Annotation Projection and Cross-Lingual approaches to Argument Mining for Norwegian

Anders Næss Evensen

Thesis submitted for the degree of
Master in Language Technology
60 credits

Department of Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Autumn 2020

Annotation Projection and Cross-Lingual approaches to Argument Mining for Norwegian

Anders Næss Evensen

Abstract

Argument Mining consists of finding and extracting argument structures from natural language texts. In this thesis we investigate the task of Argument Mining for Norwegian, and introduce the first annotated dataset for Argument Mining in Norwegian, dubbed NorArg. We also provide annotation guidelines describing the annotation process, allowing for easy expansion of NorArg in future research. In addition we perform analyses of several cross-lingual argument component classification systems, trained on annotated data in a resource-rich language, English, and tested on Norwegian data. We also provide detailed overviews of results from the cross-lingual systems being tested on NorArg.

Acknowledgements

First of all I would like to thank my supervisors, Samia Touileb and Lilja Øvrelid, for their continued support and valuable feedback. I would also like to thank Samia and Petter for contributing to the thesis and assisting me in the sometimes gruelingly frustrating task of annotating argument components. Finally I would like to thank K.A.I. and the Fortran gang for some sorely needed diversion throughout the process.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The goal of argument mining is to identify, extract and link arguments in textual documents, in order to structure and prepare them for further processing. The study of argumentation is a complex field which encompasses many different sub-tasks and disciplines like logic, philosophy, law and computer science (Lippi and Torroni, 2016). Argument mining is considered by some as a natural next step in sentiment mining (Lippi and Torroni, 2015), the key difference being that where sentiment mining looks at *what* people think about something, argument mining focuses on the reasons *why* they think that (Lippi and Torroni, 2015).

Some cognitive science theories suggest that argumentation is a central part of human reasoning (Lippi and Torroni, 2015), and achieving technology that automatically identifies argument structures could be a major step on the path to a computer program capable of reasoning. Argument mining is today considered to be one of the most promising research areas in artifical intelligence (Cabrio and Villata, 2018).

As with so many Natural Language Processing (NLP) tasks, the lack of high quality annotated data is a problem in argument mining. Where other tasks in Machine Learning (ML), like image recognition, can easily be performed by just about anyone, many tasks in NLP require expert annotators to produce high quality datasets. For this reason, many researchers look for ways of circumventing this lack of resources by using alternative sources to increase the performance of low-resource NLP systems. What is considered low-resource may vary depending on the task. Schulz et al. (2018) investigate different sets of training data consisting of datasets of various sizes, and examine the effects of applying multi-task learning to the task of argument component detection in a limited resource environment, by using a small portion of a data set. They argue that due to the subjective nature of arguments, and the difficulty of correctly annotating arguments, even for trained experts, obtaining a large annotated dataset of high quality is difficult (Schulz et al., 2018).

At the present time there are no annotated datasets for argument mining in Norwegian. The first contribution of this thesis is then to create the first annotated dataset for argument mining in Norwegian dubbed NorArg.

The second contribution of this thesis is a comprehensive set of guidelines describing the annotation process in detail, allowing for the expansion of the existing dataset in the future. We also provide inter-annotator agreement scores to verify the effectiveness of the guidelines.

In the field of argument mining, most of the existing research so far has been focused on using a single language (Eger et al., 2018). Because of the complexity of the argument mining task, and in particular the difficulties related to producing high quality annotated datasets for the task and the resulting lack of a sufficiently large dataset for Norwegian, we set out to investigate techniques for cross-lingual which allow us to leverage annotations from a high-resource language, namely English.

The third contribution of this thesis is then an experimental comparison of two cross-lingual learning techniques; the first using annotation projection, the second using multilingual transfer learning.

1.1 Outline

This thesis is structured in the following way.

Chapter 2 provides an overview of the previous work done in the field of argument mining, as well as descriptions of the neural architectures used in this thesis.

Chapter 3 describes the datasets used to train and test the models used in our experiments, as well as the underlying dataset used to create NorArg.

Chapter 4 describes the process of annotating in order to create NorArg, and provides guidelines for further annotation. This chapter also contains statistics for the finished dataset as well as inter-annotator agreement from the annotation process.

Chapter 5 describes the process required to format our datasets for annotation projection, as well as the translation and projection processes.

Chapter 6 details the model settings and training process of our experiments.

Chapter 7 gives an extensive overview of the various results from our experiments.

Chapter 8 summarises the results obtained in the thesis, and gives suggestions for possible future work.

Chapter 2

Background

In this chapter we take a look at what has been done in argumentation theory, the previous work done in the field of computational argumentation, and the current state of the art systems.

2.1 A brief history of argumentation theory

The idea that systems of logic could be applied to everyday life in rhetoric and debates has existed since the philosophers of ancient Greece (Groarke, 2017). The traditional way of looking at an argument has been to use what is known as a monological approach, looking at an argument in isolation, and deciding whether a conclusion logically follows from some premises. Intermittently, other approaches have been attemped, but a real change in direction happened in the last half of last century, particularly after Hamblin's Fallacies (1970). Hamblin adopted a dialogical approach by viewing arguments in the context of a dialogue with two opposing sides. This new approach, known as informal logic or argumentation, examines how two sides of an argumentation interact, with one side attacking and the other defending. Informal logic typically has four distinct tasks (Walton, 2009):

1. Identification: Identify the premises and the conclusion of an argument. (explained in more detail later)
2. Analysis: Find implicit arguments or conclusions. Implicit parts of the argument that are presumed to be already known by the reader, typically relations that are considered to be general knowledge. These are very common in natural texts.
3. Evaluating argument strengths: Determine the strength of arguments.
4. Invention: Construct new arguments to prove a specific conclusion.

Whereas in argumentation theory the focus is often on deciding which side has the more convincing arguments or whether a conclusion logically follows from its premises, the focus in argument mining is primarily separating out the arguments themselves and identifying the relations between them.

2.2 Definition of an argument in computational argumentation

Argument models used in computational argumentation can be divided into two subgroups.

Abstract argumentation models: Each argument is represented without any particular internal structure, and the aim is typically to analyse the relations between arguments. This style of modelling is based mostly on the work done by Dung (Lippi and Torroni, 2015).

Structured argumentation models: In these models each argument has a defined internal structure. This is crucial in most forms of computational argumentation mining, where an important sub-task is to identify the different components of an argument. There are many ways of structuring arguments, the minimal definition described in Walton (2009) defines an argument as a set of statements which can be split into three parts:

1. A conclusion. A controversial statement, typically the main component of the argument.
2. A set of premises: Evidence that supports or attacks the conclusion.
3. An inference from the premises to the conclusion.

This example sentence from Eger et al. (2017) contains all three:

$$\text{"Since } \underline{\textit{it killed many marine lives}}_{\text{Premise}} \\ \underline{\textit{tourism has threatened nature}}\text{"}_{\text{Claim}}.$$

The premise being "it killed many marine lives", which supports the claim "tourism has threatened nature", due to the implicit inference; marine lives are part of nature, and killing them threatens nature.

Toulmin's model: In The Uses of Argument (1958). Stephen Toulmin proposed a variation on the standard argumentation model. Instead of viewing arguments as being composed of statements that can either be claims or premises, Toulmin proposes six different roles:

1. Claims: the same as conclusion described above.
2. Data: Facts that explain or justify claims, the same as premises.
3. Warrants: Statements that explain the links between data and claims.
4. (Modal) qualifiers: Expressions that indicate the various strengths of warrants. The warrant can for instance lead to the data being "necessarily" true or "probably" true, hence why Toulmin uses the modifier modal.

5. Rebuttal: some warrants are not universally valid, and there are some cases where they can be falsified and have to be set aside.
6. Backing: in similar cases where the warrant is not universally valid or the warrant is challenged in some other way, a backing can be introduced to strengthen it (Freeman, 2011).

2.3 Argument Mining tasks

Due to the complexity of the argument mining problem, it has historically been divided into several subtasks, each requiring specific parts of the NLP toolset to complete.

Argument detection: The first problem is identifying the arguments in a document. This is a typical classification problem, determining which sentences contain arguments and extracting them. Historically, the main focus of the research done on this task has been on which feature sets to employ (Lippi and Torroni, 2015). One of the biggest issues with this task is that feature sets have shown to be very domain specific.
Argument segmentation: The problem of separating arguments, and identifying the components defined in the argumentation model we are using (usually a claim/premise model as described above). There are three ways an argument could be split across sentences:

1. A sentence contains only part of an argument component:
   "A significant number of republicans assert that *hereditary monarchy is unfair and elitist.* $_{\text{Claim}}$" (Lippi and Torroni, 2015)
2. Two or more argument components in one sentence.
   "Since *it killed many marine lives* $_{\text{Premise}}$
   tourism has threatened nature." $_{\text{Claim}}$
3. An argument component contains several sentences.
   "When New Hampshire authorized a state lottery in 1963, it represented a major shift in social policy. $_{\text{Claim}}$
   No state governments had previously directly run gambling operations to raise money. Other states followed suit, and now the majority of the states run some type of lottery to raise funds for state operations." $_{\text{Premise}}$
   (Lippi and Torroni, 2015)

Argument structure prediction: The final, and perhaps most difficult task is identifying the relations between the components of an argument. Historically, this is the task that has yielded the fewest results. (Lippi and Torroni, 2015).

2.4 Automated identification and extraction of argument structures

Whereas the theorists have gone into great depths regarding the validity and logical truthfulness of an argument, most practical applications have focused on detecting the argument components and the structure of arguments in natural language. In this section we attempt to give a comprehensive overview of previous work done in the field of argument mining.

2.4.1 Annotation for argument mining

As with many machine learning and NLP tasks, achieving good results in argumentation mining depends largely on the quality and quantity of data. Although information and data is steadily becoming more and more available, argument mining typically requires data annotated by experts, which takes a lot of time and money to produce. Several approaches have been used to overcome this issue. One of such is the use of crowdsourcing. Crowdsourcing means enlisting a large number of people to do the job you want, either paid or unpaid, typically through an internet platform. It has proven to be a good source of annotated data for less complicated tasks.

However, annotating the arguments present in a text has shown to be very difficult for non-expert annotators, compared to e.g. the classification of images which is a task that most non-expert individuals can perform.

2.4.2 Annotation Schemes

There are several existing annotated data sets for argument mining freely available. An extensive overview of the existing corpora is given in Cabrio and Villata (2018).

However, most annotation schemes typically only address one or a few of the tasks in the argument mining pipeline. For our own annotation scheme, we chose to use the scheme from Stab and Gurevych (2017) which has a more general approach and annotates components as well as relations, covering all of the argument mining tasks. Unlike most of the other studies, Stab and Gurevych (2017) also provide detailed guidelines which make our own annotation process much easier.

Stab and Gurevych (2017) look at the domain of persuasive essays and expand on their previous annotation scheme from Stab and Gurevych (2014), which covers every aspect of argumentation mining by annotating argument components as well as relations between arguments. They also provide comprehensive guidelines to assist readers in putting the scheme into action.

The most distinctive difference from the standard structured model, is that each persuasive essay has a *major claim*, a claim that the whole essay revolves around and attempts to justify. The scheme also includes support

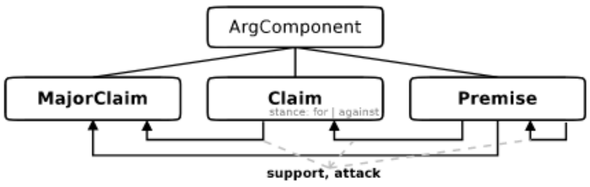


Figure 2.1: Diagram of the annotation structure. (Stab and Gurevych, 2014)

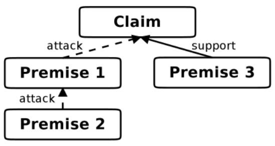


Figure 2.2: Diagram from (Stab and Gurevych, 2014) showing the inner structure of the argument described in section 2.4.1

and attack relations between claims and premises within arguments. In addition, each argument has a *stance attribute* expressing whether it supports or attacks the major claim. The inner relation labels allow for correctly labeling the relations within a more complicated sentence, like this example from Stab and Gurevych (2014):

"*Living and studying overseas is an irreplaceable experience when it comes to learn standing on your own feet.*$_{\text{Claim}}$ **One who is living overseas will of course struggle with loneliness, living away from family and friends** $_{\text{Premise}_1}$ but **those difficulties will turn into valuable experiences in the following steps of life.** $_{\text{Premise}_2}$ Moreover, **the one will learn living without depending on anyone else** $_{\text{Premise}_3}$"

In the above argument, premise$_1$ attacks the claim, while premise$_2$ refutes premise$_1$. Premise$_3$ supports the claim. The argument structure of this argument is shown in figure 2.2.

This scheme seems to be suitable for our own dataset, as reviews can be seen to have a major claim in the form of the reviewers final opinion on the object being reviewed.

2.4.3 Previous work

In this section we attempt to give an overview of some of the research done in the field of argument mining.

Argument mining domains

Argument mining systems have been applied to several different domains. In their paper, Cabrio and Villata (2018) give an overview of some of the more recently used ones, we list some of them here:

- **Education**: Within the education domain most of the research has been focused on two particular fields:
  - Persuasive essays: Essays concerning a specific topic where the author attempts to convince the reader that their particular point of view is the right one. In Stab and Gurevych (2017) the authors propose using sequence labeling on the token level to identify argument components, and create a corpus of annotated persuasive essays.
  - Scientific articles: Described in some of the earliest work on argument mining, Teufel et al. (2009). Arguments here typically consist of the author's view of related work, and opinions about problem-solving processes (Cabrio and Villata, 2018).
- **Web Based Content**:
  - Wikipedia: Researched in several papers, for instance in some earlier work by IBM, as a part of their effort to develop debating technologies (Cabrio and Villata, 2018).
  - Microblogs and web debating platforms: This is an interesting domain, containing user-generated discourse that is more natural and unrefined than what can be found in the other domains. Several works have been done in this domain focusing on tasks like argument detection and relation prediction (Cabrio and Villata, 2018).
  - Online product reviews: Argument mining in this domain to some extent overlaps with sentiment mining. Sentiments about the different aspects of the product also often contain the reasoning behind the author's view. (Cabrio and Villata, 2018)

Feature sets

Features are tools used to identify and single out the parts of the text that we are most interested in. They are often handcrafted and can be very specialised. Aker et al. (2017) describe and evaluate some of the most common feature sets used in argument mining:

- **Structural features**: Statistical features containing info about tokens and punctuation. Found to be the most significant feature set for both the argument identification and argument structure prediction tasks in Aker et al. (2017).

- **Lexical features**: Unigram frequencies and verbs and adverbs that stand out. Second most significant feature set in Aker et al. (2017).
- **Syntactic features**: Occurences of frequent POS-sequences. The least relevant feature set in Aker et al. (2017).
- **Indicators**: A list of keywords that indicate the presence of claims or premises.
- **Contextual features**: Structural and lexical features of surrounding sentences.
- **Word embeddings**: Every word is represented as a vector of numbers. Word embeddings that are pre-trained on large corpora can be used, as in Aker et al. (2017), where they used word embeddings trained on the Google News Corpus.

2.5 Neural approach

In recent years, neural network architectures have outperformed the previous state of the art in most fields within NLP, as well as in machine learning in general. One big difference from previous architectures is that neural networks in most cases do not depend on handcrafted feature sets, which means that they are far less time-consuming to implement.
In this section we attempt to give a comprehensive overview of the most popular forms of neural networks architectures used in NLP in general, and the field of Argument Mining in particular. We also go into more detail about the specific architectures used in our thesis, and their components.

2.5.1 Recurrent neural networks (RNN)

The RNN (Elman, 1990) is a neural network that is well suited to sequences of input, due to its ability to "remember" previous parts of the sequence. The RNN can be visualised as a number of timesteps, one for each part of the input (if the input is a sentence, each timestep corresponds to a word in the sentence) with a corresponding set of weights for each timestep. At each timestep, the RNN learns from a part of the input, but it also takes into account some information from what was learnt at the preceding timestep, from the preceding part of the input. In this way the RNN preserves some of the inputs structural information. Figure 2.3 shows the basic architecture of the RNN. In the figure, each box represents a timestep. s_n indicates the state information being passed from timestep to timestep containing information about previous words in the sequence. x_n are the input tokens and y_n the output usually function of s_n and x_n at each timestep. Due to the fact that the RNN remembers some information from each and every timestep, if the input sentence sequence is very long it's "memory" over time grows very large, and this in turn leads to long-range depencies between different parts of input maybe being lost. This problem is known as vanishing or exploding gradients.

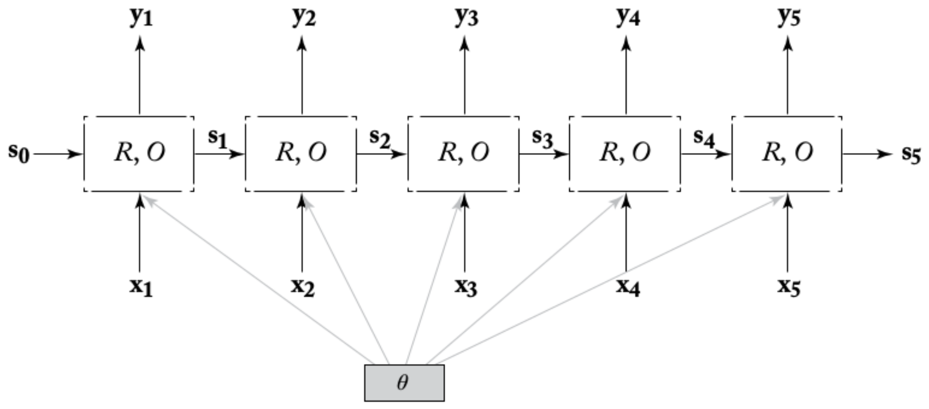


Figure 2.3: Illustration from Goldberg and Hirst (2017) showing the architecture of a basic RNN. x_n are the input tokens, y_n the output at each timestep, and s_n the current state information at each timestep, used to calculate the output based on information from the previous timesteps. θ symbolises the models parameters, indicating that they are the same for all timesteps (Goldberg and Hirst, 2017).

2.5.2 Long Short-term memory

The LSTM is a variant of the RNN architecture, first introduced in Hochreiter and Schmidhuber (1997). It contains different types of gates, which at a given timestep, determine how much of the input should be taken in, how much of the information from the previous states should be remembered, and how much should be included in the output. This alleviates the problem of exploding or vanishing gradients. For each token, the input is sent through a tanh (hyperbolic tangent) activation function and a component label is predicted. The component label, in embedding form, is then used as history input for the next token. The LSTM can be expressed mathematically as shown in equation 2.1 from Goldberg and Hirst (2017),

$$\begin{aligned}
i_t &= \sigma(W^{(i)}x_t + U^{(i)}h_{t-1} + b^{(i)}),\\
f_t &= \sigma(W^{(f)}x_t + U^{(f)}h_{t-1} + b^{(f)}),\\
o_t &= \sigma(W^{(o)}x_t + U^{(o)}h_{t-1} + b^{(o)}),\\
u_t &= tanh(W^{(u)}x_t + U^{(u)}h_{t-1} + b^{(i)}),\\
c_t &= i_t \odot u_t + f_t \odot c_{t-1},\\
h_t &= o_t \odot tanh(c_t)
\end{aligned} \tag{2.1}$$

where the gates at time t are expressed by i_t (input gate), o_t (output gate), and f_t (forget gate). The gate values are computed using a sigmoid activation function and a linear combination of the current input x_t and the previous hidden state h_{t-1} (Goldberg and Hirst, 2017). The update u_t is computed using a tanh activation function and a linear combination of the current input x_t and the previous hidden state h_{t-1}. The memory, c_t is then computed using the input gate i_t to control how much of the update to keep, and the forget gate f_t to control how much of the info from the previous memory c_{t-1} to keep. Then the hidden state h_t is computed by using the output gate and the memory cell passed through a tanh activation layer. The hidden state in this case corresponds to the output y_t in figure 2.3 (Goldberg and Hirst, 2017).

2.5.3 Attention

Bahdanau et al. (2014) introduced a model that uses two RNNs, an encoder and a decoder, to translate an input sentence from a source language to a sentence in a target language.

The encoder RNN transforms the input sentence from the source language into a fixed size vector, which becomes the input for the decoder RNN. The decoder then outputs a sentence in the target language. The problem with this technique according to Bahdanau et al. (2014), is the fact that the decoder requires a fixed size vector as input. This means that no matter the length of the input sentence, the encoder needs to compress it into a vector of a fixed size, which might lead to a loss of information when the input sentence is long, especially if the sentence is longer than any sentence in the models training set. Bahdanau et al. (2014) propose a solution to this by creating a mechanism that lets the model focus on the most relevant parts of the input and use those parts to output a word in the target language.

The mechanism, later known as attention, consists of an addition to the traditional encoder-decoder architecture which allows the model to align and translate jointly (Bahdanau et al., 2014). When the model predicts a word in the target language, it searches for the positions in the source sentence that have the most relevant information for the predicted word, and predicts a target word based on the information from these positions and information from the previously generated target words (Bahdanau et al., 2014). This allows the model to no longer rely on fixed input vectors as input to its decoder, but instead encodes a context vector for each word in the input sentence, and lets the decoder choose a subset of these context vectors to output a target word, based on the alignment information which shows what words are the most important for deciding the output.

Bahdanau et al. (2014) evaluated their model on the task of English-French translation, while comparing it to a model introduced in (Cho et al., 2014). This paper uses a similar architecture, a RNN Encoder-Decoder, but without the attention mechanism. They train two models: one based on their own architecture, which they refer to as RNNsearch, and one based on Cho et al. (2014), called RNNencdec. They train two version of each model. One version where the training set is restricted to sentences of up to 30 words, and one version where the sentences are up to 50 words long. As is shown in figure 2.4, the models using the attention mechanism outperform the other models consistenly, especially when tested on long sentences. RNNsearch-30 even outperforms RNNenc-50 when tested on

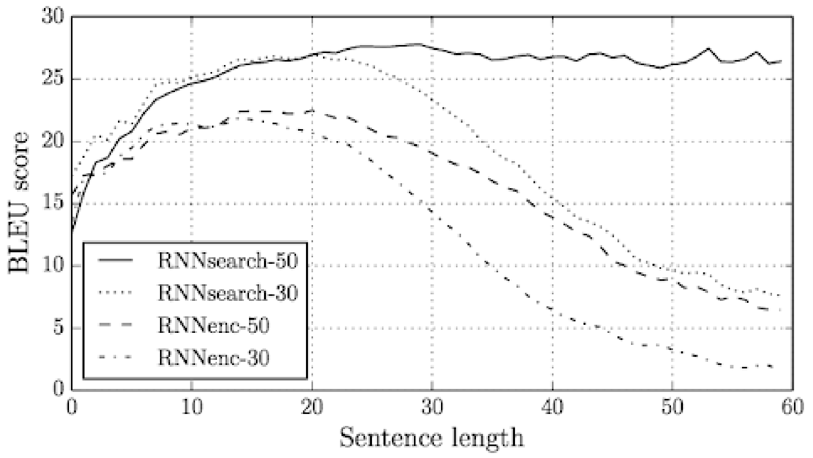


Figure 2.4: Illustration from Bahdanau et al. (2014) showing BLEU scores from the two models they tested. Each model was trained twice, once with sentences of up to 30 words, and once with sentences of up to 50 words.

sentences of length 50, even though RNNenc-50 has sentences of the same length in its training set, while RNNsearch-30 does not. This indicates that the model using attention is indeed better at understanding long sentences even when not trained on sentences of the same length (Bahdanau et al., 2014).

2.5.4 Transformers

Vaswani et al. (2017) introduced the Transformer which is a sequence transduction model that is basically a model that receives an input sequence and transforms it in some way to produce an output sequence. It is the first sequence transduction model that relies solely on the attention mechanism by Bahdanau et al. (2014) discussed in the previous section. At the time the paper was written, most other neural sequence transduction models relied on recurrent or convolutional networks acting as encoders and decoders (Vaswani et al., 2017).

The Transformer uses the same encoder-decoder architecture, but instead of RNNs uses layers of attention. The Transformer architecture is shown in figure 2.5. The encoder part of the Transformer consists of 6 identical layers. Each layer has two sub-layers; a self-attention layer and a fully connected feed-forward layer. Self-attention is a form of attention that focuses on the most important parts of the input itself to use in the encoding of the input. The self-attention in the Transformer is implemented in a way that makes it attend to each position in the output from the previous layer in the encoder (Vaswani et al., 2017).

The feed-forward network consists of two linear transformations and a ReLU activation function. After each layer some normalization methods are applied to the output (Vaswani et al., 2017), as shown in figure 2.5.

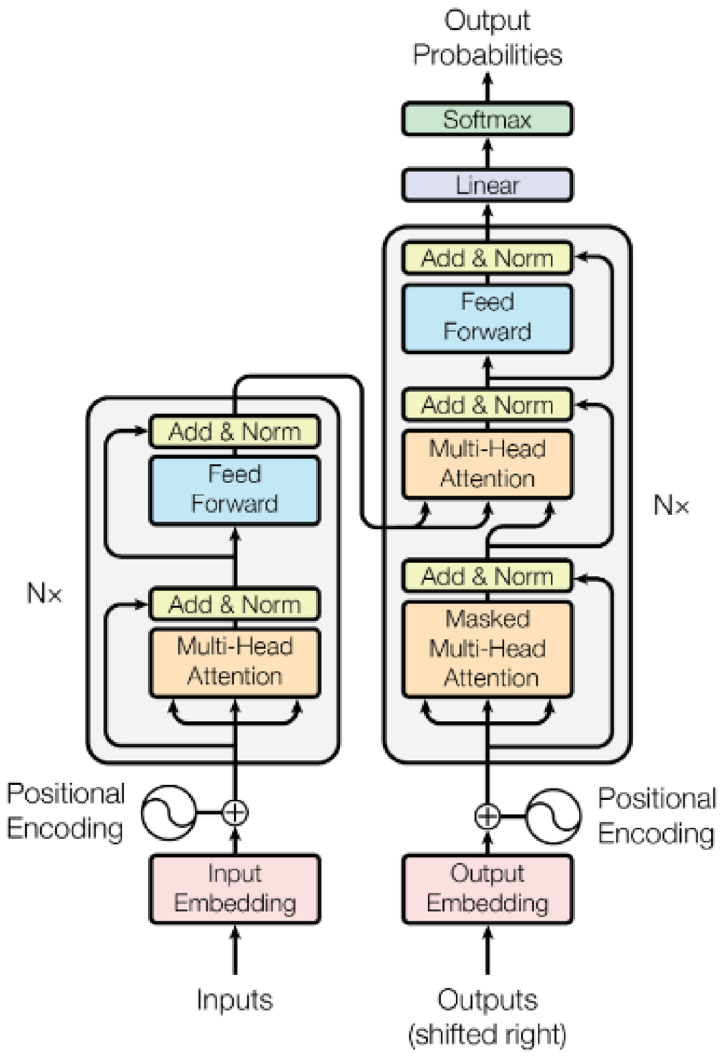


Figure 2.5: Illustration from Vaswani et al. (2017) showing the Transformer architecture. The left part of the figure shows the encoder, and the right side is the decoder. The encoder has 6 layers, each consisting of a self-attention layer and a fully connected feed-forward network. The decoder has 6 similar layers, but in addition to the two sub-layers in the encoder, each layer in the decoder has a multi-head attention layer for the output from the encoder stack.

The decoder has a very similar architecture, also with 6 identical layers. Each layer in the decoder, however, has an additional sub-layer that applies attention to the output from the encoder stack, as is illustrated in figure 2.5.

Because the model doesn't use RNNs or CNNs, which naturally incorporate information about the order of the sequence by iteratively moving through the sequence from left to right or right to left, the model needs some way to include information about the ordering of the sequence. This is solved by applying something called positional encoding to the inputs of both the encoder and decoder. The chosen form of positional encoding in Vaswani et al. (2017) is to apply sine and cosine functions of different frequencies to the inputs, which transforms each position into relative positions. Using this method means that for two given positions, one position can be expressed as a linear function of the other (Vaswani et al., 2017)

. Vaswani et al. (2017) hypothesize that this will allow the model to gain information about positioning of each word in the sequence. When tested on the WMT (Workshop for Machine Translation) 2014 English-to-German and WMT 2014 English-to-French translation tasks, the Transformer architectures outperform all previous state-of-the-art models (Vaswani et al., 2017).

2.5.5 Word Embeddings

Word embeddings were first introduced in Mikolov et al. (2013), and have come to be a more or less required component in most modern implementations of NLP. Word embeddings are a way of presenting words in a form that reflects the context in which they usually appear, based on on large collections of text. Several pre-trained word embeddings are freely available for use.

2.5.6 Model settings

There are some aspects of neural models that are shared across most different types of models, for instance techniques to avoid being stuck in local maxima, normalizing inputs or mapping outputs to a probability distribution. In this section we describe some of the mechanisms used in this thesis, as well as types of hyperparameters used to train different models.

Dropout

Dropout (Hinton et al., 2012; Srivastava et al., 2014) is a regularisation method that has a chance to zero out some elements of an input. This helps diversify the inputs, and can prevent overfitting. Removing different parts of the input from iteration to iteration by some of the elements being zeroed out to some extent simulates using different types of, or more input. This alleviates some of the need for large amounts of data.

2.6 Toolkits and systems

Neural systems are typically built up of many different components, consisting of the different neural architectures in different combinations. In this section we describe the toolkits and systems we have used in this thesis to perform our experiments.

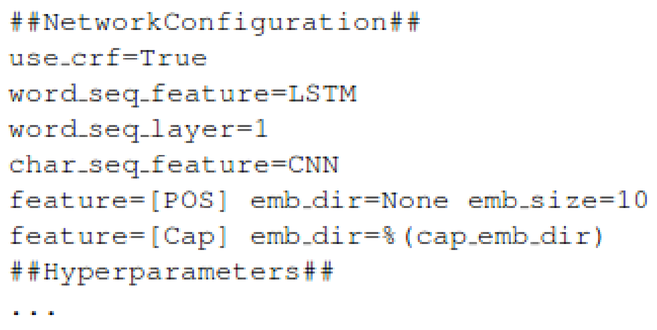
LSTM-ER

In Eger et al. (2017), the authors set out to build a single neural system to perform all the argument mining tasks on an annotated data set consisting of persuasive essays. The single system architecture is different from most earlier attempts, where several models are combined in a pipeline, each one trained on a particular task and/or domain. These architectures often heavily depend on domain specific feature sets. Another critique of the pipeline approach is that it doesn't take into account the relations between the sub-tasks, which can lead to errors propagating through the pipeline (Eger et al., 2017).
Eger et al. (2017) approach the task in four different ways, then compare the results: first as a dependency parsing problem, due to the tree-like structure arguments often form, second as a sequence tagging problem (using a BiLSTM to classify sequences), which seems natural, as a big part of argumentation mining is defining the span of the different argument components. The challenge in this approach is identifying the relations between arguments, as they can be very far apart. This is solved using the standard BIO (beginning, inside, outside) tagging common in entity recognition problems, and coding the distances between linked components into the tag label (Eger et al., 2017). Their third approach frames the problem as a multi-task (tagging) problem, using sub-tasks of argument mining as auxiliary tasks to see if this increases performance. In the fourth and final approach they used a previous system, the LSTM-ER (Miwa and Bansal, 2016), which combines sequential (entity) and tree structure (relation) information (Eger et al., 2017).

They found that treating argument mining as a token-based dependency parsing problem is largely ineffective (Eger et al., 2017), and that the sequence tagging approach performed well across domains, and generally better than the current state of the art at the time. They also found that multi-task learning increased performance (Eger et al., 2017). For our initial experiments we decided to implement our own version of the LSTM-ER.

NCRF++

NCRF++ was introduced in Yang and Zhang (2018), and is a toolkit for neural sequence labeling. Sequence labeling is a central part of many NLP tasks like named entity recognition (NER), chunking, word segmentation and part-of-speech (POS) tagging (Yang and Zhang, 2018). Sequence labelling has traditionally been performed using statistical models (Yang and Zhang, 2018), in which the addition of the CRF architecture (Lafferty et al., 2001) has proven to be an effective tool (Yang and Zhang, 2018). While there exists several open-source toolkits that allow you to implement CRF sequence labeling models, there are not so many available choices for complete neural sequence labeling toolkits (Yang and Zhang, 2018). NCRF++ aims to provide an easy-to-use base for neural sequence labeling tasks, and provides implementations of the most commonly used neural sequence models, such as LSTM-CRF



```
##NetworkConfiguration##
use_crf=True
word_seq_feature=LSTM
word_seq_layer=1
char_seq_feature=CNN
feature=[POS] emb_dir=None emb_size=10
feature=[Cap] emb_dir=%(cap_emb_dir)
##Hyperparameters##
...
```

Figure 2.6: Illustration of a configuration file from Yang and Zhang (2018), showing part of the module setup. NCRF++ allows for switching between different pre-implemented architecture components simply by changing the configuration files, making it possible to use several different model setups without having to code anything.

(Yang and Zhang, 2018). NCRF++ is fully configurable with premade model architectures through configuration files as shown in figure 2.6. The configuration files also allow for setting hyperparameters, making it relatively easy to modify your experiments. NCRF++ is written using the PyTorch library, which allows for implementation of your own custom modules and using them in the NCRF++ setup. The architecure of the NCRF++ system consists of three layers, a character sequence layer, a word sequence layer and an inference layer (Yang and Zhang, 2018), as shown in figure 2.7. The character sequence layer and word sequence layers turn input sentences into character and word embeddings, respectively, then pass their input to the inference layer which assigns labels to each word (Yang and Zhang, 2018).

In accordance with the idea that NCRF++ should be easily modifiable, there are several interchangeable modules that can be used to serve as the different layers. The character sequence layer has several different encoders available, such as an RNN, along with variants of RNN such as GRU or LSTM, or a CNN (convolutional neural network). The same architectures are available for the word sequence layer (Yang and Zhang, 2018). The inference layer turns the output from the previous layers into labels to apply to the words in the input sentence. There are two main inference mechanisms available for the inference layer, Softmax and CRF.

Yang and Zhang (2018) compared their system to several state-of-the-art models and found their own results to be comparable, showing that their system is up to par with most contemporary systems.

BERT

BERT (*Bidirectional Encoder Representations from Transformers*) was introduced in Devlin et al. (2018). It is a language representation model, with a

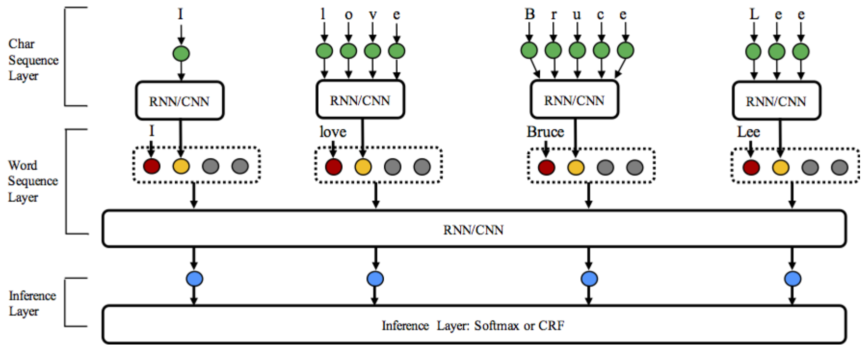


Figure 2.7: Illustration of the architecture of the NCRF++ framework from Yang and Zhang (2018). In the figure, the input is the sentence "I love Bruce Lee." The character sequence layer receives each character as input in the form of its embedding, encodes it, and sends it to the word sequence layer, where word embeddings, character sequence representations and embeddings of handcrafted features are encoded into word sequence representations. Finally the inference layer assigns each word a label by using word sequence representations from the word sequence layer (Yang and Zhang, 2018).

Transformer based architecture (Vaswani et al., 2017). The main difference between BERT and most of the contemporary Transformer based models, is that BERT aims to have a bidirectional approach to training and fine-tuning. This means that when the model is fine-tuned for a specific task, each token is encoded with context information from tokens on both the left and right sides of it. Most other transformer based models at the time, such as OpenAI GPT (*Generative Pre-trained Transformer*) (Radford et al., 2018) and ELMo (Peters et al., 2018) are unidirectional, meaning that each token only receives context information from the left or right side of it (Devlin et al., 2018). Devlin et al. (2018) argue that obtaining context from both the left and right sides is essential when performing sentence level tasks, and propose a novel model which aims to do exactly that.

There are two separate parts of the BERT architecture; pre-training and fine-tuning (Devlin et al., 2018). The architecture is a multi-layer bidirectional Transformer encoder, and is more or less built the same way as in Vaswani et al. (2017), as was explained in the section about the Transformer. During the pre-training phase, the model uses two forms of unsupervised learning with the goal to make the model more bidirectional than previous architectures. The first technique is using a "masked language model" (Devlin et al., 2018). The masked language model works by masking some of the tokens in the input at random, then having the model learn how to predict the masked token by using the non-masked context around it. This forces the model to predict a token based only on the context surrounding it, allowing it to use both the left and right

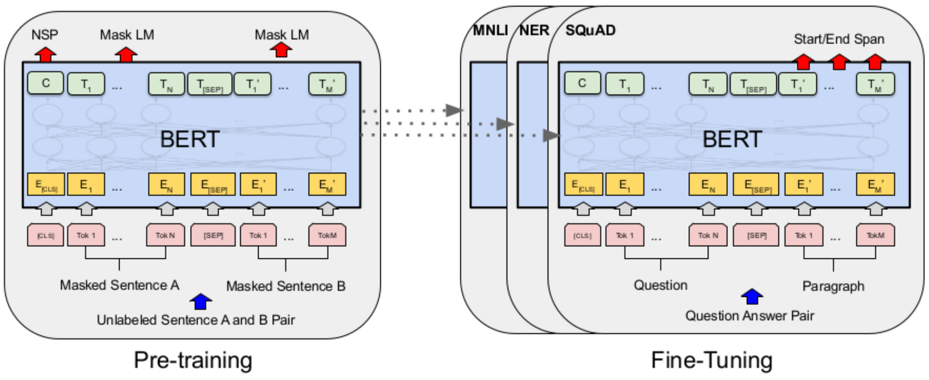


Figure 2.8: Illustration from Devlin et al. (2018) showing the architecture of BERT. The left side shows the pre-training techniques used, Masked LM and Next Sentence Prediciton (NSP). The right side shows some downstream tasks used for fine-tuning. The same Transformer architecture, apart from the output layers, is used for both pre-training and fine-tuning (Devlin et al., 2018).

sides.

The second technique is a "next sentence prediction" task (Devlin et al., 2018), which consists of a binary sentence prediction task. Given an input sentence pair, the model classifies whether the second sentence is a natural continuation of the first sentence. The reasoning behind why this technique is useful is that many NLP tasks are based on understanding the relationship between sentences, for instance in question answering or natural language inference (Devlin et al., 2018). The sentence pairs are made by pairing sentences from the corpus. 50% of the sentence pairs are actual sentences that follow each other in the training corpus and are labeled as positive examples. The remaining half of the sentence pairs are two unrelated sentences and are labeled as negative examples (Devlin et al., 2018).

The fine-tuning begins with the BERT model being initialized with the pre-trained parameters. Because of the self-attention mechanism inherent in the Transformer architecture, switching between fine-tuning for different downstream tasks simply consists of changing the inputs and outputs. There are some differences in the input format required by BERT and other contemporary transformer models (Devlin et al., 2018).

In order for BERT to be able to handle several different NLP tasks, (Devlin et al., 2018) allow their input to be both a single sentence, and a pair of sentences. This is possible because of the self-attention mechanism which makes the model encode the input sequence with cross attention, attending to each word in both of the input sentences. This is useful for instance for the question answering task, where both the question and the answer is needed. Figure 2.9 shows the format of BERT inputs for various

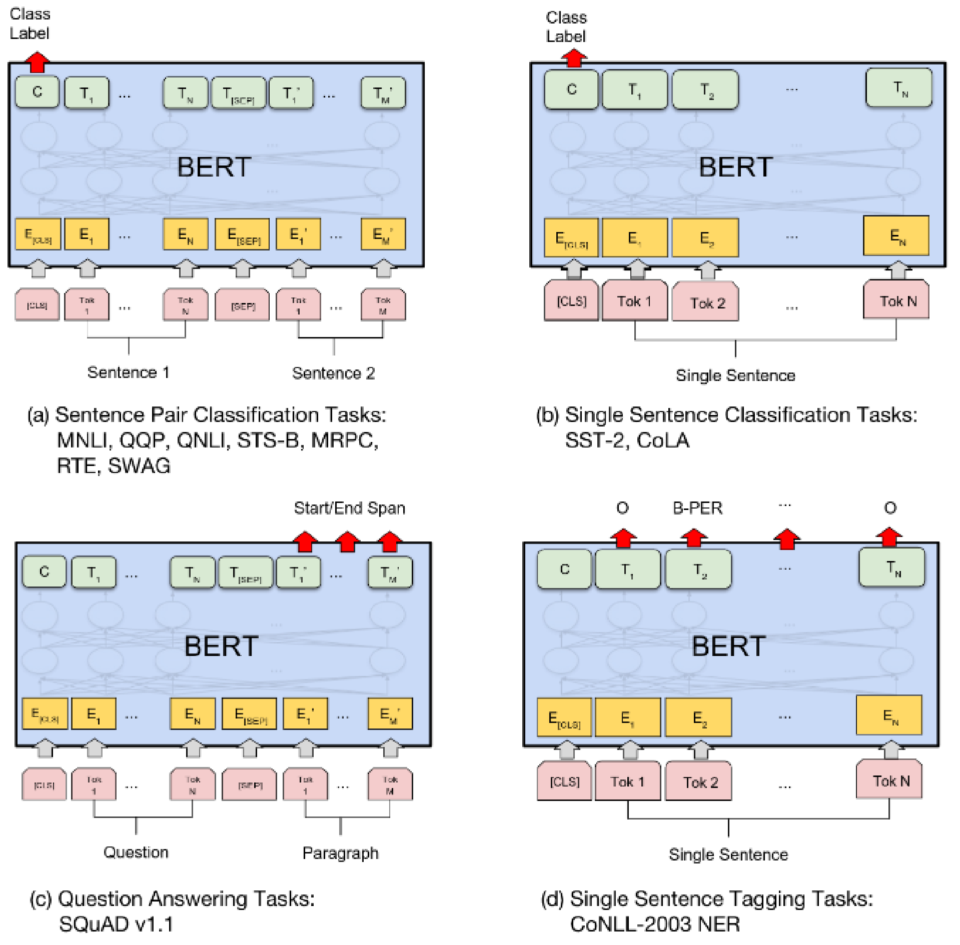

Class Label
C
T1
TN
T[SEP]
T1'
TM'
BERT
E[CLS]
E1
EN
E[SEP]
E1'
EM'
[CLS]
Tok 1
Tok N
[SEP]
Tok 1
Tok M
Sentence 1
Sentence 2
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG
Class Label
Tok 2
Single Sentence
(b) Single Sentence Classification Tasks: SST-2, CoLA
Start/End Span
Question
Paragraph
(c) Question Answering Tasks: SQuAD v1.1
O
B-PER
O
(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Figure 2.9: Illustration from Devlin et al. (2018) showing BERT input formats for various NLP tasks.

NLP tasks. Every sequence begins with the classification token *[CLS]*, and if the input sequence contains two sentences, the sentences are separated by the *[SEP]* token.

Multilingual BERT

Not long after the initial release of BERT, a version of BERT trained to understand several languages was released. Multilingual BERT is simply a BERT model trained to understand several languages, by training on articles from the 100 languages with the largest Wikipedias. Because the sizes of the Wikipedias for the different languages vary greatly, a number of smoothing techniques were applied to weight the various input languages. Multilingual BERT was tested on the XNLI dataset[1], an evaluation corpus

[1]https://github.com/facebookresearch/XNLI

for cross-lingual sentence translation in 15 languages, with results showing that BERT performs better than the XNLI baseline on all languages[2].

[2]https://github.com/google-research/bert/blob/master/multilingual.md

Chapter 3

Existing datasets

In this chapter we describe some of the existing Argument Mining datasets, as well as the underlying dataset using to create our annotated dataset for Argument Mining in Norwegian, NoReC.

3.1 English - Persuasive essays

The first version of the persuasive essay dataset was introduced in Stab and Gurevych (2014). At the time, there were no other existing datasets of persuasive essays annotated with an annotation scheme as detailed as the one used in Stab and Gurevych (2014). The first version consisted of 90 essays selected from essayforum [1]. Each text was manually reviewed, and in the end the corpus contained 1673 sentences with 34917 tokens (Stab and Gurevych, 2014). In Stab and Gurevych (2017) the original argument mining dataset from Stab and Gurevych (2014) is expanded from 90 essays to 402 essays. Some statistics for the set are shown in table **??**.

3.2 Annotation Guidelines

While the annotation process might sound straight forward in theory, in real life situations, as with so many things, that is rarely the case. In order to demystify the process and assist anyone who might want to attempt reproducing it at a later stage, it is good practice to include guidelines to elaborate on the choices made in the process. For the most part, our annotation guidelines are modeled on the very detailed guidelines produced in Stab and Gurevych (2014), with some small changes to make it more suitable to our domain of reviews. Our annotation process consists of two steps; annotating argument components, and identifying the relations between them.

Stab and Gurevych (2017) describe the process they went through in order to come up with a working annotation scheme. Their scheme consists of the argument components claims and premises, as well as two forms of relations; support and attack. Because their scheme is made for persuasive

[1]http://www.essayforum.com

	all	*average per essay*
Sentences	7116	18
Tokens	147271	366
Paragraphs	1833	5
Arg.components	6089	15
MajorClaims	751	2
Claims	1506	4
Premises	3832	10
Claims (for)	1228	3
Claims (against)	278	1

Table 3.1: Table from Stab and Gurevych (2017) showing the various inter-annotator agreement scores on the different argument components.

essays, it also includes a major claim, which can be thought of as the conclusion of the essay, the main point the author is arguing for. This translates well to our own domain (reviews), where each review can be expected to contain some final conclusion on the target of the review, a final verdict on whether the reviewer likes it or not.

3.2.1 Pre-study

To determine whether the initial scheme was usable and how efficient it was, Stab and Gurevych (2017) performed a pre-study. 14 short text snippets (1-2 sentences) were produced, either by gathering them from example essays or by the authors writing them themselves. Five non-trained annotators were then asked to identify which of the sentences were argumentative, and identify claims and premises in the sentences marked as argumentative. Their first results were not very convincing (inter-rater agreement of 58.6%), which they found was mostly caused by the annotators not knowing the context of the text snippets. To prevent this when annotating the actual corpus, the annotators were instructed to read the entire texts before annotating.

3.2.2 Top-down process

After conducting their pre-study the annotation is generalised in these three steps:

1. Topic and stance identification: Annotators identify the topic and stance of the essay by reading the whole text before annotating.
2. Annotation of argument components: First, annotate the major claim (usually found in the introduction or conclusion of the essay). Then the annotators find claims and premises in each paragraph. Each

Component type	*Observed agreement*	*Fleiss' κ*	α_U
MajorClaim	97.9%	.877	.810
Claim	88.9%	.635	.524
Premise	91.6%	.833	.824

Table 3.2: Table from Stab and Gurevych (2017) showing the various inter-annotator agreement scores on the different argument components.

argument component is annotated as a statement covering an entire sentence or less.

3. Annotation of argumentative relations: The final step is linking claims and premises within each paragraph and linking claims to the major claim with a support or attack relation.

3.2.3 Inter-annotator agreement

In Stab and Gurevych (2017), three annotators annotated a subset of the dataset consisting of 80 essays. The rest of the essays were annotated by an expert on argument mining. Stab and Gurevych (2017) begin by evaluate whether annotators agree on the presence of argument component in a given sentence, using the metrics *observed agreement* and *Fleiss' κ* Fleiss (1971). In addition to that they use a metric that takes into account the component boundaries, Krippendorff's α_U Krippendorff (2004). Their results in table 3.2 show that their guidelines provide a good explanation of how to annotate argument components, judging by the agreement between annotators.

3.3 Norwegian - Reviews

The data for our Norwegian dataset is taken from NoReC, the Norwegian review corpus. NoReC consists of more than 35,000 full text reviews from various genres, in Norwegian. The corpus was created in collaboration with the Norwegian Broadcasting Corporation (NRK), Schibsted Media Group and Aller Media, three of Norway's largest media groups (Velldal et al., 2017). Although these reviews typically have an unstructured quality about them (at least compared to the persuasive essays our model is trained on), they are in essence argumentative in nature. NoReC is categorised into 9 different categories. For our experiments we selected 100 texts at random from the 'screen' category, which consists of 13,085 reviews of movies and TV-series. Although reviews inherently have an unstructured quality about them (at least compared to the persuasive essays our model is trained on), they are in essence argumentative in nature.

Chapter 4

Annotating arguments in Norwegian

A vital part of our contribution in this thesis is the creation of the first Argument Mining dataset in Norwegian, NorArg, as well as guidelines for how to proceed in the annotation process. In this chapter we describe in detail how we constructed these guidelines, and the process of annotating our data set.

4.1 Guidelines

Our annotation process consisted of the main annotator annoting 40 documents randomly selected from the 100 screen reviews we began with, which were taken from NoReC, as described in chapter 3. To follow up, we had two additional annotators annotate the first 10 documents of the set, in order to look at the agreement between annotators.

4.2 Annotating argument components

Since inter-annotator agreement was found to increase significantly after the annotators familiarised themselves with the context of the components (Stab and Gurevych, 2014), we had the annotators read the entire document and understand it well before beginning the annotation. We made sure they understood the object of the review and the author's view towards it before beginning.

Following the annotation scheme described earlier, there are three different components to annotate, major claims, claims and premises, each uniquely identifiable. There are some general rules that apply for all the components:

- Punctuation should not be included (full stops, commas, exclamation marks)
- As much as possible, try to include complete sentences (this is harder in our domain than in persuasive essays, as reviews typically are less

structured.)

4.2.1 Major claims

As previously explained, major claims can be seen as statements that sum up the entire text (if appearing in the conclusion), or as a statement that the whole text revolves around attempting to justify (if appearing in the introduction). After reading the entire document and understanding the contents of it, the annotator should find a statement that sums up the view of the author regarding the item being reviewed. As in persuasive essays (Stab and Gurevych, 2014), these are typically found in the introduction or conclusion of the review (and often both). If there are several statements that express the same sentiment, choose the most explicit, or the one closest to a complete sentence. In some rare cases there is no clear major claim, and in such cases no major claim should be marked. Once a major claim is identified, the annotator marks the stance of the major claim in regards to the topic of the review (for, against or neutral).

These questions might help in deciding whether a sentence contains a major claim (Stab and Gurevych, 2014):

- Does the sentence include a statement that represents the major stance of the author with respect to the topic?
- Does the sentence include the most explicit stance expression in the introduction and conclusion of the essay?

4.2.2 Claims

Because claims are so context dependent, Stab and Gurevych (2014) advise looking at one paragraph at a time and trying to identify the most general claim in each paragraph. In persuasive essays a paragraph in most cases contains a single argument with one claim and reasons supporting that claim (Stab and Gurevych, 2014). Because reviews are typically not as structured as persuasive essays, this does not work as well in our case. Instead, it seems sensible to identify a section of the document discussing a certain topic, and marking the most general claim about that topic before moving on to the next topic being discussed and identifying the next claim, and so on.

Claims typically come in two forms, either as an initial assertion backed up by following premises, or as a conclusion supported by preceding premises. In some cases it appears in both forms, in that case both should be marked (Stab and Gurevych, 2014). Each time we identified a claim, we annotated it and marked the stance of the claim (in reviews, claims some times only describe the plot of the show, in those cases the stance is not marked), whether it is against or for the opinion expressed by the major claim.

Then we continued in this manner until all claims in the document had been found and annotated.

Stab and Gurevych (2014) describe a simple test to check whether you have found a claim. The statement "It is true that, <claim>" should be grammatically correct. It might be necessary to rearrange the claim statement to make this test work. (Or add missing words, especially in the case of subheadings)

Stab and Gurevych (2014) also provide us with some questions the annotator can use to decide whether a given sentence or statement is a claim:

- Is the statement supported by at least one other statement?
- Is there a reason given why the statement should be considered as true? (This one is not required for there to be a claim, sometimes claims appear without support)
- Is the statement an assertion with respect to a certain aspect?

4.2.3 Premises

Premises are statements that support or attack claims. In order to find the premises we went through the whole document again, topic by topic as when annotating claims, looking at the previously annotated claims, and finding the premises that supported or attacked them. Most likely a fair few of them were already identified in the previous step.

These questions might be helpful in deciding whether a statement is in fact a premise (Stab and Gurevych, 2014):

- Is the statement a reason or justification (or an attack) for the considered claim?
- Is the statement supporting another premise?
- Does the statement contribute to the confirmation of the claim?

4.2.4 Annotating argument relations

After all the components in the document had been identified, the next step was to annotate the relations linking them together. Possible relations are attack or support relations, and they can be between a premise and another premise, or a premise and a claim (Stab and Gurevych, 2014).

Attack relations describe source statements that try to disprove or undermine their target statement. A simple test to see if a relation between a source and a target is an attack relation is to see if the following sentence makes sense: "It is not true that <target statement> because <source statement>" (Stab and Gurevych, 2014). If it does, the relation is an attack relation and should be labeled as such. (in the Brat annotation tool, this corresponds to the label "against")

Support relations, on the other hand, attempt to justify and validate the target statement. A similar test for a support relation would be: "It is true that <target statement> because <source statement>"(Stab and Gurevych, 2014). If a support relation is found, it should be labeled as such (the "for" label in the Brat annotation tool).

4.2.5 Linking claims and premises

The first part of annotating argument relations was to systemically go through the document again, iteratively looking at each topic being discussed, in the same way as before. For each topic, we linked premises and claims in the way described above, while keeping in mind that premises can be related to claims or other premises. All the while we used the tests to ascertain what type of relation to mark. As in previous cases, it was often necessary to reformulate the sentences in order for the tests to make any sense.

Stab and Gurevych (2014) give a good summary of the process:

1. For each topic start with a claim.
2. If a premise obviously supports or attack the claim, link it to the claim.
3. For all not connected premises in the paragraph, test if it could be connected to an already connected premise. if that is not possible reformulate the premise and connect it to a matching claim or premise in the same paragraph.

If there are more than one claim connected to a topic, repeat the process until all premises concerning said topic are linked.

To reiterate, these questions will help in deciding whether two components are linked, and which relation type to assign (Stab and Gurevych, 2014):

- Is the target statement underpinned or rebutted by the source sentence?
- Is one of the following patterns meaningful?
  - support relation: it is true that <target> because <source>
  - attack relation: it is not true that <target> because <source>

4.2.6 Linking claims and major claim

Once all relations between claims and premises had been marked, the final step consisted of linking claims in all the topics being discussed, to the major claim. The major claim was usually found in the introduction or in the conclusion and should already have been identified at this point. Again we iteratively went through all the topics, and linked each claim to the major claim, and designated whether the claim was for or against the major claim. This process was fairly similar to linking claims and premises in the previous step (Stab and Gurevych, 2014). Some times the claim could be a statement relating to something other than what was being discussed in the review, in those cases it was not linked to the major claim.

4.3 Differences between persuasive essays and reviews

Major claim

In persuasive essays (Stab and Gurevych, 2017), the major claim is usually located in the conclusion or introduction to the essay. This seems to be the case with reviews as well. It seems natural to conclude with your personal opinion about the item in question in a review. However, in persuasive essays the major claim is usually expressed in a complete sentence. Reviews are not as explicit, and the form the major claim takes can vary a lot.

Claim

(Stab and Gurevych, 2017) look at one paragraph at a time and try to identify the most general claim in that paragraph. Because persuasive essays follow a stricter pattern than reviews, this does not work as well for claims in our case. Instead, it seems sensible to identify a part of the text discussing a certain topic, and identify the most general claim about that topic before moving on to the next topic being discussed and identifying the next claim.

In certain cases, the reviews have subheadings that work well as claims:

4.4 Inter-annotator agreement

For our annotation process, we had one main annotator annotating all 40 reviews of the dataset, with 2 additional annotators annotating the first 10 texts, in order to provide a measure of agreement in the annotation. We used the same metrics as in Stab and Gurevych (2017); *observed agreement*, *Fleiss' κ* Fleiss (1971) and Krippendorff's α_U, as described in chapter 3. Our inter-annotator agreement scores are shown in table 4.1. Our results are not quite up to par with the ones achieved by Stab and Gurevych (2017). This might be due to some aspect of the guidelines being unclear, or it might simply be due to the fact that reviews are in essence harder to perform argument annotation on than persuasive essays. How structured a review is depends in large part on the author, where in persuasive essays an inherent structure in the text is more or less required in order for the text to convey its point.

Component type	*Observed agreement*	*Fleiss' κ*	α_U
MajorClaim	97.5%	.505	.454
Claim	79.7%	.426	.508
Premise	79.6%	.546	.492

Table 4.1: Table showing the inter-annotation scores after 3 annotators annotated the first 10 texts of NorArg.

Chapter 5

A dataset for annotation projection

Because of the limited scope of our project, it was never feasible to produce a dataset large enough to be able to train a model exclusively on our own annotated data in the time frame of the project. The difficulty of argument annotation also means that the cost of producing such data is relatively high, and outside the scope of our project. Instead we decided to use the existing dataset of persuasive essays produced by Stab and Gurevych (2017) to train a model, then evaluate its performance on our own smaller dataset in Norwegian, which meant that we needed to find a way to make our system multilingual.

5.1 Cross-lingual NLP

Due to the importance of high quality datasets in NLP, and the significant costs in acquiring such datasets, there has been a lot of focus on cross lingual work in recent research across a range of different topics within NLP, including part-of-speech-tagging (Zhang et al., 2016), dependency parsing (Agić et al., 2016) and other fields (Eger et al., 2018).

As for the field of argument mining, most of the existing research so far has been focused on using a single language (Eger et al., 2018). Because of the complexity of the argument mining task, especially the difficulties related to producing high quality annotated datasets for the task, investigating the possibilities of cross-lingual systems could prove very beneficial.

Eger et al. (2018) set out to do exactly that. They find that existing argument mining data sets are not sufficiently homogeneous to facilitate cross lingual argumentation mining (Eger et al., 2018), and investigate the effectiveness of different translation and projection strategies for mapping data sets from one language to another. In addition, they produce several novel data sets based on the existing data set consisting of persuasive essays published by Stab and Gurevych (2017).

The two main techniques Eger et al. (2018) investigate are annotation projection and bilingual word embeddings based direct transfer. Eger et al.

	Original English phrase	English explanation shown to evaluators
1	fly out of London	take an airplane from London
2	like a bat out of hell	escaping as quickly as possible
3	out cold	unconscious
4	out of bounds	unacceptable
5	out of breath	gasping for air (for example, after running)
6	out of curiosity	because a person is casually interested in something
7	out of focus	not clear to see (blurry)
8	out of his mind	crazy
9	out of milk	the supply of milk is finished
10	out of order	does not function (broken)
11	out of pocket	paid for something from personal money
12	out of steam	no more energy (exhausted)
13	out of style	unfashionable
14	out of the closet	openly homosexual
15	out of the game	no longer participating in a game
16	out of the office	away from the office
17	out of this world	excellent
18	out of time	a deadline has passed
19	out of wedlock	between partners who are not married
20	out on the town	having a fun time going shopping or to bars/restaurants(carousing)

Table 5.1: *Phrases used in the Google Translate study (Benjamin, 2019).*

(2018) conclude that annotation projection performs considerably better than direct transfer. Annotation projection consists of mapping each word in a sentence in the source language, to a word in the same sentence in the target language, in order to transfer the annotations from the source sentence to the target sentence. The first step in applying annotation projection to our dataset is translating our data. Based on the findings from Eger et al. (2018), that translations produced by machine translation services is near as good as human produced translation, we decided to use Googles translation API for the job.

5.2 Google Translate

Eger et al. (2018) compare human translation and machine translation, and find that for the languages they use (German, English, Chinese) the resulting translations produced by human and machine translation respectively, were perceived to be of comparable (Eger et al., 2018).

5.2.1 Analysis

A recent study of Google Translate (GT) investigates its performance on 107 different languages. For each language, 20 English phrases (Table 5.1) are translated then rated by native or highly competent speakers (Benjamin, 2019).

The readers are not shown the original phrase, only the intended meaning of the phrase. The study found that reading the original phrase could cause readers to be more lenient in scoring, perhaps giving points

	Description	Scoring
Bard	A weighted ranking that indicates the proportion of translations that were judged close to human quality.	A = 5, B = 2.5, C = 0. Maximum score = 100.
Tarzan	Indicates the percentage of times that translations could be understood, regardless of whether they were judged as human quality.	A = 5, B = 5, C= 0. Maximum score = 100.
Fail	The percentage of times that translations were judged as completely wrong.	A = 0, B = 0, C = 5. Maximum score = 100.

Table 5.2: *Scoring scheme in Benjamin (2019).*

Language	Bard	Tarzan	Fail
Danish	40	70	30
Norwegian	27.5	45	55
Swedish	45	60	40

Table 5.3: *Scores for Danish, Norwegian, and Swedish (Benjamin, 2019).*

for individual words being correctly translated (Benjamin, 2019). Each translation is scored using three different scores, as shown in Table 5.2.

In the study, Norwegian translations scored lower than many other western European languages (Benjamin, 2019). Swedish and Danish, both being for the most part mutually intelligible with Norwegian, and members of the same language family, scored considerably higher (Table 5.3). The phrases that are used in the study are most likely phrases that are particularly difficult to translate, and no context was given to each expression, something that might have improved the accuracy of the translation.

5.2.2 Review corpus

As opposed to the dataset used in Eger et al. (2018) the writing style of our dataset tends to be more focused on being entertaining, with the writers often being very creative in their choice of words. This is probably due to the commercial nature of our dataset, where the authors have extra incentive to write witty and amusing texts. This writing style makes these types of phrases and expressions that GT typically performs worse on, abundant in our dataset. And, despite having plenty of context, many of these expressions found in our own texts were equally poorly translated:

(5.1) *Jeg kunne tenkt meg og sett Ashes to Ashes uten **halvgjort tidsreisetull**, og heller som en rendyrka krimserie satt til tidlig åttitall* $_{\text{Original}}$

(5.2) *I could have imagined seeing Ashes to Ashes without **half time travel duties**, and rather as a pure culture crime series set to the early eighties*

Translation

(5.3) *Russell Crowe og Denzel Washington **braker sammen** i "American Gangster", men på en overraskende fredelig måte* Original

(5.4) *Russell Crowe and Denzel Washington **break up** in "American Gangster", but in a surprisingly peaceful way* Translation

Nevertheless, we felt the translations for the most part were adequate for our use, in that they preserved most of the structures we were interested in analysing. As far as intelligibility is concerned, GT performed above our initial expectations. Most sentences were understandable and felt natural to the reader. The possibly dubious quality of the translations should however be kept in mind as a possible source of inaccuracy when assessing error factors. It did have problems with some words and word combinations that perhaps appear less frequently, but the general semantics of the sentence was usually mostly intact, and the argument structure is in most cases not significantly altered.

(5.5) *Dette er nemlig ikke en gangsterfilm med skuddvekslinger* Original

(5.6) *This is not a gangster movie with gunfire every ten minutes* Translation

It also performed consistently poorly on idioms.

(5.7) *Nå tilspisser intrigene seg , vår kjære favorittdverg er **ute i hardt vær** og serien tar en mørkere vending , for som vi alle vet :* Original

(5.8) *Now the intrigue is peaking , our dear favorite dwarf is **out in severe weather** and the series is taking a darker turn , as we all know :* Translation

5.3 Fast-align

The second part of annotation projection, is the projection itself. Because it is not a given that the order of the words is the same in the source and target language, it is necessary to apply some sort of mapping function from the sentence in the source language to the sentence in the target language in order to transfer the annotations from the source sentence. Eger et al. (2018) used Fast-align, the algorithm described in Dyer et al. (2013). Fast-align is a form of a lexical translation model (Dyer et al., 2013), and works in the following way: given a source sentence and a target sentence, a number is generated for each of the words in the target sentence. These numbers indicate which word in the source sentence the target word was translated from. The program returns a list of number pairs, the first number being the index of the source word, the second number the index of the target word. After the texts were translated and the annotations projected onto the translated texts, the amount of different components in the dataset changed slightly, as can be seen figures 5.1 and 5.2.

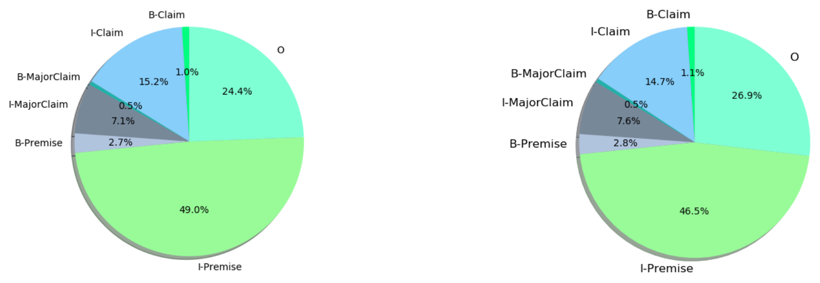


Figure 5.1: Distribution of component types for the two versions of the persuasive essay test set from Eger et al. (2017). On the left, the original version, on the right, the translated Norwegian version.

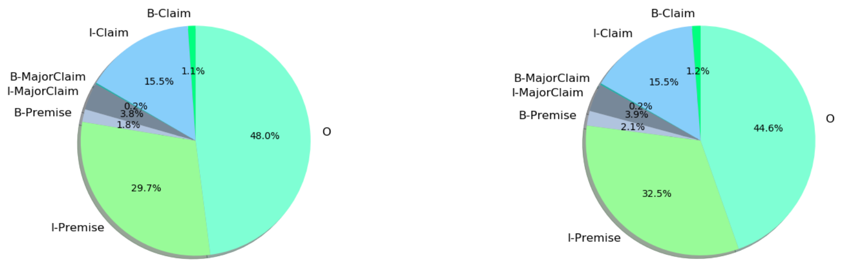


Figure 5.2: Distribution of component types for the two versions of the NoReC test set. On the left, the English translated version, on the right, the original Norwegian version.

Chapter 6

Experimental Setup

When choosing which architecture to use in our experiments, we wanted a pipeline to cover all the different tasks in argument mining; argument detection, argument segmentation, and argument structure prediction. Historically, most research has been focused on completing only one of the tasks. Eger et al. (2017) focus on achieving a complete system that encompasses all the different argument mining tasks. One of the systems they experiment with is the model from Miwa and Bansal (2016), a system designed for extracting entities and relations from sequence based data. The system can be used on any similarly framed problem, and is well suited for the task of argument mining. Therefore, for our pipeline architecture, we have chosen to use the model from Miwa and Bansal (2016) referred to in Eger et al. (2017) as the LSTM-ER, as this was the model that produced the best overall results. In addition, we investigate how other models perform. We use a model based on the NCRF++ framework for sequence labeling tasks, and a model using Multilingual BERT, to serve as comparison systems.

As described in the previous chapter, the corpus we extracted our test set from is in Norwegian, and the training set we use to train our model is in English. Our goal is to explore the effectiveness of our English trained model on a translated version of our Norwegian test dataset. This requires using techniques for cross-lingual mapping and annotation projection in order to translate our dataset to English and prepare it for testing.

In this chapter we start by describing the process to prepare our data (Illustrated in Figure 6.1). Most of the preprocessing is applied only to our own annotated test set, but some steps of the process is required for the training set used in Eger et al. (2017) as well, in order to make it suitable for our implementation of LSTM-ER. We also describe our models, NCRF++, Multilingual BERT and LSTM-ER, the modifications we made to LSTM-ER, and the tools we used to implement our own version of the model, namely the machine learning library PyTorch and a selection of its modules.

6.1 Data pre-processing

In order to convert our data into a format that is suitable for the machine learning models to use for training, and create a parallell dataset in English and Norwegian, a number of pre-processing steps are required. Figure 6.1 illustrates the process: first we select a number of texts from the NoReC corpus. Then we annotate the texts using the annotation tool brat and format the resulting files into a CoNLL-format, and complete the Norwegian version of our dataset. To create the English version of the dataset, we first translate the texts using Google Translate, and tokenise them using the UDPipe Tokeniser. In order to transfer our annotations to the English version of the dataset we use Fast-Align, which creates a mapping between two sentences in different languages. This allows us to transfer our annotations to the English sentences, and create the English version of the dataset. Finally, we obtain some additional information about our texts using the Stanford POS tagger, which is used when training the LSTM-ER. In the following sections we describe each part of the process in more detail.

6.1.1 NoReC to Brat

Our test set, as described in chapter 3, consists of screen reviews randomly selected from the NoReC corpus. The very first step in preparing our test data is formatting it for annotation in the annotation tool brat (as shown in figure 6.1). Brat comes with its own standalone server functionality which allows you to run a local server from which you can handle all your annotation, and setting it up is relatively straight forward. Since the NoReC texts come pre-tokenised and preprocessed, preparing them does not take much extra work. For each review, Brat requires a .txt file containing the review, and an empty .ann file with the same filename, so all we needed to do was to copy all the selected text files from NoReC into the designated data folders, and create a .ann file for each text file. Any annotations made using the software is then stored in a text format in the accompanying .ann-files, ready for further processing.

6.1.2 Brat to CoNLL

The next step in the data processing is converting the output files from the Brat annotation tool into a CoNLL format (figure 6.1). Eger et al. (2017) devise a scheme to convert argument component annotations into a CoNLL format, complete with BIO-tag, component type, relation type, and distance to related component (if applicable). The scheme is defined by this equation from Eger et al. (2017), where each token is assigned a

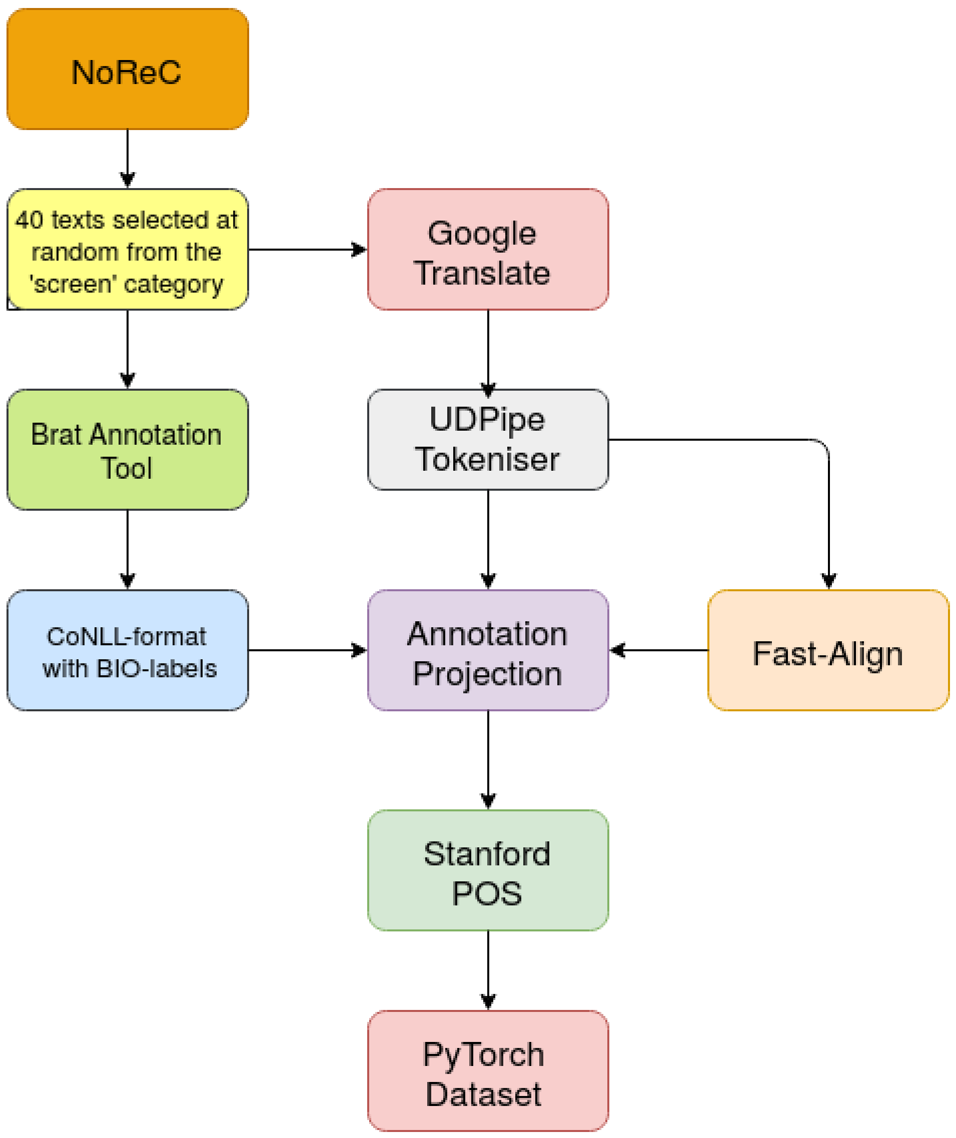


Figure 6.1: Overview of the data preprocessing.

label which is a part of Y, where

$$\begin{aligned} Y = \{(b,t,d,s) \mid b \in \{B, I, O\},& \\ t \in \{P, C, MC, \perp\},& \\ d \in \{..., -2, -1, 1, 2, ..., \perp\},& \\ s \in \{Supp, Att, For, Ag, \perp\}\}.& \end{aligned} \tag{6.1}$$

Which means that each token label has a b, indicating whether the token is not part of an argument (O), the beginning of a component (B), or inside a component (I); t tells us the type of the component the token is a part of, "P" if a premise, "C" if a claim, "MC" if a major claim, or "$\perp$"

if outside a component. *"d"* indicates the distance measured in number of argument components to the related component, and finally, *"s"* indicates the relation type or stance, "Support" or "Attack" for premises, "For" or "Against" for claims, indicating their stance towards the major claim, or "$\perp$" if not applicable (Eger et al., 2017).

Input size

The dataset of persuasive essays used in Eger et al. (2017) is for the most part written in a style that uses paragraphs to separate different subtopics of discussion, leaving each argument neatly structured into a paragraph of its own. A side effect of this is that the argumentation structures in the essays are all completely contained within one paragraph (Eger et al., 2017). This means that argument components in a paragraph are never related to components outside said paragraph, and it follows that a natural document size to use for training on the essays is one paragraph. Since the whole argumentation structure is contained within a paragraph, a paragraph is very likely to contain both claims and premises; premises don't appear in a paragraph without there being a claim present to support or attack. Claims can appear alone in paragraphs in theory, but in most cases they are accompanied by at least one premise. Because a paragraph contains fewer components than the whole essay would, it is also easier to predict relations on paragraph sized documents, as there are fewer possible component pairs to choose from. The dataset used in Eger et al. (2017) contains 2235 paragraphs in total, with each paragraph having an average length of 66 tokens (Eger et al., 2017). In theory, using the whole essay as a document size allows for relations spanning the entire length of the document. However, Eger et al. (2017) found no such relations in their dataset. In fact, in the persuasive essay dataset, 30% of all related components immediately follow the component they are related to, and two thirds of all relations are less than 3 components apart (Eger et al., 2017).

Intuition also tells us that most authors would prefer to keep their claims and premises close to each other, in order to not confuse the reader, ideally containing their argumentation about a given subtopic within a single paragraph. Eger et al. (2017) created two versions of their dataset. One where the document size is the paragraph, and another where each essay is an input, and used both versions in some of their experiments. For the most part, the models trained and tested on the paragraph version performed better (table 6.1).

The first obstacle then, in converting the test set to the CoNLL format used in Eger et al. (2017) (Table 6.2), consisted of splitting the text into paragraphs. As discussed earlier, the structure of reviews is less clear than that of persuasive essays, and producing a paragraph separated version of our dataset similar to the original dataset took some thinking. The NoReC dataset already contains information to separate the reviews into paragraphs, but we found that the division didn't really separate each argument into its own paragraph in the same way it did in the persuasive

Models	**Paragraph level**						
	Acc.	C-F1		R-F1		F1	
		100%	50%	100%	50%	100%	50%
MST-PARSER	31.23	0	6.90	0	1.29	0	2.17
Mate	22.71	2.72	12.34	2.03	4.59	2.32	6.69
Kiperwasser	52.80	26.65	61.57	15.57	34.25	19.65	44.01
LSTM-Parser	55.68	58.86	68.20	35.63	40.87	44.38	51.11
STagBLCC	59.34	66.69	74.08	39.83	44.02	49.87	55.22
LSTM-ER	**61.67**	**70.83**	**77.19**	**45.52**	**50.05**	**55.42**	**60.72**
ILP	60.32	62.61	73.35	34.74	44.29	44.68	55.23
	Essay level						
STagBLCC	**60.46**	63.23	69.49	**34.82**	**39.68**	**44.90**	**50.51**
LSTM-ER	54.17	**66.21**	**73.02**	29.56	32.72	40.87	45.19

Table 6.1: This table from Eger et al. (2017) shows their results from running various models on the persuasive essay corpus. From the top, MST-PARSER and Mate are feature-based dependency parses, Kiperwasser and LSTM-Parser are neural dependency parsers, STagBLCC and LSTM-ER are neural sequence taggers, and ILP is the feature based model from Stab and Gurevych (2017), acting as a comparison system (Eger et al., 2017). Highest scores are in bold. C-F1 indicates component prediction score, R-F1 relation prediction score. The scores clearly indicate that most systems perform better on paragraph sized documents. Most argument structures in persuasive essays are completely contained within the paragraph unit. Few relations go outside the paragraph they originate from, which might explain why there are less errors made when using the paragraph as a document size. The paragraph size also limits the amount of possible targets for a given relation, reducing the probability of wrong predictions. Only the STagBLCC and LSTM-ER models were run with essays as input documents, because the task was too memory heavy for the dependency parsers (Eger et al., 2017). However, the scores from STagBLCC and LSTM-ER indicate that paragraph units produce better results overall.

essays, so we decided to use a different approach. Based on the idea that a change of paragraphs also signifies a change of topic being discussed, we decided to center each paragraph around one claim. Once a new claim was encountered, or a premise linking to a new claim (or linking to another premise, which recursively linked to a new claim), we started a new paragraph. This problem forced some new considerations in the outline of the annotation guidelines; namely, a premise can not link to claim in a paragraph other than the one it itself is part of. In practice, this lead to few changes, premises rarely relate to far off claims without there being another claim closer in distance that is more or less equivalent in meaning. Once we solved the problem of splitting the text into paragraphs, the rest was

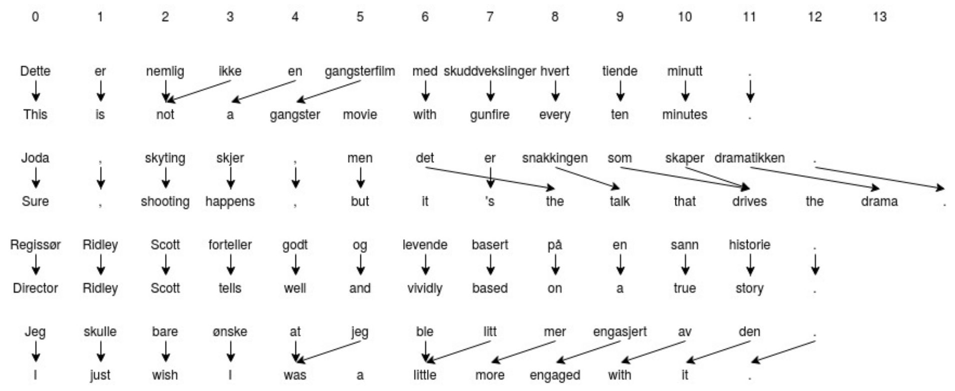

0
1
2
3
4
5
6
7
8
9
10
11
12
13
Dette er nemlig ikke en gangsterfilm med skuddvekslinger hvert tiende minutt .
This is not a gangster movie with gunfire every ten minutes .
Joda , skyting skjer , men det er snakkingen som skaper dramatikken .
Sure , shooting happens , but it 's the talk that drives the drama .
Regissør Ridley Scott forteller godt og levende basert på en sann historie .
Director Ridley Scott tells well and vividly based on a true story .
Jeg skulle bare ønske at jeg ble litt mer engasjert av den .
I just wish I was a little more engaged with it .

Figure 6.2: Cross-lingual mapping for the sentences in table 6.4 and alignments in 6.3. A good example of how the alignments work are the mappings of both "nemlig" and "ikke" to the word "not" in the first sentence pair. The alignments are not always as accurate as we would want however. For instance, in the first sentence, the word "gangsterfilm" is aligned with the word "gangster", even though it might seem better if it were aligned to the word "movie", or ideally, to both.

not too complicated. The CoNLL-format used by Eger et al. (2017) displays one token per line, along with its BIO-tag, argument component type, and relation type if applicable, following the sequence tagging framework defined in equation 6.1, with an example in table 6.2.

6.1.3 Google translate

As mentioned in chapter 5, we used Google Translate to translate our Norwegian corpus into English. This is done in a parallell branch of the data processing, as illustrated in figure 6.1. As was also discussed in chapter 5, the translations are not always as good as we had hoped they would be, but we refrained from changing them manually. For the translation we used the Google Cloud Translation API[1]. We also used the UDPipe tokeniser (Straka and Straková, 2017) with the help of the very useful UDPipe toolkit for Python[2], to tokenise both our Norwegian and English sentences after translation.

6.1.4 Fast-Align

The next step, as shown in figure 6.1, was preparing the sentences for alignment. The Fast-Align (Dyer et al., 2013) system requires a particular type of format for its input; the source and target language sentences separated by three vertical pipes, as illustrated in table 6.4. When given the four sentences in table 6.4 as input, Fast-Align produces the alignments

[1]https://cloud.google.com/translate/docs
[2]https://github.com/eivindbergem/ud-toolkit

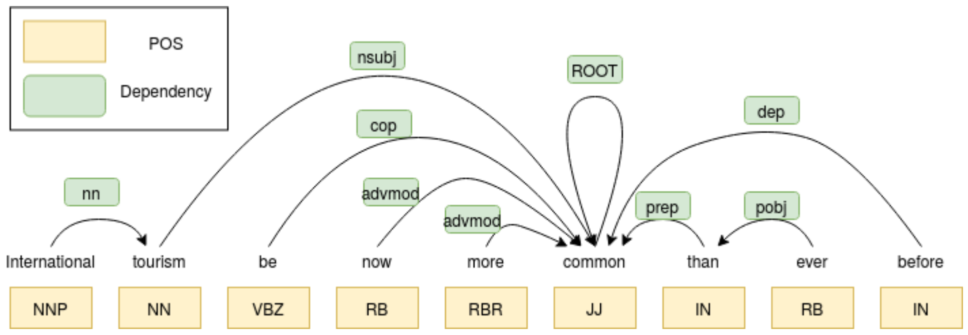


Figure 6.3: Dependency graph and POS tags for the sentence *International tourism is now more common than ever before*, described in table 6.5, using the lemmatised version of the words.

in figure 6.2, where the first number in each number pair indicates the index of the word in the source language, and the second number indicates the index of the word in the target language the first word translates to. The finished alignment of the sentences in table 6.4 is shown in figure 6.2, where each word in the sentence from the source language has an arrow pointing to a word in the target language sentence, which corresponds to the number pair mapping produced by Fast-Align.

6.1.5 Stanford POS

The LSTM-ER model used in Eger et al. (2017) is based on the model from Miwa and Bansal (2016). The first layer of the model takes inputs comprised of embeddings of words and their parts of speech (POS). In order to acquire POS tags and dependencies (which are required for a later step in the pipeline) for their dataset, Miwa and Bansal (2016) employ the Stanford neural dependency parser[3] (Chen and Manning, 2014). We used scripts[4] from Eger et al. (2017) to convert our test set to the format required by the Stanford parser. The output from the Stanford parser, for the sample sentence "International tourism is now more common than ever before" is shown in table 6.5, corresponding to the POS and dependencies illustrated in figure 6.3.

6.1.6 Modifying the data for our model

We use the same training set as the one used in Eger et al. (2017). However, because we implemented our own version of the LSTM-ER from Eger et al. (2017), some further pre-processing of the data was required. For both our test set and the training set from Eger et al. (2017), we followed the

[3]https://stanfordnlp.github.io/CoreNLP/
[4]https://github.com/UKPLab/acl2017-neural_end2end_am/tree/master/progs/LSTM-ER

instructions and used the scripts described in Eger et al. (2017), and parsed the resulting Stanford POS files into suitable datafiles for our PyTorch implementation. The Stanford files contain several types of information; lemmatised tokens, part of speech tags and dependency relations, as well as dependency targets. Lemmatisation is the practice of reducing a word to its base form. For instance, a lemmatised version of "was" is "be", as can be seen in figure 6.5. The idea behind it is that it gives a more general form of the word, meaning that a potential model parsing the words understands that the words "was" and "is", for instance, serve much of the same purpose in a context. For our implementation, we disregarded using the lemmatised tokens because it was unclear whether they had used them in Eger et al. (2017). We instead extracted the tokens from the accompanying textfiles. Using the lemmatised version of the tokens is something that could be included in potential future work.

6.1.7 PyTorch

Because of recent successful innovations in the field of NLP like word embeddings (Mikolov et al., 2013), it has become increasingly important to use deep learning techniques as opposed to more traditional methods when doing NLP. In recent years, several of the world's largest tech companies have joined the race to create a good machine learning library. PyTorch (Paszke et al., 2019) is an open source machine learning library developed mainly by Facebook's Research AI lab. PyTorch provides easy to use implementations of the most popular machine learning models, and makes it relatively easy to modify these models to suit your purpose. For our system we mainly used the LSTM module[5], making some modifications to fit it to our dataset and hyperparameters. For loading our data into PyTorch, we used torchtext, a fairly recent addition to the library. It provides an interface for loading and preprocessing textual data. Among other things, it provides tools for batching data, padding inputs to the same length, preprocessing documents by applying tokenising and similar procedures. It also has built in methods for numericalising and converting your input text to embeddings. In addition, it provides easy access to the most commonly used word embeddings.

6.2 Neural Modeling

As previously mentioned, when considering what type of model to use for our project, we ended up deciding to find a system that could perform all of the tasks in the argument mining pipeline. Those tasks, as described in chapter 2, are the following: argument detection, argument segmentation, and argument structure prediction. Eger et al. (2017) describe several different approaches to achieving this type of system; framing the problem as a dependency parsing problem, sequence tagging problem, a multi-task tagging problem, and using a combined sequence tagging and tree

[5]https://pytorch.org/docs/stable/nn.html#lstm

structure model. Among all the models tested in Eger et al. (2017), the best performing model on paragraph size inputs (table 6.1) was the combined sequence tagging and tree structure model, named LSTM-ER, based on a model by Miwa and Bansal (2016). The architecture of LSTM-ER is illustrated in figure 6.4.

6.2.1 LSTM-ER

LSTM-ER is based on a general model for extracting entities and identifying relations between them, by Miwa and Bansal (2016). The architecture of LSTM-ER consists of three parts; an embedding layer that converts raw inputs to embeddings, a sequence layer that predics labels given the word and POS tag embeddings, and a dependency layer that receives dependency and label embeddings and predicts relations between different argument components. Figure 6.4 illustrates the LSTM-ER model architecture. Due to the limited time of our project, we decided to focus on the first two layers of the model, the part of the model that predicts component labels. Our dataset contains the information types required to complete the entire model, which would make it easier for a possible future work to implement relation prediction as well.

Embedding layer

The first part of LSTM-ER is an embedding layer, where each token and its POS tag (obtained from the Stanford POS tagger) are converted into embeddings. Following Eger et al. (2017), we used the pretrained GloVe (Pennington et al., 2014) embeddings with a dimension size of 50. These embeddings are easy to download and apply to our input tokens using the Torchtext library described in section 6.1.7. The POS embeddings are randomly initialised based on the POS tags, using PyTorch with a dimension size of 25. The word embeddings and POS embeddings are then concatenated and sent to the sequence layer. The embedding layer also handles embeddings for labels and dependencies, which are both randomly initialised using a normal distribution between 0 and 1, with dimensions of 100 and 25, respectively.

Sequence layer

The sequence layer consists of a bi-directional LSTM (chapter 2) which is ideal for working on sequences of inputs, in our case, sentences. The input is comprised of the word embeddings concatenated with the POS embeddings.

Hyperparameters

In machine learning, hyperparameters is a term that describes parameters that are used to optimise the learning process of the model. A typical way to decide the optimal values of hyperparameters is through trial and error,

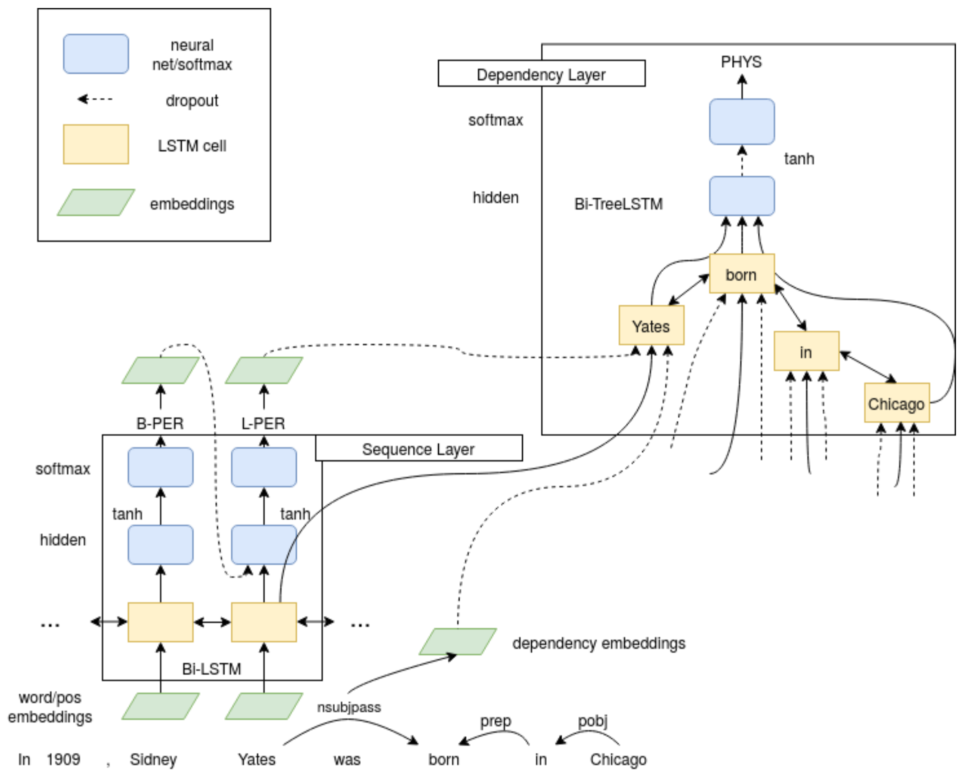


Figure 6.4: LSTM-ER as described in Miwa and Bansal (2016)

testing different combinations of values for the parameters, and finding a combination that produces the best results, based on different metrics. Miwa and Bansal (2016) tuned their hyperparameters on several different development sets and tested a range of different settings before ending on the settings shown in table 6.6. In this section we will explain some of the different settings used in Eger et al. (2017). Eger et al. (2017) did less hyperparameter tuning on the LSTM-ER model than on their other models, citing the fact that the LSTM-ER has more regularisation techniques than the others, namely scheduled sampling and entity pretraining, as the reason why they deemed it less necessary for the LSTM-ER to be fine-tuned. Eger et al. (2017) also mention that LSTM-ER took a lot longer to train.

Dropout

In Miwa and Bansal (2016), dropout is applied to the embedding layer, and to the layers for classifying entities and relations (Miwa and Bansal, 2016) (Figure 6.4). In Eger et al. (2017) however, only the embedding dropout and dropout for the relation classifying layer is applied, so we follow suit and do the same. And because our model doesn't include the final relation classification layers, the only dropout we apply is the input dropout, which

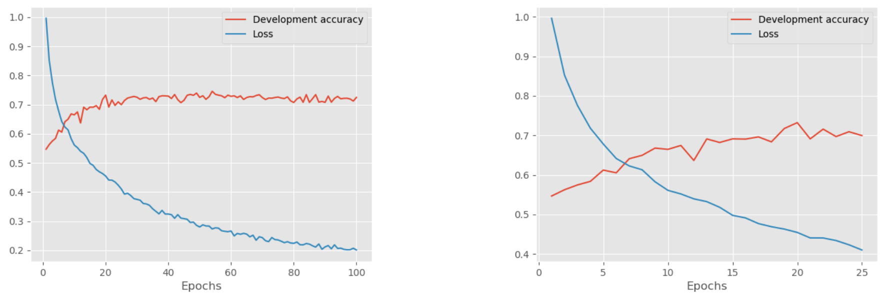


Figure 6.5: Development set accuracy and average loss per epoch from training LSTM-ER on the original training set for 100 and 25 epochs.

is set to a value of 0.5 in Eger et al. (2017). This means that there is a 50% chance for a given unit to be dropped. The PyTorch implementation of dropout[6], samples from a Bernoulli distribution as described in Srivastava et al. (2014).

Training

This section describes the process of training our LSTM-ER on the training set of persuasive essays from Eger et al. (2017), using the accuracy on the development set as an indicator of the models performance. The results we achieved were not as good as we had initially hoped, but there are a number of factors that might help account for this discrepancy, which we will discuss in greater detail in the next chapter. Where Eger et al. (2017) decided to train their model for 100 epochs, judging from our loss and accuracy, that seemed not to be necessary for our training. As figure 6.5 indicates, judging by how early the growth in accuracy on the development set peters out, training for 100 epochs might only cause us to overfit the model on the training set. In fact, our model seems to reach the same level of score after only 25 epochs, as can be seen in figure 6.5. The fact that our model requires fewer epochs is probably a side effect of only using a simplified version of the model. In the original model, the label embeddings are shared between the label classification task and the relation prediction task, and training and fine-tuning them is a more complicated task than in our case, where they are only used for the label classification task.

6.2.2 NCRF++

NCRF++ (Yang and Zhang, 2018) is a toolkit written in Python that allows you to implement different neural models for sequence labeling tasks. The toolkit allows for easy modification of the model architecture through the

[6]https://pytorch.org/docs/stable/nn.html?highlight=dropout#torch.nn.Dropout

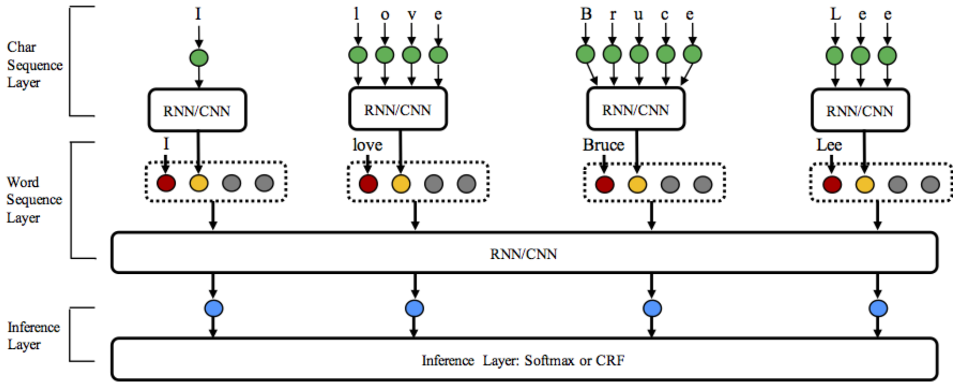


Figure 6.6: Illustration of the architecture of the NCRF++ framework from https://github.com/jiesutd/NCRFpp

use of configuration files, which means you can change the model setup and hyperparameters virtually without coding anything yourself.

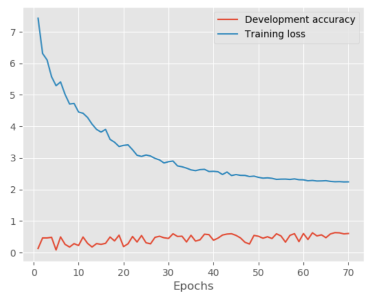
When used "out of the box", NCRF++ consists of three layers, a character sequence layer, a word sequence layer and an inference layer. The model architecture is illustrated in figure 6.6. The character layer and word layers can be either CNNs or RNNs, or various subtypes of the two architectures. The inference layer can be a Softmax layer or a CRF layer (Yang and Zhang, 2018). It is also possible to exchange any of these layers with your own implementations fairly easily. For our experiments we used the default settings for the basic architecture, which means a CNN for the character sequence layer, an LSTM in the word sequence layer, and a CRF in the inference layer.

Hyperparameters

Because of the limited time available for hyperparameter search, we restricted our search to different values of learning rates, number of epochs, batch size and LSTM layers, as shown in table 6.7. In the end we chose the settings that gave the best results on the various development sets, displayed in table 6.8. All the models used a dropout of 0.5. For the Norwegian data set we used GloVe embeddings trained on Norsk Aviskorpus, and the NoWaC and NBDigital corpora. When training on the English dataset we used GloVe 300 dimension embeddings trained on Wikipedia and the Gigaword corpora.

Training

When training our NCRF models, we tried a number of different settings, as displayed in table 6.7. The configuration that proved to give the best performance among the limited settings we tested, had a number of epochs set to 70, with a learning rate of 1e-3, batch size of 16, and 2 LSTM layers for



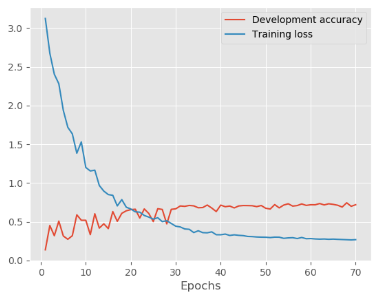


Figure 6.7: Development set accuracy and average loss per epoch on the Norwegian (left) and English (right) training sets. The model trained on the Norwegian data had a learning rate of 0.001, batch size 8 and two LSTM layers, and was trained for 70 epochs. The model trained on the English data set had the same settings except for a batch size of 16. (The configurations are listed in table 6.8)

the model trained on the English data set, and the same settings except for a batch size of 8 for the model trained on the Norwegian data. (Although the difference in accuracy between the model with a batch size of 8 and the model with a batch size of 16 was not very large). (Table 6.8) When looking at the graphs in figure 6.7, it seems clear that the model had a much harder time learning the structures of the Norwegian data set, and the model trained on the Norwegian data had a much lower performance as a result. We will discuss the possible reasons behind this in more detail in chapter 7.

Vocabulary

The Norwegian data set has a vocabulary that consists of 12935 unique words. For the model trained on the Norwegian data set, using the aforementioned GloVe embeddings trained on Norsk Aviskorpus and the NoWaC and NBDigital corpora, 704 words were unknown, meaning they did not occur in the corpora the embeddings were trained on. The English version of the data set has a vocabulary size of 9779. When using the GloVe embeddings trained on Wikipedia and the Gigaword corpora, there were 234 unrecognised words in our data set.

6.2.3 Multilingual BERT

Multilingual BERT is trained on the 100 largest languages on Wikipedia, which includes both Norwegian and English. The model is available through the huggingface/transformers library[7], which provides an easy-to-use interface to load and run the model.

[7]https://huggingface.co/transformers/multilingual.html#bert

Hyperparameters

Again we were limited by time, but we tested a small set of different hyperparameters to find the best model, shown in table 6.7, before ending on the models with the highest performance, using the settings shown in 6.8.

84	Minst	B-Claim
85	like	I-Claim
86	viktig	I-Claim
87	er	I-Claim
88	den	I-Claim
89	som	I-Claim
90	nasjonal	I-Claim
91	myte	I-Claim
92	:	O
93	Den	B-Premise:-1:Support
94	markerte	I-Premise:-1:Support
95	på	I-Premise:-1:Support
96	mest	I-Premise:-1:Support
97	mulig	I-Premise:-1:Support
98	brutalt	I-Premise:-1:Support
99	vis	I-Premise:-1:Support
100	brudd	I-Premise:-1:Support
101	med	I-Premise:-1:Support
102	den	I-Premise:-1:Support
103	«	I-Premise:-1:Support
104	uskyldige	I-Premise:-1:Support
105	»	I-Premise:-1:Support
106	og	I-Premise:-1:Support
107	idealistiske	I-Premise:-1:Support
108	hippietiden	I-Premise:-1:Support
109	i	I-Premise:-1:Support
110	California	I-Premise:-1:Support
111	.	O
112	Tilværelsen	B-Premise:-1:Support
113	var	I-Premise:-1:Support
114	nå	I-Premise:-1:Support
115	ikke	I-Premise:-1:Support
116	engang	I-Premise:-1:Support
117	peace	I-Premise:-1:Support
118	&	I-Premise:-1:Support
119	love	I-Premise:-1:Support
120	.	O
121	Dertil	B-Premise:-3:Support
122	kom	I-Premise:-3:Support
123	det	I-Premise:-3:Support
124	skremmende	I-Premise:-3:Support
125	faktum	I-Premise:-3:Support
126	at	I-Premise:-3:Support
127	drapsmannen	I-Premise:-3:Support
128	manipulerte	I-Premise:-3:Support
129	både	I-Premise:-3:Support
130	media	I-Premise:-3:Support
131	og	I-Premise:-3:Support
132	politiet	I-Premise:-3:Support
133	:	O
134	Han	B-Premise:-1:Support
135	sendte	I-Premise:-1:Support
136	stadige	I-Premise:-1:Support
137	meldinger	I-Premise:-1:Support
138	til	I-Premise:-1:Support
139	aviser	I-Premise:-1:Support
140	og	I-Premise:-1:Support
141	TV-stasjoner	I-Premise:-1:Support
142	.	O
143	Og	B-Premise:-2:Support
144	sørget	I-Premise:-2:Support
145	selv	I-Premise:-2:Support
146	for	I-Premise:-2:Support
147	helt	I-Premise:-2:Support
148	ualminnelig	I-Premise:-2:Support
149	offentlighet	I-Premise:-2:Support
150	omkring	I-Premise:-2:Support
151	drapene	I-Premise:-2:Support
152	han	I-Premise:-2:Support
153	utførte	I-Premise:-2:Support
154	.	O
1	Filmen	B-MajorClaim
2	er	I-MajorClaim
3	rett	I-MajorClaim
4	og	I-MajorClaim
5	slett	I-MajorClaim
6	glimrende	I-MajorClaim
7	;	I-MajorClaim
8	spennende	I-MajorClaim
9	,	I-MajorClaim
10	omfattende	I-MajorClaim
11	,	I-MajorClaim
12	uhyre	I-MajorClaim
13	velspilt	I-MajorClaim

Table 6.2: Example of the different BIO-tags in the CoNLL format used in Eger et al. (2017).

0-0 1-1 2-2 3-2 4-3 5-4 6-6 7-7 8-8 9-9 10-10 11-11
0-0 1-1 2-2 3-3 4-4 5-5 6-8 7-7 8-9 9-11 10-11 11-13 12-14
0-0 1-1 2-2 3-3 4-4 5-5 6-6 7-7 8-8 9-9 10-10 11-11 12-12
0-0 1-1 2-2 3-3 4-4 5-4 6-6 7-6 8-7 9-8 10-9 11-10 12-11

Table 6.3: Output from running fast-align on the four sentences in table 6.4

Dette er nemlig ikke en gangsterfilm med skuddvekslinger hvert tiende minutt .	This is not a gangster movie with gunfire every ten minutes .
Joda, skyting skjer, men det er snakkingen som skaper dramatikken	Sure, shooting happens, but it 's the talk that drives the drama .
Regissør Ridley Scott forteller godt og levende basert på en sann historie .	Director Ridley Scott tells well and vividly based on a true story .
Jeg skulle bare ønske at jeg ble litt mer engasjert av den .	I just wish I was a little more engaged with it .

Table 6.4: Example of four sentences translated from Norwegian to English, formatted for Fast-Align.

0	57	sentence id="s0"	parse_status="success"		
0	13	tok id="t0"	base="International"	pos="NNP"	nn="t1"
14	21	tok id="t1"	base="tourism"	pos="NN"	nsubj="t5"
22	24	tok id="t2"	base="be"	pos="VBZ"	cop="t5"
25	28	tok id="t3"	base="now"	pos="RB"	advmod="t5"
29	33	tok id="t4"	base="more"	pos="RBR"	advmod="t5"
34	40	tok id="t5"	base="common"	pos="JJ"	ROOT="ROOT"
41	45	tok id="t6"	base="than"	pos="IN"	prep="t5"
46	50	tok id="t7"	base="ever"	pos="RB"	pobj="t6"
51	57	tok id="t8"	base="before"	pos="IN"	dep="t5"

Table 6.5: Stanford parser output from parsing the sentence *International tourism is now more common than ever before*. From left to right, the first two numbers indicate the token position in the text, the next column gives the token id, *base* indicates the lemmatised form of the token, *pos* is the part of speech using Penn treebank tags, and the last column indicates the dependency relation type of the current token as well as the token id of its target token. The dependency graph of the sentence is shown in figure 6.3.

	Dimensions
Embeddings	
Words	200
Part of speech tags	25
Dependencies	25
Entities	25
Intermediate layers	
SequenceLSTM	100
TreeLSTM	100
Hidden Relation	100
Hidden Entity	100

Table 6.6: Experiment settings used in Miwa and Bansal (2016)

NCRF++	
Learning rate	1e-1, 1e-2, 1e-3, 1e-4, 1e-5
Epochs	30, 50, 70
Batch size	8, 16
LSTM layers	1, 2
Multilingual BERT	
Learning rate	1e-5, 1e-6
Epochs	10, 25

Table 6.7: Hyperparameter values used in our experiments with NCRF++ and BERT.

Training set	Learning rate	Epochs	Batch size	LSTM layers
NCRF++				
English set	1e-3	70	16	2
Norwegian set	1e-3	70	8	2
Multilingual BERT				
English set	1e-5	25	-	-
Norwegian set	1e-5	25	-	-

Table 6.8: Settings used for the final NCRF++ and Multilingual BERT models

Chapter 7

Results

In this chapter we give an overview of the results of the different experiments we have run. Our goal was to investigate the possibilities of using a model trained on a training set in English, on a dataset in Norwegian. In the end we tried two different approaches. First we used Google Translate and Fast-Align to translate our Norwegian test set into English. Then we used the same techniques to translate the training and test sets used in Eger et al. (2017) from English to Norwegian. This second approach allowed for further evaluation through comparison, comparing the results from models trained on the Norwegian and English versions of the training data.

There are several potential error factors originating from using these techniques to translate and project annotations, and we attempt to give an overview of them at the end of this chapter. The fact that our test set is comprised of text from a significantly different domain than the data the model is trained on, is also very likely to be a large cause of lower scoring results. We trained three different model architectures on the training sets as described in the previous chapter, before evaluating them on our test sets. In this chapter we begin by discussing the results of our experiments on the English data set, then move on to the Norwegian set. Our model architectures are the modified LSTM-ER, NCRF++, and Multilingual BERT, as described in chapter 2.

7.1 Evaluation

Eger et al. (2017) use the evaluation metric described in Persing and Ng (2016) where they measure the F1-score on how many complete argument component matches are found. They use two different ranges of overlap between predicted argument components and true components to define a complete match; 100% overlap and 50% overlap. Before testing for overlap, they apply some heuristics to make modifications to their predicted labels. There are two main rules they follow when correcting the predicted labels. First, every component must start with a "B"-tag. Second, a component can only contain one component type. We applied the same modifications to our own results. Their scores and the various models they tested are shown in tables 6.1. In addition to the scores used by Eger et al. (2017), we give the results for label classification, when looking at each predicted token label.

7.2 LSTM-ER

The first model we look at is our implementation of the LSTM-ER from Eger et al. (2017). In this section we begin by looking at the results from the model trained on the English dataset, then look at the results from the model trained on the Norwegian dataset.

7.2.1 English test sets

The initial part of our experiments consisted of testing the model trained on the original English persuasive essay data set on the original test set and on the translated version of NorArg.

Persuasive essay test set

Our results on the persuasive essays test set using the LSTM-ER were comparable to the scores achieved by Eger et al. (2017), on the task of component identification (Table 7.18). Some more detailed information about the scores for the different component types can be seen in the per class scores in figure 7.1 and the confusion matrix in 7.2. The numbers the two figures are based on can be seen in Table 7.1.

Figure 7.1 shows that the highest F1-score is on the *I-premise* tag, followed by the score for the *O* tag. These two tags are also the most frequent occurring tags in the dataset, which might explain why the model is especially good at classifying them. For the *B-premise* and *I-premise* tags, both their recall scores are higher than their precision scores, meaning that the model managed to label most of the true *I-premises* and *B-premises* that exist in the text, but in the process they mislabeled many other components. This could mean that the model has learned that in most cases it is beneficial to guess that a word is in a *Premise* component, simply because premises are so prevalent in the data.

For both of the *MajorClaim* tags, precision is much higher than recall. This might be explained by the fact that they do not occur very often in the dataset, by definition at most once per essay. Because of this the model rarely labels something as *MajorClaim*, because it is such a rare occurrence. Looking at the confusion matrix in figure 7.2, 854 of the true *I-MajorClaim* words are correctly tagged as *I-MajorClaim* while 601 are misclassified as *I-Claim*, and 485 as *I-Premise*. Which intuitively makes sense, as the structure of the three components is not necessarily very different. Premises and claims can some times be very similar, and some times the only difference between them is their role in a given paragraph (where a premise can be thought of as a subclaim supporting the main claim, while it itself is supported by other premises). Similarly for the *B-MajorClaim* tag, the main

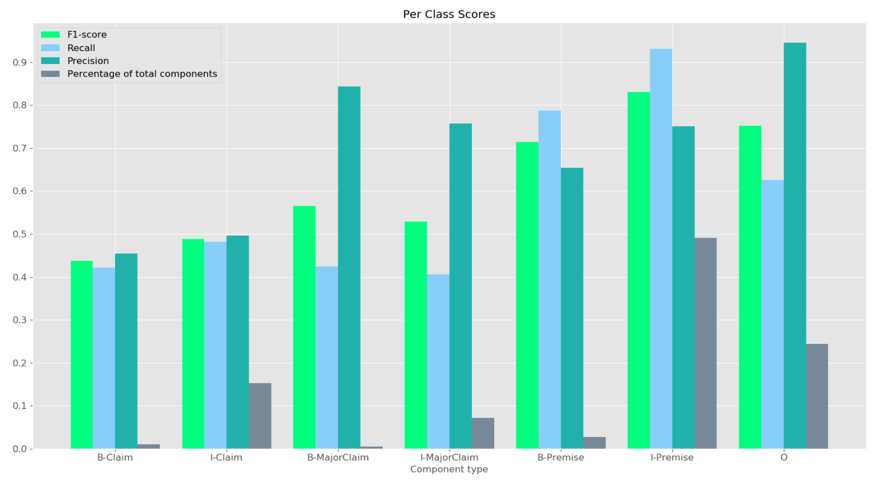


Figure 7.1: Recall, F1-score, precision, and percentage of total component amount per component class. The model used is LSTM-ER trained on the English training set, and the test set is the English version of the persuasive essays test set.

	Precision	**Recall**	**F1-score**	**Support**
I-Premise	0.75	0.93	0.83	14487
O	0.95	0.63	0.75	7206
I-Claim	0.50	0.48	0.49	4493
I-MajorClaim	0.76	0.41	0.53	2101
B-Premise	0.65	0.79	0.71	809
B-Claim	0.45	0.42	0.44	304
B-MajorClaim	0.84	0.42	0.57	153
Micro average	0.74	0.74	0.74	29553
Macro average	0.70	0.58	0.62	29553
Weighted average	0.75	0.74	0.73	29553

Table 7.1: Results of testing LSTM-ER trained on the English training set on the English version of the persuasive essay test set.

tag it is confused with is the *B-Claim* tag, followed by the *B-Premise* tag. For the *Claim* tags, the models seems to confuse them with the corresponding *Premise* tags. Roughly half of the time it chooses the wrong one, for both *B-claim* and *I-claim*. This is likely due to the high amount of premise components compared to all the other components in the dataset, as well as the similarities between premises and claims, as mentioned earlier.

NorArg test set

Next we tested the LSTM-ER on our translated test set of screen reviews, NorArg. This did not give close to the same results as testing on the

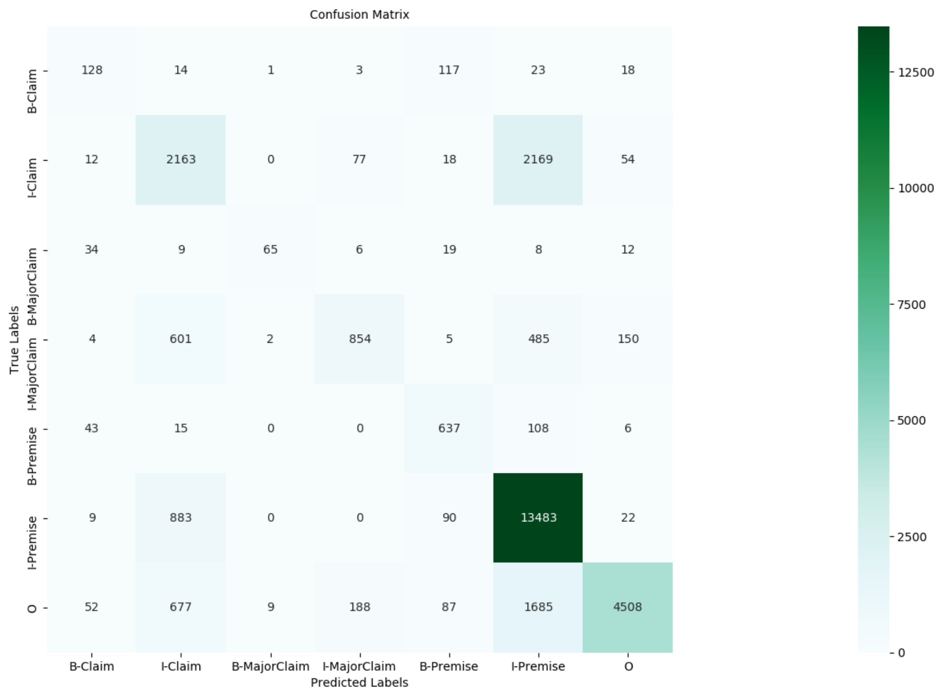


Figure 7.2: Confusion matrix of the predicted and true labels for argument mining components on the English persuasive essays test set. The model used is trained on the English version of the training set for 25 epochs.

	Precision	**Recall**	**F1-score**	**Support**
I-Premise	0.32	0.90	0.47	5745
O	0.52	0.07	0.12	9284
I-Claim	0.26	0.07	0.11	3005
I-MajorClaim	0.00	0.00	0.00	733
B-Premise	0.21	0.48	0.29	345
B-Claim	0.15	0.04	0.06	210
B-MajorClaim	0.00	0.00	0.00	38
Micro average	0.32	0.32	0.32	19360
Macro average	0.21	0.22	0.15	19360
Weighted average	0.39	0.32	0.22	19360

Table 7.2: Results of testing LSTM-ER trained on the English training set on the English version of NorArg.

persuasive essays set, which was not completely unexpected. There are a number of potential error factors in the process of translating the texts and projecting the annotations that might help to explain this large drop in performance. Not to mention the difference in the domain of our set and the persuasive essays. Judging by the results in figures 7.3 and 7.4, it seems the model has not learned to distinguish components very well at

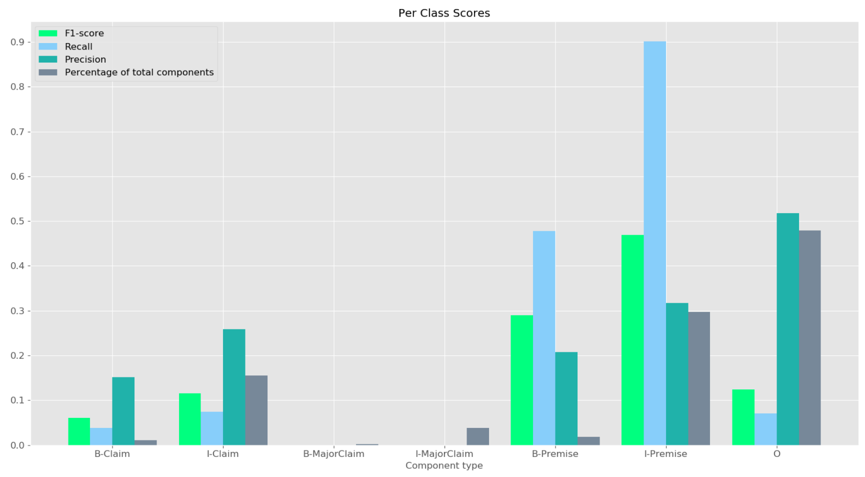
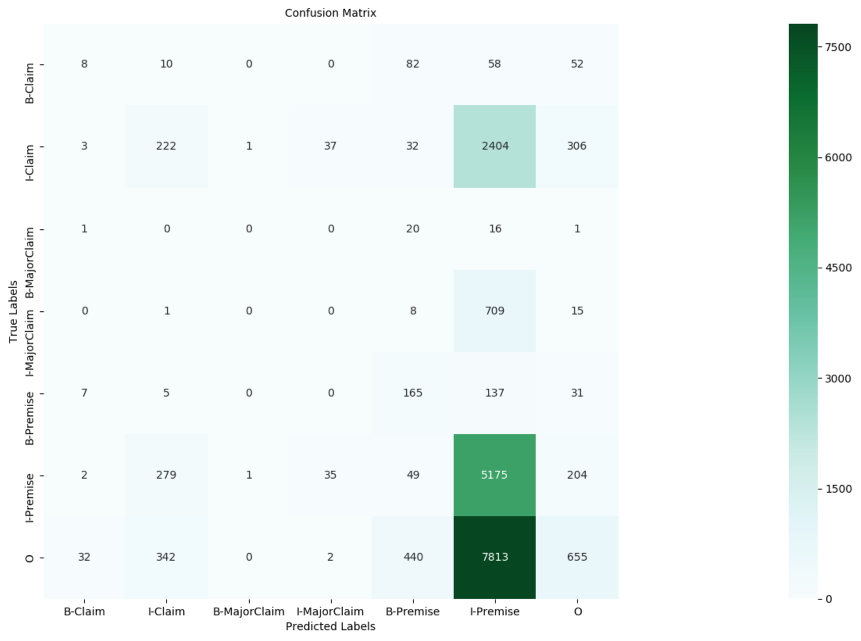


Figure 7.3: Recall, F1-score, precision, and percentage of total component amount per component class. The model used is LSTM-ER trained on the English training set, and the test set is the English version of NorArg.



	B-Claim	I-Claim	B-MajorClaim	I-MajorClaim	B-Premise	I-Premise	O
B-Claim	8	10	0	0	82	58	52
I-Claim	3	222	1	37	32	2404	306
B-MajorClaim	1	0	0	0	20	16	1
I-MajorClaim	0	1	0	0	8	709	15
B-Premise	7	5	0	0	165	137	31
I-Premise	2	279	1	35	49	5175	204
O	32	342	0	2	440	7813	655

Figure 7.4: Confusion matrix of the predicted and true labels for argument mining components on the English version of NorArg. The model used is trained on the English version of the training set for 25 epochs.

all, and that it simply predicts the most common class in the set (*I-Premise*), which can be more clearly seen in the confusion matrix in figure 7.4, and the actual numbers in table 7.2. In fact, the premise tag is the most predicted tag for every class; *B-Premise* is the most predicted tag when the gold label

	Precision	**Recall**	**F1-score**	**Support**
I-Premise	0.75	0.85	0.80	12984
O	0.74	0.71	0.73	7511
I-Claim	0.46	0.40	0.43	4100
I-MajorClaim	0.58	0.56	0.57	2117
B-Premise	0.37	0.15	0.22	782
B-Claim	0.21	0.05	0.08	298
B-MajorClaim	0.25	0.06	0.10	152
Micro average	0.69	0.69	0.69	27944
Macro average	0.48	0.40	0.42	27944
Weighted average	0.67	0.69	0.68	27944

Table 7.3: Results of testing LSTM-ER trained on the Norwegian training set on the Norwegian version of the persuasive essay test set.

is *B-Claim* or *B-MajorClaim*, and *I-Premise* for every other label. This is also reflected in the very low macro F1-score it achieves as shown in table 7.2.

7.2.2 Norwegian test sets

For our second round of experiments using LSTM-ER, we translated the English training set to Norwegian using the procedure described in chapter 6.

Persuasive essay test set

Our results on the Norwegian version of the Persuasive essay test set are quite a bit worse than on the English version of the set, although they were about as good as expected from what we saw during the training. The scores resembled the relationship between development set scores and test set scores from training and testing on the English version of the set. There are many possible reasons for why the scores are lower than on the English test set, the most obvious reason being that the translation and annotation projection process is inaccurate enough to account for the loss in performance after the translation.

One interesting aspect of the results in figure 7.5 and table 7.3, is that all beginning labels(labels beginning with *B-*) have lower scores than on the original test set. As can be seen in the confusion matrix of the results in figure 7.6, *O* is the most common labeling of all the *B*-tags, except for *B-premise*, where *I-premise* is the most frequent label assigned.

NorArg test set

The performance on the NorArg test set was, as in the previous experiment, not very impressive. However, some aspects were slightly better in this

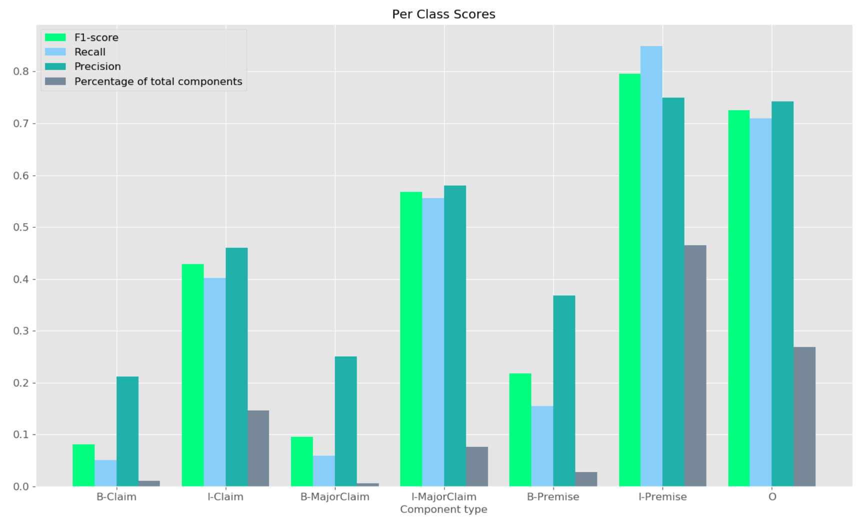


Figure 7.5: Recall, F1-score, precision, and percentage of total component amount per component class, from testing on the Norwegian version of the persuasive essays test set. This model was trained on the Norwegian version of the persuasive essays set for 25 epochs.

	Precision	**Recall**	**F1-score**	**Support**
I-Premise	0.38	0.78	0.51	5828
O	0.56	0.34	0.42	8001
I-Claim	0.33	0.11	0.17	2779
I-MajorClaim	0.12	0.02	0.04	701
B-Premise	0.37	0.13	0.19	371
B-Claim	0.20	0.01	0.02	219
B-MajorClaim	0.00	0.00	0.00	39
Micro average	0.42	0.42	0.42	17938
Macro average	0.28	0.20	0.19	17938
Weighted average	0.44	0.42	0.39	17938

Table 7.4: Results of testing LSTM-ER trained on the Norwegian training set on the Norwegian version of NorArg.

case than when we tested the English trained model on the translated version of the set. But, as in the previous case, it seems to be the case that the model simply predicts the most common component type (*I-Premise*), as is reflected in the per class scores in table 7.4 and figure 7.7, and the confusion matrix in figure 7.8.

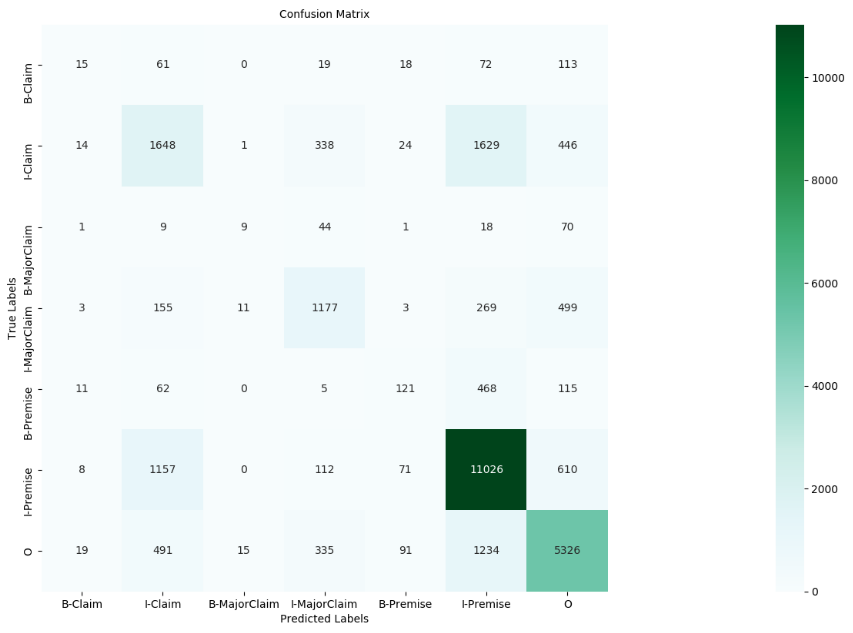


Figure 7.6: Confusion matrix of the predicted and true labels for argument mining components on the Norwegian version of the persuasive essays test set. The model used is trained on the Norwegian version of the training set for 25 epochs.

7.3 NCRF++

The first of our comparison systems is the NCRF++ toolkit, described in chapter 2. We follow the same procedure as with the LSTM-ER, and first train a model on the English training set, then train another model on the Norwegian dataset, before testing them on the corresponding test sets.

7.3.1 English test sets

As with the LSTM-ER, we begin our testing of NCRF++ by training our model on the original persuasive essay training set, and testing the model on the original persuasive essay test set, as well as on the English version of NorArg.

Persuasive essay test set

NCRF++ gives similar results on the Persuasive essay test set to what LSTM-ER did. The scores are slightly higher, but the most visible difference seems to be that the recall is generally higher for most classes, as can be seen in figure 7.9 and table 7.5 which shows F1-score, precision and recall per class for the results on the test set. Another clear difference is that the precision for the *Claim* tags has decreased, while recall has gone up slightly.

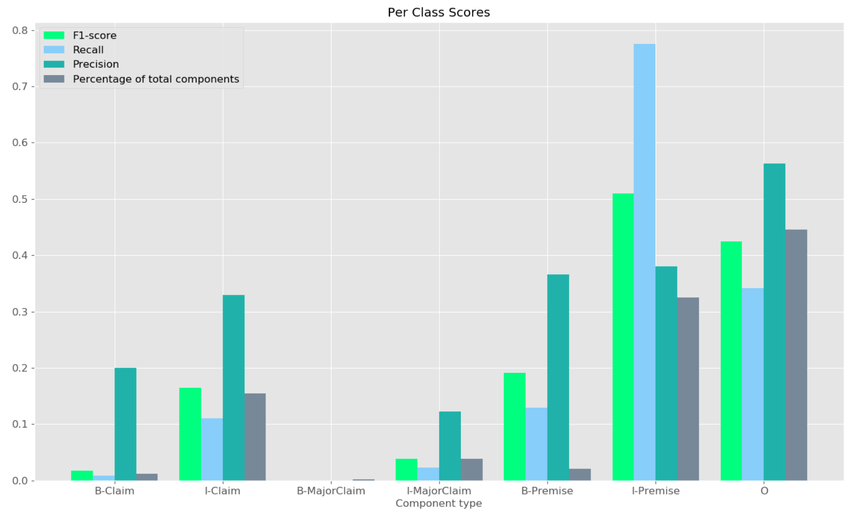


Figure 7.7: Recall, F1-score, precision, and percentage of total component amount per component class, from testing on the Norwegian version of NorArg. This model was trained on the Norwegian version of the persuasive essays set for 25 epochs.

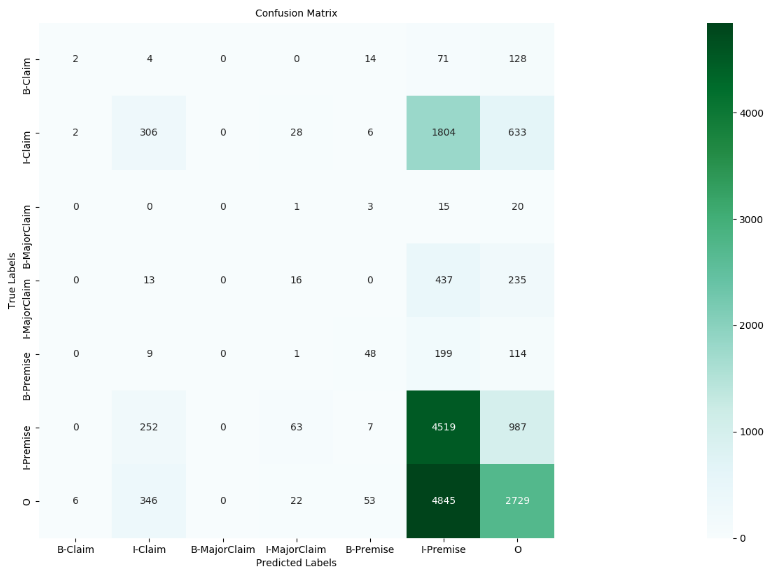


Figure 7.8: Confusion matrix of the predicted and true labels for argument mining components on the Norwegian version of NorArg. The model used is trained on the Norwegian version of the training set for 25 epochs.

Figure 7.10 shows a confusion matrix of results per class, and shows that only 972 *I-claim* tags have been correctly classified, less than half of what the LSTM-ER managed. The main component *I-claim* is confused with is still the *I-premise*, with 2387 *I-claim* components being misclassified as *I-premise*.

	Precision	Recall	F1-score	Support
B-Claim	0.58	0.41	0.48	304
B-MajorClaim	0.63	0.60	0.62	153
B-Premise	0.76	0.78	0.77	809
I-Claim	0.60	0.38	0.46	4493
I-MajorClaim	0.55	0.69	0.61	2101
I-Premise	0.83	0.91	0.87	14487
O	0.83	0.82	0.83	7206
Micro average	0.78	0.78	0.78	29553
Macro average	0.72	0.70	0.71	29553
Weighted average	0.77	0.78	0.77	29553

Table 7.5: Results of testing NCRF++ trained on the English training set on the English persuasive essay test set.

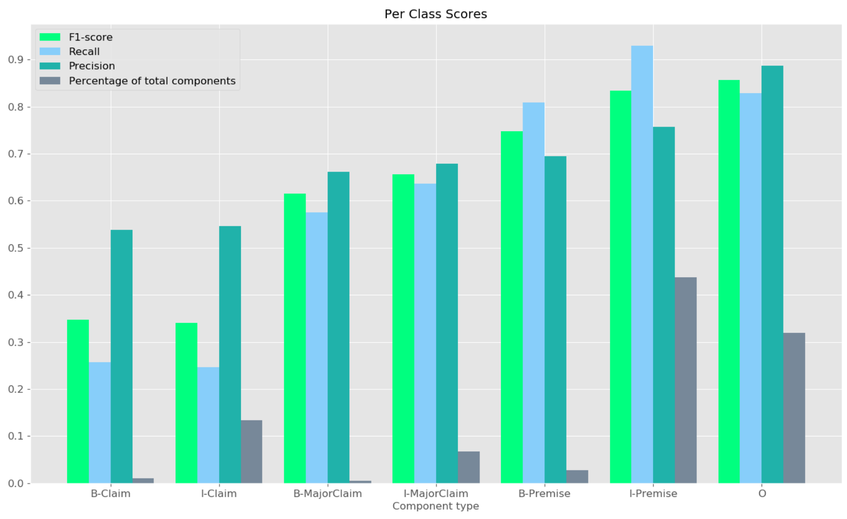


Figure 7.9: Recall, F1-score, precision, and percentage of total component amount per component class, from testing on the English version of the persuasive essays test set. This model was trained using the NCRF++ toolkit, on the English version of the persuasive essays set.

NorArg test set

The results from testing NCRF++ on the English version of NorArg are again quite similar to the results from LSTM-ER on the same set. Figure 7.11 and table 7.6 shows F1-score, recall, precision and percentage of total components per class. The differences from the LSTM-ER results are more or less the same as the differences on the persuasive essay test set in the previous section, as can be seen in the confusion matrix in figure 7.12.

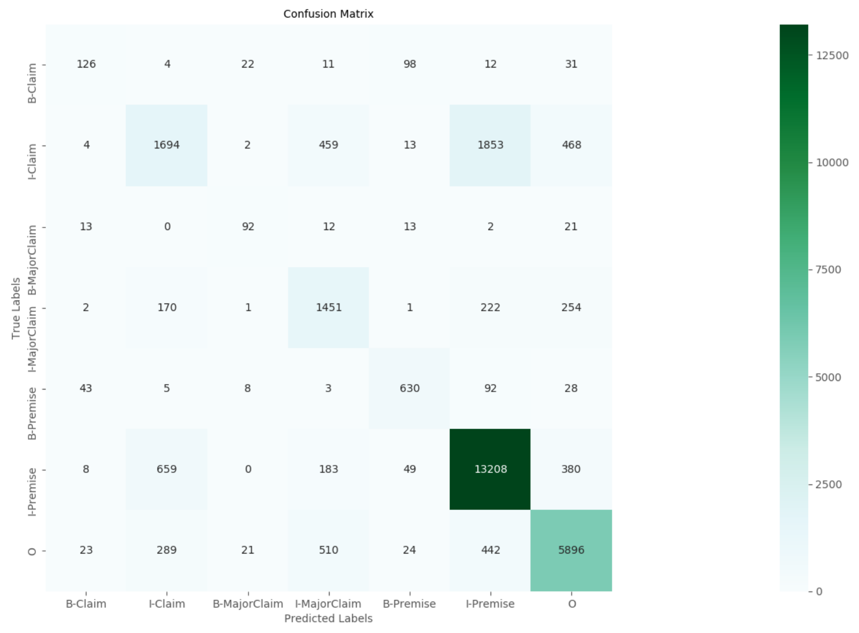


Figure 7.10: Confusion matrix of the predicted and true labels for argument mining components on the English version of the persuasive essays test set. The model was trained using the NCRF++ toolkit on the English version of the training set.

	Precision	**Recall**	**F1-score**	**Support**
B-Claim	0.23	0.02	0.03	187
B-MajorClaim	0.00	0.00	0.00	28
B-Premise	0.22	0.46	0.30	270
I-Claim	0.38	0.03	0.05	2557
I-MajorClaim	0.00	0.00	0.00	527
I-Premise	0.34	0.89	0.49	4245
O	0.58	0.20	0.29	6107
Micro average	0.37	0.37	0.37	13921
Macro average	0.34	0.32	0.27	13921
Weighted average	0.43	0.37	0.29	13921

Table 7.6: Results of testing NCRF++ trained on the English training set on the English version of NorArg.

The model does worse than LSTM-ER on the *claim* tags, most of the time assigning them the *premise* label. Although this model in general assigns the *premise* label to every word, similar to what the LSTM-ER did.

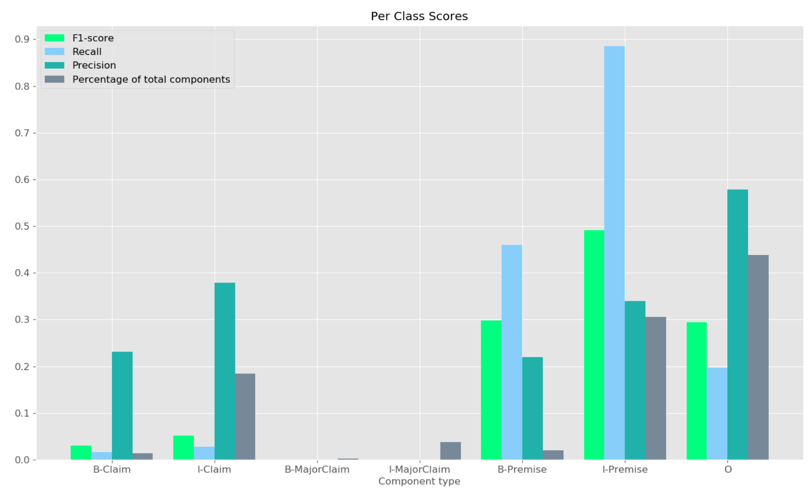


Figure 7.11: Recall, F1-score, precision, and percentage of total component amount per component class, from testing on the English version of NorArg. This model was trained using the NCRF++ toolkit, on the English version of the persuasive essays training set.

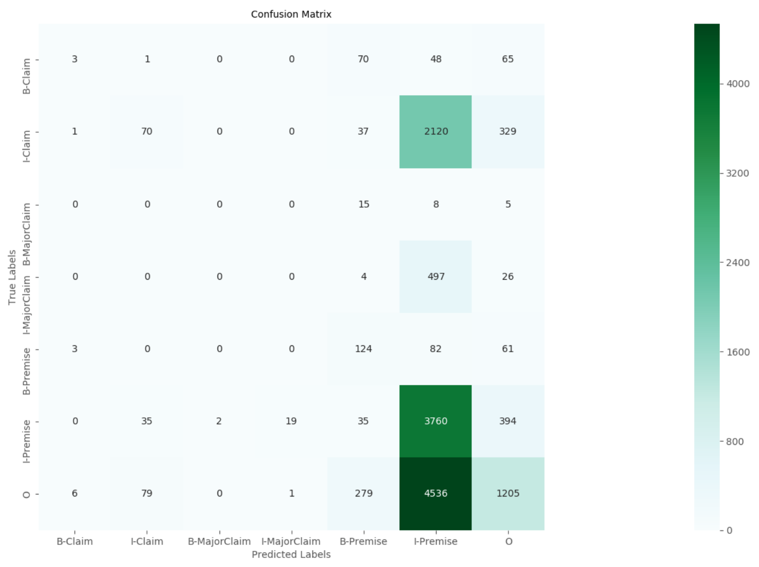


Figure 7.12: Confusion matrix of the predicted and true labels for argument mining components from testing on the English version of NorArg. The model was trained using the NCRF++ toolkit, on the English version of the training set.

7.3.2 Norwegian test sets

The second part of our experiment using NCRF++ consists of training a NCRF++-model on the Norwegian version of the persuasive essay training set, and testing it on the Norwegian version of the persuasive essay test set,

	Precision	Recall	F1-score	Support
B-Claim	0.11	0.07	0.09	298
B-MajorClaim	0.16	0.09	0.11	152
B-Premise	0.34	0.20	0.25	782
I-Claim	0.29	0.49	0.36	4100
I-MajorClaim	0.48	0.39	0.43	2117
I-Premise	0.74	0.74	0.74	12984
O	0.74	0.56	0.63	7511
Micro average	0.60	0.60	0.60	27944
Macro average	0.32	0.28	0.29	27944
Weighted average	0.63	0.60	0.61	27944

Table 7.7: Results of testing NCRF++ trained on the Norwegian training set on the Norwegian version of the persuasive essay test set.

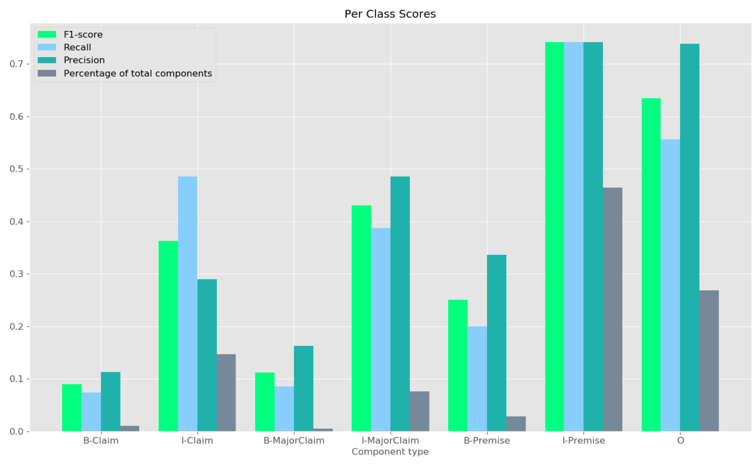


Figure 7.13: Recall, F1-score, precision, and percentage of total component amount per component class, from testing on the Norwegian version of the persuasive essays test set. This model was trained using the NCRF++ toolkit, on the Norwegian version of the training set.

as well as on the original NorArg test set.

Persuasive essays

As with the LSTM-ER, the NCRF++ does worse on the translated version of the persuasive essay test set than on the original. Figure 7.13 shows the per class scores, and figure 7.14 shows the confusion matrix containing predicted and true labels. The confusion matrix shows that this model makes similar mistakes as the other models, confusing *Claims* and *Premises*, and generally overestimating the number of *Premises* in the text.

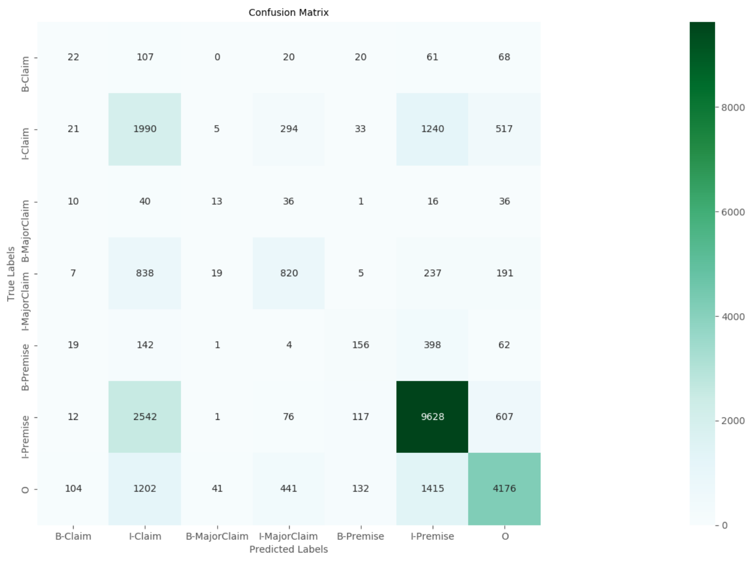


Figure 7.14: Confusion matrix of the predicted and true labels for argument mining components on the Norwegian version of the persuasive essays test set. The model was trained using the NCRF++ toolkit, on the Norwegian version of the training set.

	Precision	**Recall**	**F1-score**	**Support**
B-Claim	0.04	0.00	0.01	212
B-MajorClaim	0.00	0.00	0.00	36
B-Premise	0.23	0.25	0.24	333
I-Claim	0.28	0.04	0.08	2629
I-MajorClaim	0.00	0.00	0.00	606
I-Premise	0.36	0.89	0.52	5130
O	0.41	0.13	0.19	6588
Micro average	0.36	0.36	0.36	15534
Macro average	0.15	0.15	0.12	15534
Weighted average	0.35	0.36	0.27	15534

Table 7.8: Results of testing NCRF++ trained on the Norwegian training set on the NorArg test set.

NorArg

Again, as with the LSTM-ER experiments, the results from testing NCRF++ on the Norwegian version of NorArg are not very promising. The different scores, shown in figure 7.15 and table 7.8.

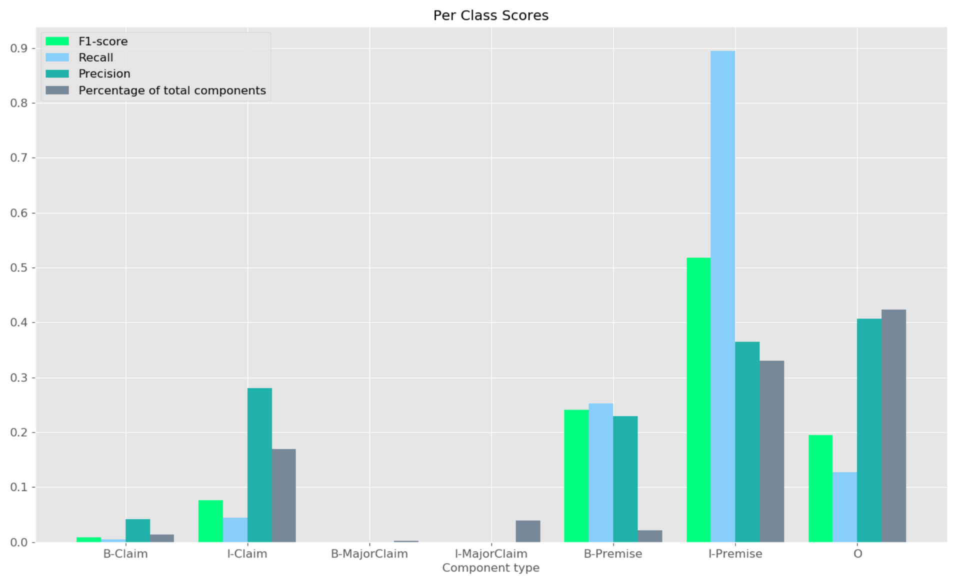


Figure 7.15: Recall, F1-score, precision, and percentage of total component amount per component class, from testing on the Norwegian version of NorArg. This model was trained using the NCRF++ on the Norwegian version of the training set.

7.4 Multilingual BERT

The final part of our experiments were conducted using our second comparison system, Multilingual BERT. The tests were done in the same as the previous two experiments, but because this is a multilingual model, we also tested it cross-lingually. Meaning that we tested the model trained on the English training set on the Norwegian test sets, and vice versa.

7.4.1 English test sets

As with the previous two models, we begin by testing on the English test sets. What is different from the previous models however, is that we use both the model trained on the English training set and the model trained on the Norwegian training set.

Persuasive essays

Multilingual BERT did the best overall of all the models tested on this set. But as can be seen in figure 7.17 and table 7.9, it struggles most with some of the same tags as the other models, namely *B-claim* and *I-claim*. As is the case in the previous experiments, the *Claims* are typically mistaken for *Premises* or *MajorClaims*, as can be seen in the confusion matrix in figure 7.18. As mentioned earlier, because of its capacity for understanding different

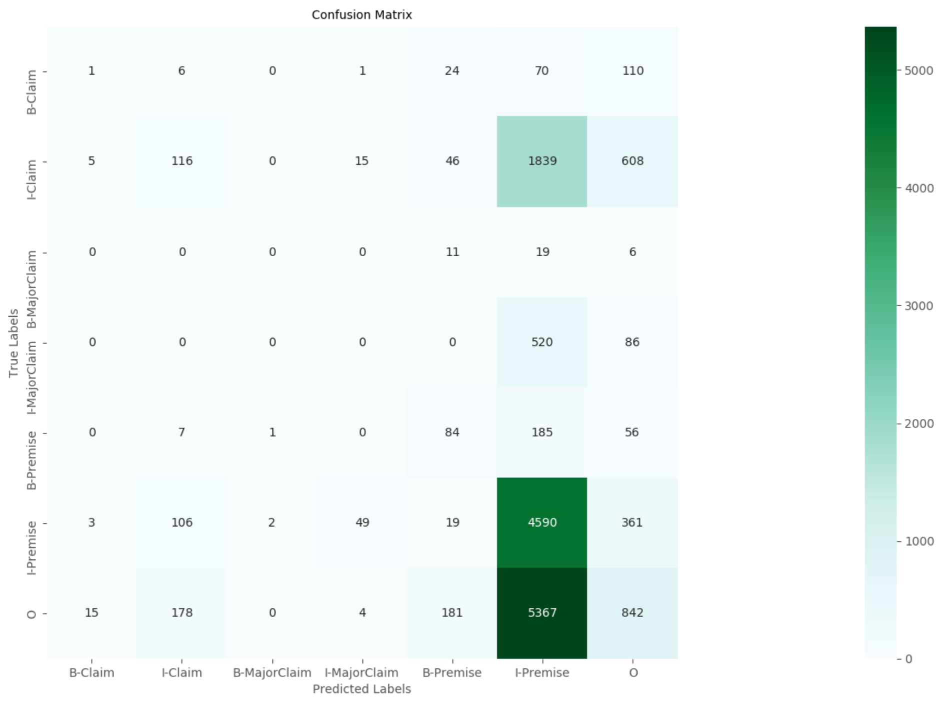


Figure 7.16: Confusion matrix of the predicted and true labels for argument mining components on the Norwegian version of NorArg. The model was trained with the NCRF++ toolkit, on the Norwegian version of the training set.

	Precision	**Recall**	**F1-score**	**Support**
B-Claim	0.55	0.55	0.55	305
B-MajorClaim	0.72	0.78	0.75	153
B-Premise	0.78	0.82	0.80	817
I-Claim	0.57	0.62	0.59	4511
I-MajorClaim	0.71	0.75	0.73	2101
I-Premise	0.88	0.88	0.88	14649
O	0.91	0.85	0.88	8107
Micro average	0.82	0.82	0.82	30643
Macro average	0.73	0.75	0.74	30643
Weighted average	0.82	0.82	0.82	30643

Table 7.9: Results of testing BERT trained on the English training set on the English version of the persuasive essay test set.

languages, we tested the Multilingual BERT trained on the Norwegian training set on the English sets. Judging by the results in figure 7.19 table 7.10 and the confusion matrix in figure 7.20 the results are worse than using the English version of the training set, but comparable to the previous

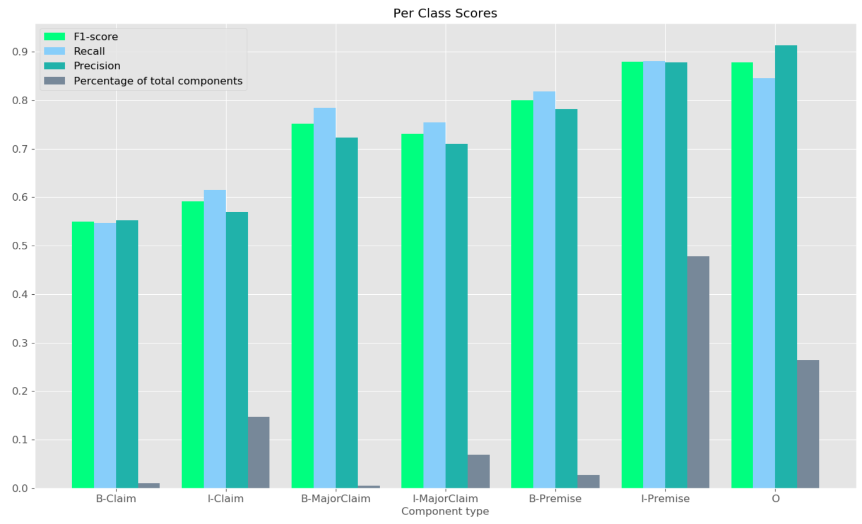


Figure 7.17: Recall, F1-score, precision, and percentage of total component amount per component class, from testing BERT trained on the English training set on the English version of the persuasive essays test set.

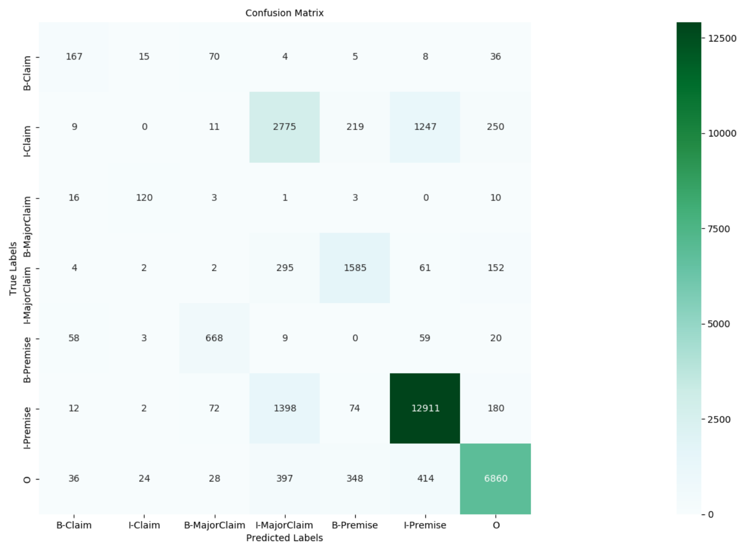


Figure 7.18: Confusion matrix of the predicted and true labels for argument mining components, from testing BERT trained on the English training set on the English version of the persuasive essays test set.

models' results on the translated version of the persuasive essay test set.

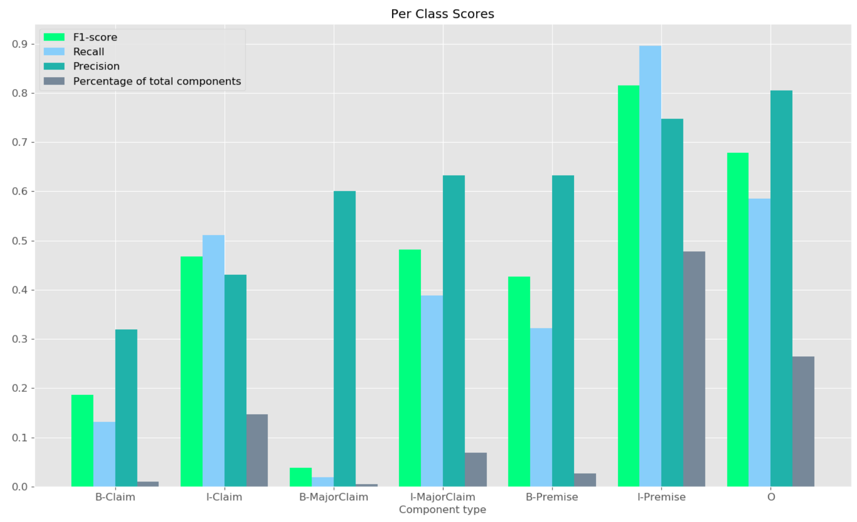
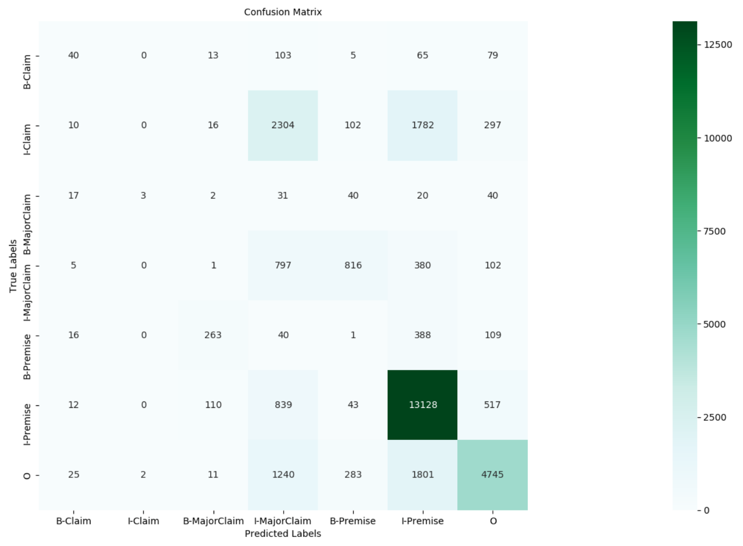


Figure 7.19: Recall, F1-score, precision, and percentage of total component amount per component class, from testing BERT trained on the Norwegian training set on the English version of the persuasive essays test set.



	B-Claim	I-Claim	B-MajorClaim	I-MajorClaim	B-Premise	I-Premise	O
B-Claim	40	0	13	103	5	65	79
I-Claim	10	0	16	2304	102	1782	297
B-MajorClaim	17	3	2	31	40	20	40
I-MajorClaim	5	0	1	797	816	380	102
B-Premise	16	0	263	40	1	388	109
I-Premise	12	0	110	839	43	13128	517
O	25	2	11	1240	283	1801	4745

Figure 7.20: Confusion matrix of the predicted and true labels for argument mining components, from testing BERT trained on the Norwegian training set on the English version of the persuasive essays test set.

NorArg

The next test is on the translated version of NorArg. As with previous tests on this set, the results are not very convincing, shown in figure 7.21, table 7.11 and in the confusion matrix in figure 7.22

What is interesting when testing the Multilingual BERT trained on the

	Precision	Recall	F1-score	Support
B-Claim	0.32	0.13	0.19	305
B-MajorClaim	0.60	0.02	0.04	153
B-Premise	0.63	0.32	0.43	817
I-Claim	0.43	0.51	0.47	4511
I-MajorClaim	0.63	0.39	0.48	2101
I-Premise	0.75	0.90	0.82	14649
O	0.81	0.59	0.68	8107
Micro average	0.70	0.70	0.70	30643
Macro average	0.60	0.41	0.44	30643
Weighted average	0.70	0.70	0.68	30643

Table 7.10: Results of testing BERT trained on the Norwegian training set on the English version of the persuasive essay test set.

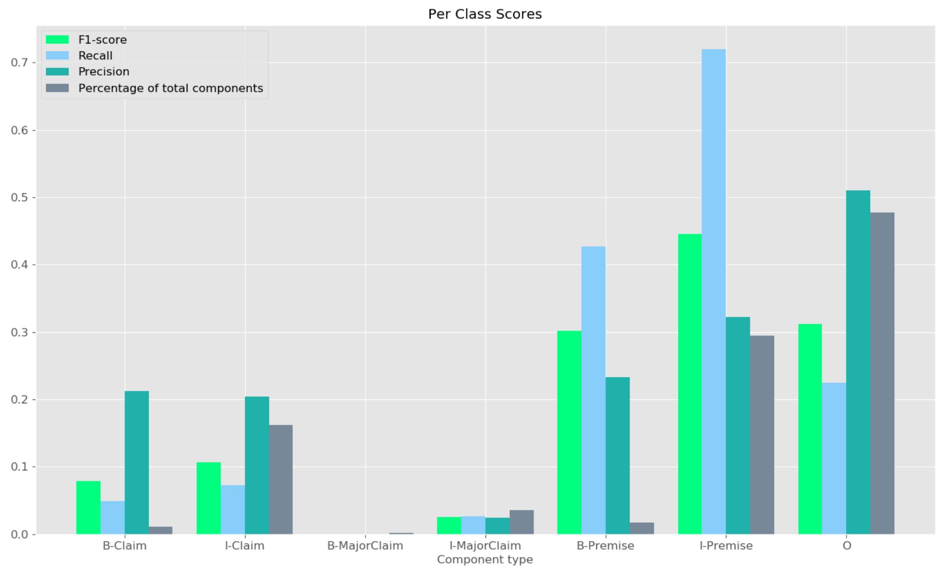


Figure 7.21: Recall, F1-score, precision, and percentage of total component amount per component class, from testing BERT trained on the English training set on the English version of the NorArg test set.

Norwegian training set and tested on the English version of NorArg, is that it does slightly better than the model trained on the English training set, perhaps indicating that it has picked up some structures from the Norwegian training set that were absent in the English training set. Results from this test are shown in figure 7.23, table 7.12 and the confusion matrix in figure 7.24.

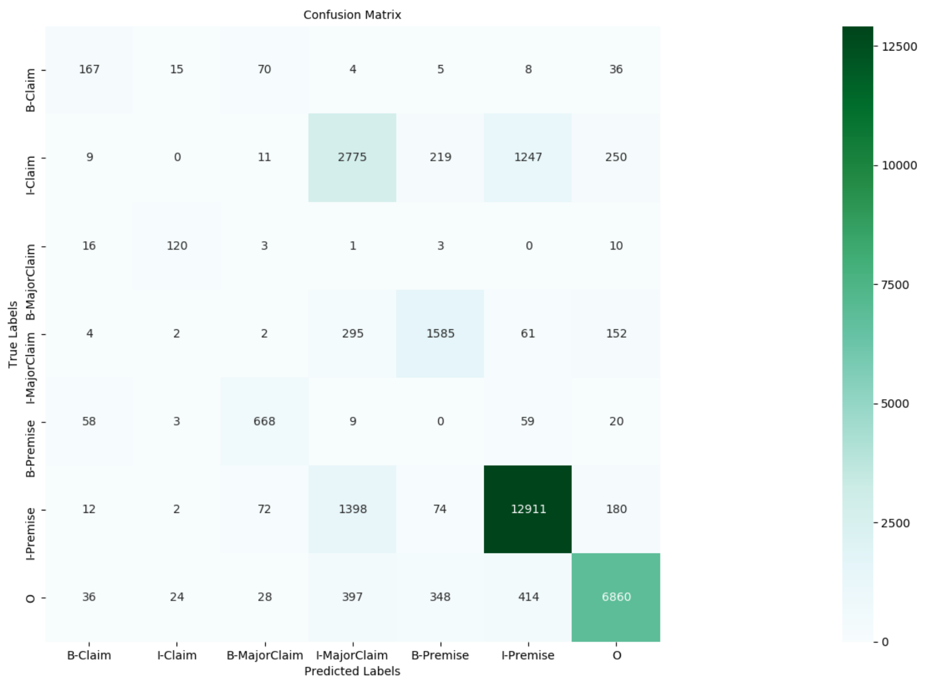


Figure 7.22: Confusion matrix of the predicted and true labels for argument mining components, from testing BERT trained on the English training set on the English version of the NorArg test set.

	Precision	**Recall**	**F1-score**	**Support**
B-Claim	0.21	0.05	0.08	205
B-MajorClaim	0.00	0.00	0.00	33
B-Premise	0.23	0.43	0.30	321
I-Claim	0.20	0.07	0.11	2933
I-MajorClaim	0.02	0.03	0.03	639
I-Premise	0.32	0.72	0.45	5343
O	0.51	0.22	0.31	8638
Micro average	0.34	0.34	0.34	18112
Macro average	0.22	0.22	0.18	18112
Weighted average	0.38	0.34	0.30	18112

Table 7.11: Results of testing BERT trained on the English training set on the English version of NorArg.

7.4.2 Norwegian test sets

The last section of our Multilingual BERT experiments consists of, as with the two previous models, testing on the Norwegian versions of our test sets; the translated version of the persuasive essay test set, and the NorArg test set.

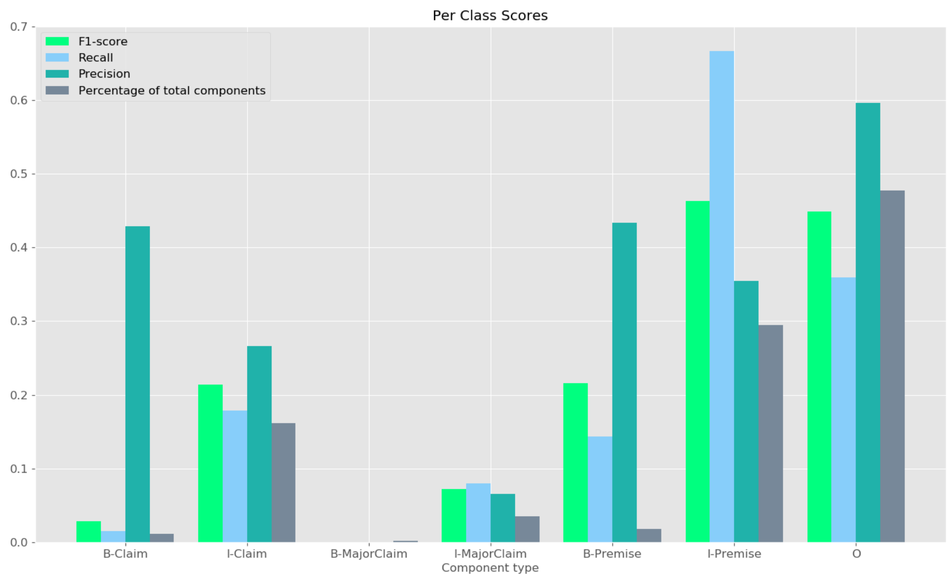


Figure 7.23: Recall, F1-score, precision, and percentage of total component amount per component class, from testing BERT trained on the Norwegian training set on the English version of the NorArg test set.

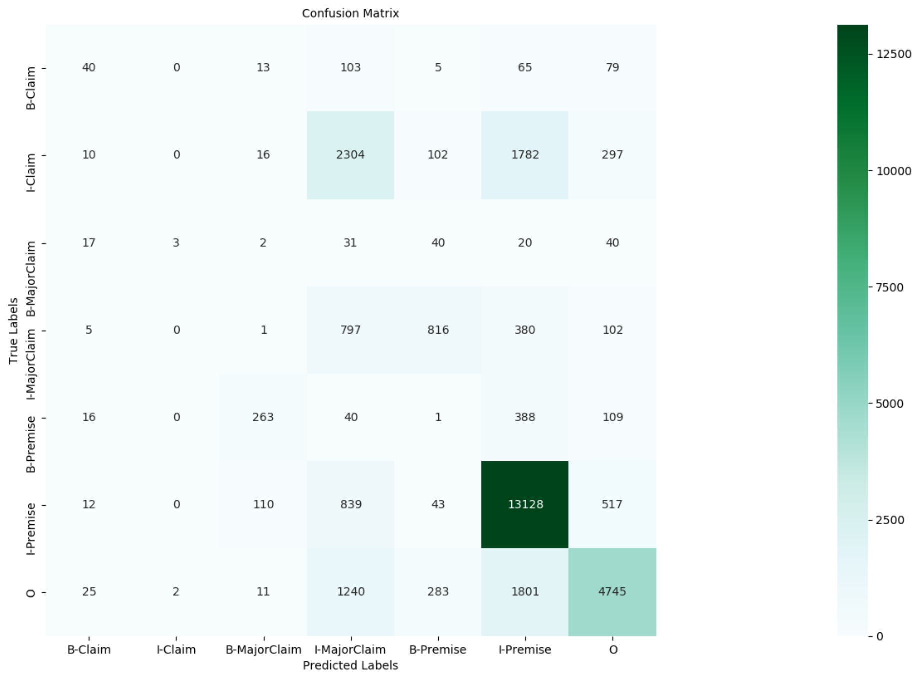
	Precision	Recall	F1-score	Support
B-Claim	0.43	0.01	0.03	205
B-MajorClaim	0.00	0.00	0.00	33
B-Premise	0.43	0.14	0.22	321
I-Claim	0.27	0.18	0.21	2933
I-MajorClaim	0.07	0.08	0.07	639
I-Premise	0.35	0.67	0.46	5343
O	0.60	0.36	0.45	8638
Micro average	0.40	0.40	0.40	18112
Macro average	0.31	0.21	0.21	18112
Weighted average	0.45	0.40	0.39	18112

Table 7.12: Results of testing BERT trained on the Norwegian training set on the English version of NorArg

Persuasive essays

Multilingual BERT trained on the Norwegian data set is the model that performed the best on the Norwegian version of the Persuasive Essays test set, as can be seen in table 7.20. The specific scores for the test are shown in figure 7.25 and table 7.13. The confusion matrix in 7.26 gives an overview of the distribution of predictions from the experiment.

The results from Multilingual BERT trained on the English training set and tested on this set can be seen in figure 7.27 and table 7.14. Again they



	B-Claim	I-Claim	B-MajorClaim	I-MajorClaim	B-Premise	I-Premise	O
B-Claim	40	0	13	103	5	65	79
I-Claim	10	0	16	2304	102	1782	297
B-MajorClaim	17	3	2	31	40	20	40
I-MajorClaim	5	0	1	797	816	380	102
B-Premise	16	0	263	40	1	388	109
I-Premise	12	0	110	839	43	13128	517
O	25	2	11	1240	283	1801	4745

Figure 7.24: Confusion matrix of the predicted and true labels for argument mining components, from testing BERT trained on the Norwegian training set on the English version of the NorArg test set.

	Precision	**Recall**	**F1-score**	**Support**
B-Claim	0.23	0.03	0.06	299
B-MajorClaim	0.00	0.00	0.00	153
B-Premise	0.35	0.18	0.24	788
I-Claim	0.51	0.54	0.52	4104
I-MajorClaim	0.64	0.62	0.63	2136
I-Premise	0.77	0.88	0.82	13094
O	0.82	0.71	0.76	8233
Micro average	0.73	0.73	0.73	28807
Macro average	0.48	0.42	0.43	28807
Weighted average	0.72	0.73	0.72	28807

Table 7.13: Results of testing BERT trained on the Norwegian training set on the Norwegian version of the persuasive essay test set.

are better than may be expected, and show off the cross-lingual capabilities of Multilingual BERT.

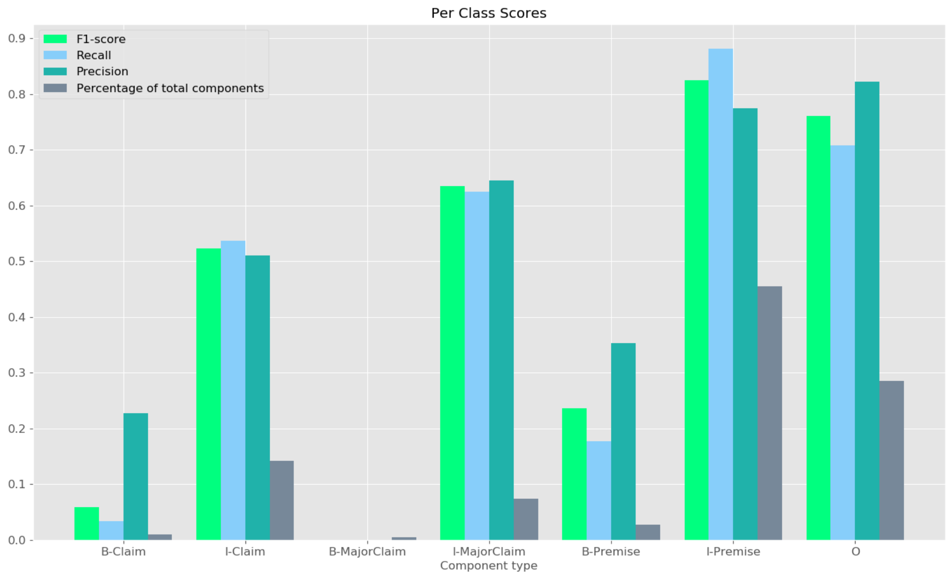


Figure 7.25: Recall, F1-score, precision, and percentage of total component amount per component class, from testing BERT trained on the Norwegian training set on the Norwegian version of the persuasive essays test set.

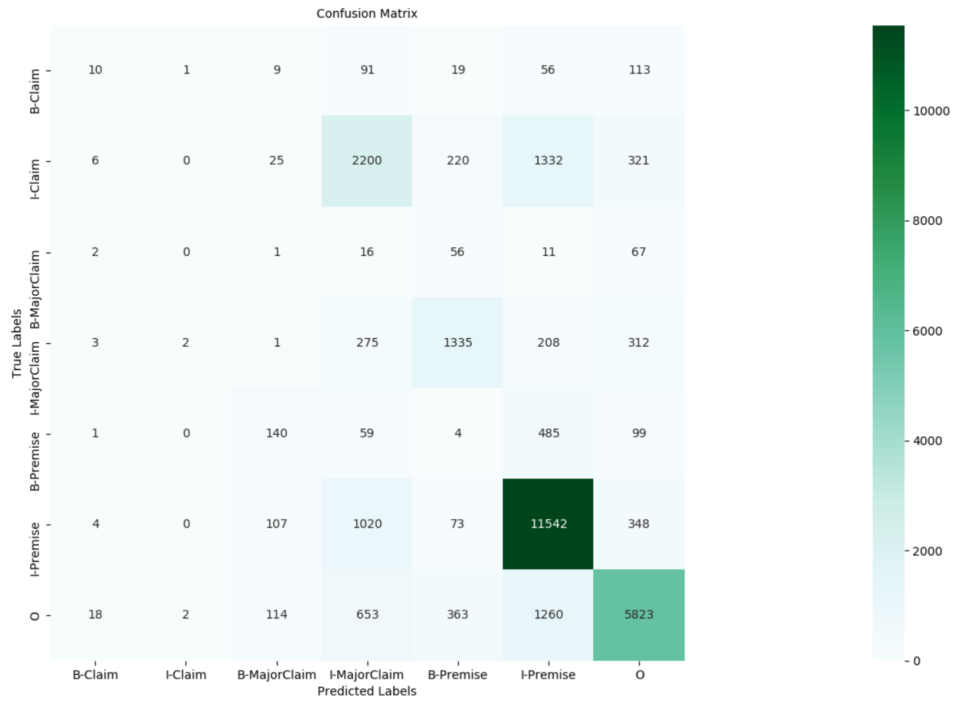
	Precision	**Recall**	**F1-score**	**Support**
B-Claim	0.13	0.07	0.09	299
B-MajorClaim	0.22	0.08	0.12	153
B-Premise	0.22	0.29	0.25	788
I-Claim	0.36	0.41	0.39	4104
I-MajorClaim	0.70	0.23	0.34	2136
I-Premise	0.64	0.86	0.73	13094
O	0.77	0.44	0.56	8233
Micro average	0.60	0.60	0.60	28807
Macro average	0.43	0.34	0.35	28807
Weighted average	0.62	0.60	0.58	28807

Table 7.14: Results of testing BERT trained on the English training set on the Norwegian version of the persuasive essay test set.

NorArg

Finally, we tested the various Multilingual BERT models on the Norwegian version of our own test set, NorArg. The results from testing the Multilingual BERT trained on the Norwegian training set can be seen in figure 7.29, table 7.15 and the confusion matrix in figure 7.30.

While the Norwegian version of BERT did comparably well on this corpus (although not very well at all), the model that did best in regards to complete component identification, is somewhat surprisingly, the BERT trained on the English training set, as shown in table 7.19. There can be



True \ Predicted	B-Claim	I-Claim	B-MajorClaim	I-MajorClaim	B-Premise	I-Premise	O
B-Claim	10	1	9	91	19	56	113
I-Claim	6	0	25	2200	220	1332	321
B-MajorClaim	2	0	1	16	56	11	67
I-MajorClaim	3	2	1	275	1335	208	312
B-Premise	1	0	140	59	4	485	99
I-Premise	4	0	107	1020	73	11542	348
O	18	2	114	653	363	1260	5823

Figure 7.26: Confusion matrix of the predicted and true labels for argument mining components, from testing BERT trained on the Norwegian training set, tested on the Norwegian version of the persuasive essays test set.

many reasons for this, perhaps the translation and annotation projection removes some key structures in the translated Norwegian training set, that are preserved in the English version, and enables to Multilingual BERT trained on the English set to perform better. More details about the results from the English trained version are shown in figure 7.31, table 7.16, and the confusion matrix in figure 7.22

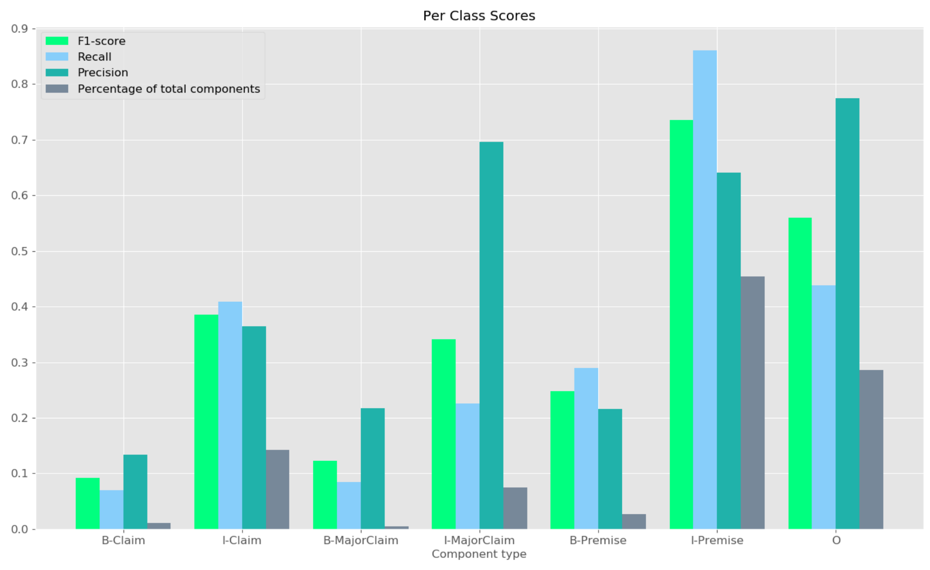


Figure 7.27: Recall, F1-score, precision, and percentage of total component amount per component class, from testing BERT trained on the English training set on the Norwegian version of the persuasive essays test set.

	Precision	**Recall**	**F1-score**	**Support**
B-Claim	0.67	0.01	0.02	220
B-MajorClaim	0.00	0.00	0.00	39
B-Premise	0.56	0.20	0.29	380
I-Claim	0.21	0.17	0.19	2795
I-MajorClaim	0.11	0.16	0.13	701
I-Premise	0.40	0.61	0.48	5961
O	0.57	0.41	0.48	8800
Micro average	0.42	0.42	0.42	18896
Macro average	0.36	0.22	0.23	18896
Weighted average	0.45	0.42	0.41	18896

Table 7.15: Results of testing BERT trained on the Norwegian training set on the Norwegian version of NorArg

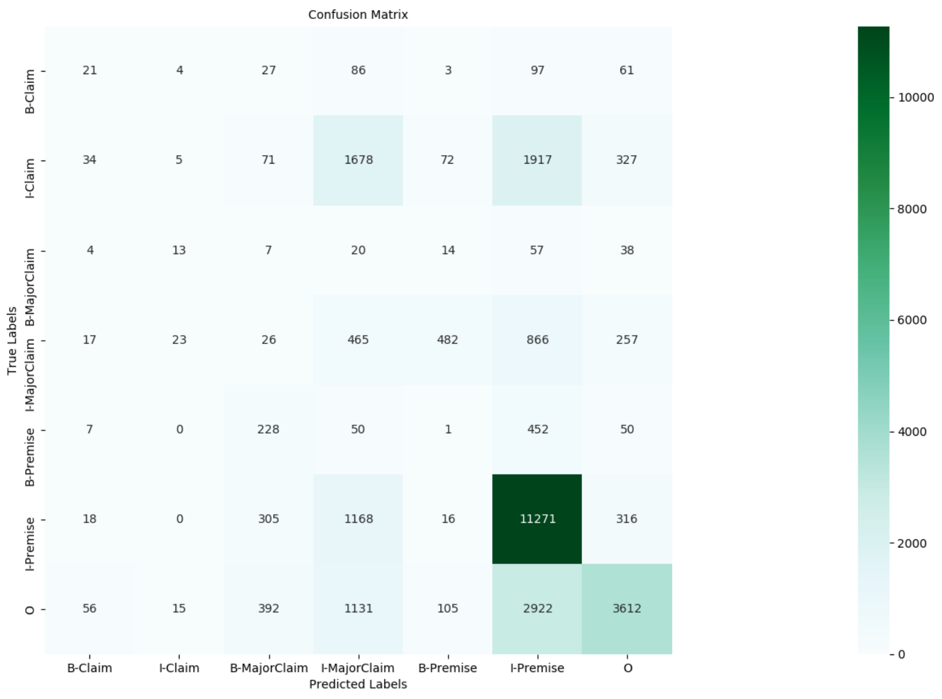


Figure 7.28: Confusion matrix of the predicted and true labels for argument mining components, from testing BERT trained on the English training seton the Norwegian version of the persuasive essays test set.

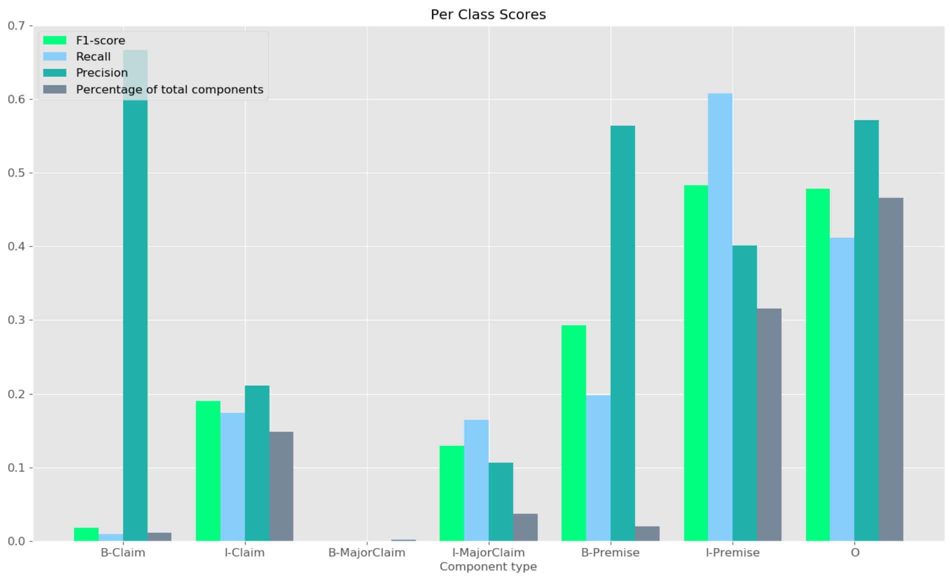
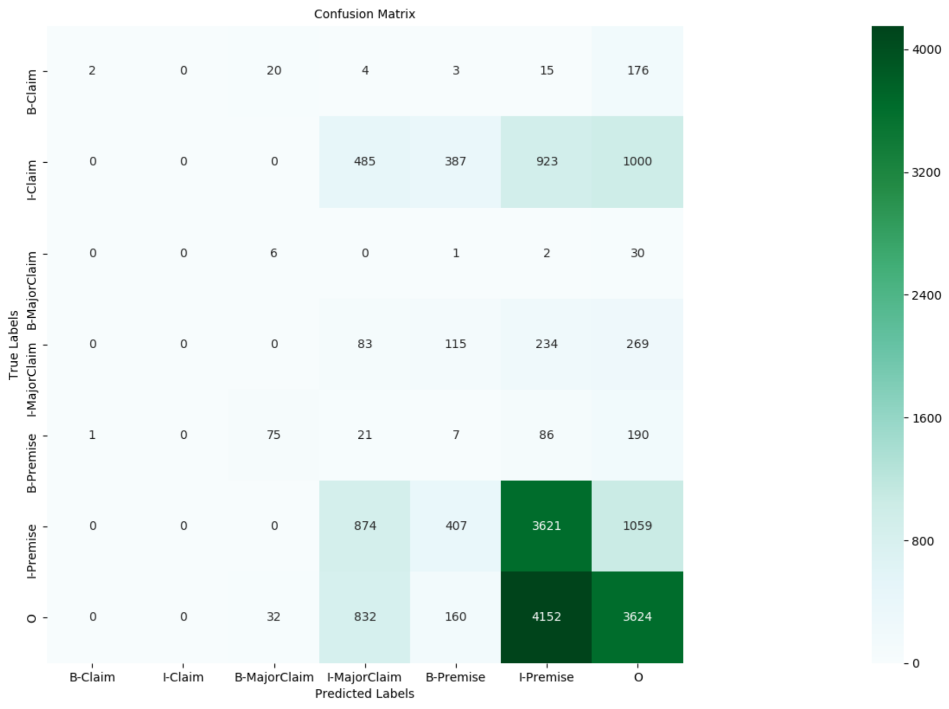


Figure 7.29: Recall, F1-score, precision, and percentage of total component amount per component class, from testing BERT trained on the Norwegian training set on the Norwegian version of the persuasive essays test set.



	B-Claim	I-Claim	B-MajorClaim	I-MajorClaim	B-Premise	I-Premise	O
B-Claim	2	0	20	4	3	15	176
I-Claim	0	0	0	485	387	923	1000
B-MajorClaim	0	0	6	0	1	2	30
I-MajorClaim	0	0	0	83	115	234	269
B-Premise	1	0	75	21	7	86	190
I-Premise	0	0	0	874	407	3621	1059
O	0	0	32	832	160	4152	3624

Figure 7.30: Confusion matrix of the predicted and true labels for argument mining components, from testing BERT trained on the Norwegian training set on the Norwegian version of the persuasive essays test set.

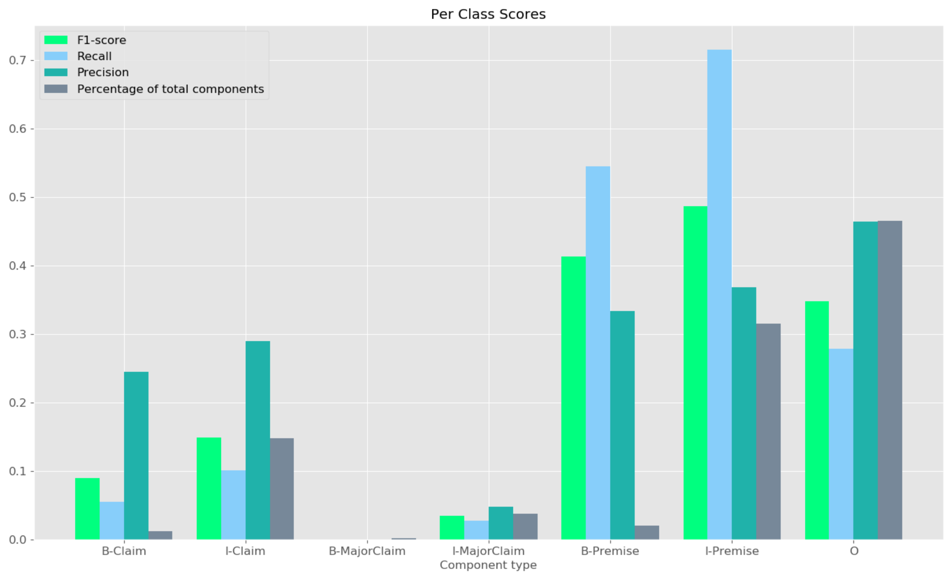


Figure 7.31: Recall, F1-score, precision, and percentage of total component amount per component class, from testing BERT trained on the English training set on the Norwegian version of the persuasive essays test set.

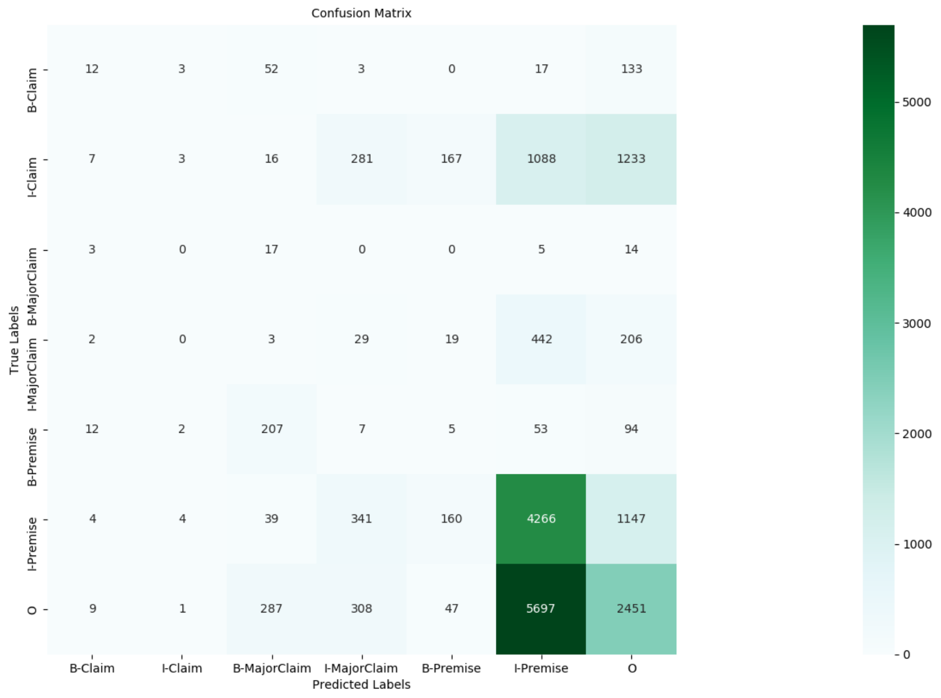


Figure 7.32: Confusion matrix of the predicted and true labels for argument mining components, from testing BERT trained on the English training set on the Norwegian version of the persuasive essays test set.

	Precision	**Recall**	**F1-score**	**Support**
B-Claim	0.24	0.05	0.09	220
B-MajorClaim	0.00	0.00	0.00	39
B-Premise	0.33	0.54	0.41	380
I-Claim	0.29	0.10	0.15	2795
I-MajorClaim	0.05	0.03	0.03	701
I-Premise	0.37	0.72	0.49	5961
O	0.46	0.28	0.35	8800
Micro average	0.38	0.38	0.38	18896
Macro average	0.25	0.25	0.22	18896
Weighted average	0.39	0.38	0.35	18896

Table 7.16: Results of testing BERT trained on the English training set on the Norwegian version of NorArg.

NorArg(ENG)	Acc.	C-F1	
Models		100%	50%
Multilingual BERT (ENG)	0.34	0.07	0.30
Multilingual BERT (NOR)	**0.40**	0.00	0.28
LSTM-ER (ENG)	0.32	0.06	0.30
NCRF++ (ENG)	0.37	**0.28**	**0.41**

Table 7.17: Component overlap scores from the various models tested on the English translated version of NorArg. (NOR) after a model name indicates that the model was trained on the Norwegian training set, (ENG) that it was trained on the English.

Persuasive Essays (ENG)	Acc.	C-F1	
Models		100%	50%
Multilingual BERT (ENG)	**0.82**	0.69	**0.77**
Multilingual BERT (NOR)	0.70	0.16	0.60
LSTM-ER (ENG)	0.74	0.56	0.65
NCRF++ (ENG)	0.77	**0.72**	**0.77**

Table 7.18: Component overlap scores from the various models tested on the English version of the Persuasive Essay test set. (NOR) after a model name indicates that the model was trained on the Norwegian training set, (ENG) that it was trained on the English.

NorArg (NOR)	Acc.	C-F1	
Models		100%	50%
Multilingual BERT (ENG)	0.38	**0.06**	**0.29**
Multilingual BERT (NOR)	0.42	0.002	0.26
LSTM-ER (ENG)	**0.43**	0.01	0.16
NCRF++ (ENG)	0.36	0.02	0.20

Table 7.19: Component overlap scores from the various models tested on the Norwegian version of the NorArg test set. (NOR) after a model name indicates that the model was trained on the Norwegian training set, (ENG) that it was trained on the English.

Persuasive Essays (NOR)	Acc.	C-F1	
Models		100%	50%
Multilingual BERT (ENG)	0.60	0.12	**0.54**
Multilingual BERT (NOR)	**0.73**	**0.17**	**0.54**
LSTM-ER (NOR)	0.69	0.09	0.37
NCRF++ (NOR)	0.60	0.14	0.41

Table 7.20: Component overlap scores from the various models tested on the Norwegian version of the Persuasive Essays test set. (NOR) after a model name indicates that the model was trained on the Norwegian training set, (ENG) that it was trained on the English.

Chapter 8

Conclusion

This thesis has deal with the task of Argument Mining in Norwegian text. We have created the first annotated data set for argument mining in Norwegian: NorArg. In addition, we provide guidelines that make it easier for future research efforts expand upon that dataset. We modeled our guidelines on the guidelines provided by Stab and Gurevych (2017). To assess the quality of our guidelines we provde inter-annotator agreement scores based on a quality test set of 10 documents. Our scores reflect the fact that identifying arguments is a complex task, and we hypothesize that especially the choice of domain is very influential in how easy it is to successfully detect arguments. How structured and organised a review is differs greatly from the structure in a persuasive essay, for instance, often depending on the style of the author. Details about various domain differences are discussed more in detail in chapter 4.

The next part of this thesis consists of an experimental analysis of two forms of cross-lingual learning techniques, which are evaluated on our annotated dataset NorArg. The first technique, annotation projection, is described in detail in chapters 5 and 6. In chapter 5 we describe the process of translating our corpora using Google Translate and using Fast-Align to project our annotations from the text in the original language to the translated text. This process also includes many potential inaccuracies, which are further analysed in chapter 5. The second cross-lingual learning technique consists of using multilingual transfer learning by using a Multilingual BERT model. The results from all our experiments are discussed in detail in chapter 7. Overall, we find that Multilingual BERT performs better than the other models. Even so, evaluation shows that none of the models perform very well on our new dataset. As discussed in chapter 7 there are probably many reasons for this, such as errors in translation and projection annotation when using the translated versions of the training and test sets and the fact that our training set is from a domain that is significantly different from our test set. In addition, the task of argument mining is a complex task that is challenging both to annotate manually and to automate.

8.1 Future Work

When creating our dataset we included information about the relations between components in the texts. The plan was to also create a system that would include these, but due to an increased complexity in modeling we decided to leave the relation analysis for future work. However, the relations are already annotated and are available in NorArg, making it easy to include them in potential future work.

As mentioned earlier, our thesis includes guidelines on how to annotate arguments in review texts, making it easier to continue annotating arguments in Norwegian reviews and expanding NorArg. The bottleneck in NLP tasks is often the amount of data available, so this is something that could improve the results of similar experiments greatly.

Lastly, because of the limited time and scope of the thesis, we did not perform a very thorough hyperparameter search for our various models, and could possibly have achieved better results if this had been done.

Bibliography

Agić, Ž., Johannsen, A., Plank, B., Martínez Alonso, H., Schluter, N., and Søgaard, A. (2016). Multilingual projection for parsing truly low-resource languages. *Transactions of the Association for Computational Linguistics*, 4:301–312.

Aker, A., Sliwa, A., Ma, Y., Lui, R., Borad, N., Ziyaei, S., and Ghobadi, M. (2017). What works and what does not: Classifier and feature analysis for argument mining. In *Proceedings of the 4th Workshop on Argument Mining*, pages 91–96.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Benjamin, M. (2019). Teach you backwards: An in-depth study of google translate for 108 languages.

Cabrio, E. and Villata, S. (2018). Five years of argument mining: a data-driven analysis. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 5427–5433. AAAI Press.

Chen, D. and Manning, C. D. (2014). A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Dyer, C., Chahuneau, V., and Smith, N. A. (2013). A simple, fast, and effective reparameterization of IBM model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, Georgia. Association for Computational Linguistics.

Eger, S., Daxenberger, J., and Gurevych, I. (2017). Neural end-to-end learning for computational argumentation mining. *arXiv preprint arXiv:1704.06104*.

Eger, S., Daxenberger, J., Stab, C., and Gurevych, I. (2018). Cross-lingual argumentation mining: Machine translation (and a bit of projection) is all you need! In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 831–844, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.

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

Freeman, J. B. (2011). *Argument Structure:: Representation and Theory*, volume 18. Springer Science & Business Media.

Goldberg, Y. and Hirst, G. (2017). *Neural Network Methods in Natural Language Processing*. Morgan & Claypool Publishers.

Groarke, L. (2017). Informal logic. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, spring 2017 edition.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Krippendorff, K. (2004). Measuring the reliability of qualitative text analysis data. *Quality and quantity*, 38:787–800.

Lafferty, J., McCallum, A., and Pereira, F. C. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Lippi, M. and Torroni, P. (2015). Argument mining: A machine learning perspective. In *International Workshop on Theorie and Applications of Formal Argumentation*, pages 163–176. Springer.

Lippi, M. and Torroni, P. (2016). Argumentation mining: State of the art and emerging trends. *ACM Transactions on Internet Technology (TOIT)*, 16(2):10.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.

Miwa, M. and Bansal, M. (2016). End-to-end relation extraction using lstms on sequences and tree structures. *CoRR*, abs/1601.00770.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Persing, I. and Ng, V. (2016). End-to-end argumentation mining in student essays. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1384–1394, San Diego, California. Association for Computational Linguistics.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. *CoRR*, abs/1802.05365.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training.

Schulz, C., Eger, S., Daxenberger, J., Kahse, T., and Gurevych, I. (2018). Multi-task learning for argumentation mining in low-resource settings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 35–41.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Stab, C. and Gurevych, I. (2014). Annotating argument components and relations in persuasive essays. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1501–1510.

Stab, C. and Gurevych, I. (2017). Parsing argumentation structures in persuasive essays. *Computational Linguistics*, 43(3):619–659.

Straka, M. and Straková, J. (2017). Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada. Association for Computational Linguistics.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *CoRR*, abs/1706.03762.

Velldal, E., Øvrelid, L., Bergem, E. A., Stadsnes, C., Touileb, S., and Jørgensen, F. (2017). Norec: The norwegian review corpus. *arXiv preprint arXiv:1710.05370*.

Walton, D. (2009). *Argumentation Theory: A Very Short Introduction*, pages 1–22. Springer US, Boston, MA.

Yang, J. and Zhang, Y. (2018). NCRF++: an open-source neural sequence labeling toolkit. *CoRR*, abs/1806.05626.

Zhang, Y., Gaddy, D., Barzilay, R., and Jaakkola, T. (2016). Ten pairs to tag – multilingual POS tagging via coarse mapping between embeddings. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1307–1317, San Diego, California. Association for Computational Linguistics.