-
tag that BERT uses extracts a sentence embedding and it is one of the promis-
ing things to look further into incorporating into the span embedding pipeline in order to enhance the model by providing additional context informa-
tion.

Another interesting future experiment is to train a shared pruner, that would prune away unlikely spans for all three modules and be trained jointly with all three tasks.

It would also be interesting to look into explicitly encoding information about span width - the number of words that the span is based on - into the span embedding, as it is likely that the probability distribution of a span belonging to different entity types, belonging to a coreference cluster, and taking part in a relation is dependent on the length of the span. By explicitly encoding this information in the span, this information would be passed to each of the modules and also used to more effectively prune away unlikely spans in the coreference resolution and relation extraction modules.

A third direction that would be interesting to look into would be to anno-
tate new additional tasks and incorporating them into the architecture to see whether performance can be improved further with this approach. It would be interesting to combine this approach with a more systematic comparison on how model performance is impacted by more tasks to train on compared to more annotations on the specific task of interest.

It would also be interesting to investigate what model simplifications can be done while keeping high performance. Here, model simplifications re-
fer in part to using more shallow networks with fewer nodes in each layer, but also to how the the different modules and kinds of propagations can be
made more unified with fewer module specific attributes and fewer hyperparameters to tune. This is especially interesting since a more unified framework requires more sophisticated multi-task learning techniques in order to be successful.

It would be interesting to explore combining the LSTM-approach with fine-tuning BERT; the results indicate that it would be interesting to investigate further whether these span-based approaches benefit from using and fine-tuning BERT, followed by an LSTM. The results in this report indicate that this could potentially improve results further.

In this project, I only ran experiments with one type of graph propagation at a time. It would have been interesting to run more experiments to establish the interactions between coreference propagation and relation propagation and the effect on performance when running them together. Using both these together has been shown by Luan et al. [19] to help boost performance further, for a system using ELMo embeddings.
Chapter 6

Conclusions

In this final chapter, I take a step back and provide general conclusions.

6.1 Conclusions

In this degree project I studied a multi-task learning model that I and another student implemented in AllenNLP. This project explored different ways of improving the contextualization of span representations, by improved contextualization of word embeddings through supervised finetuning of BERT, as well as by using different kinds of graph propagations. The ablation studies indicate that graph propagation enhances performance across a wide majority of the tasks, by allowing information to propagate between spans on a document-level. The project provides an evaluation of the importance of in-domain pretrained language models for the scientific domain and show that in-domain pretraining significantly outperforms more general language models. This project also provides an evaluation of the importance of using in-domain finetuning of large language models, and concludes that this also benefits model performance. This work sets new state-of-the-art results across all tasks for all four datasets considered, each stemming from a different domain.
Bibliography


Appendix A

Class Labels for the Datasets

The class labels for each of the datasets are as follows:

A.1 SciERC

Entity and relation categories for the SciERC dataset are provided below.

Entity Categories for SciERC

- Task
- Generic
- Metric
- Material
- OtherScientificTerm
- Method

Relation Categories for SciERC

- COMPARE
- PART-OF
- FEATURE-OF
• USED-FOR
• CONJUNCTION
• HYPONYM-OF
• EVALUATE-FOR

A.2 ACE 2005

Entity and relation categories for the ACE 2005 dataset are provided below.

**Entity Categories for ACE 2005**

• LOC, (Location)
• WEA, (Weapon)
• GPE, (Geographical/Political)
• PER, (Person)
• FAC, (Facility)
• ORG, (Organisation)
• VEH, (Vehicle)

**Relation Categories for ACE 2005**

• PHYS, (Physical)
• PART-WHOLE, (Part-Whole)
• ART, (Artifact)
• ORG-AFF, (Org-Affiliation)
• PER-SOC, (Person-Social)
• GEN-AFF, (Gen-Affiliation)
A.3 GENIA

Entity categories for the GENIA dataset are provided below.

Entity Categories for GENIA

- Protein
- DNA
- RNA
- Cell-line
- Cell-type

A.4 WLPC

Entity and relation categories for the Wet Lab Protocol Corpus, WLPC, dataset are provided below.

Entity Categories for WLPC

- Action
- Amount
- Conc.
- Device
- Gen.-Measure
- Location
- Meas.-Type
- Mention
- Method
- Modifier
- Numerical
- Reagent
• Seal
• Size
• Speed
• Temperature
• Time
• pH

Relation Categories for WLPC

• Acts-on
• Creates
• Site
• Using
• Setting
• Count
• MeasureType-Link
• Coreference
• Mod-Link
• Measure
• Meronym
• Or
• Of-Type