ARGUING WITH BERT: ARGUMENTATION MINING USING CONTEXTUALISED EMBEDDING AND TRANSFORMERS

A DISSERTATION SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF MASTER OF SCIENCE
IN THE FACULTY OF SCIENCE AND ENGINEERING

2019

Gergely Dániel Németh
Student id: 10389223

Department of Computer Science

Contents

Abstract **7**

Declaration **8**

Copyright **9**

1 Introduction **10**

2 Background: Argumentation theory **12**

2.1 Argument structures 12

2.2 Argument Scheme Structures 14

2.2.1 Argument from Expert Opinion 14

2.2.2 Argument from Sign 15

2.2.3 Appeal to Popular Opinion 15

3 Background: Neural Networks in NLP **16**

3.1 Feedforward neural network 16

3.1.1 Dropout 18

3.2 Recurrent neural network 18

3.2.1 LSTM 18

3.3 Word representations in NLP 20

3.3.1 One-hot word representation 20

3.3.2 Word embedding 21

3.3.3 Contextualised word embedding 23

3.4 Transformers 25

3.4.1 Sequence-to-sequence models 25

3.4.2 Attention 27

3.5 BERT 27

3.5.1 Masked LM . . . 29
3.5.2 Next Sentence Prediction . . . 29
3.5.3 Pre-Training . . . 29
3.6 Transfer Learning . . . 29

4 Related Work: Argumentation Mining techniques 33
4.1 Argument Component Identification . . . 33
4.2 Argument Component Classification . . . 35
4.3 Argument Connection Detection . . . 36
4.4 Argumentative Structure Determination . . . 36
4.4.1 Topic Similarity . . . 37
4.5 Scheme Structure Selection . . . 38
4.6 Combined methods . . . 38
4.7 Discourse Indicators . . . 39
4.8 Feature Selection . . . 39
4.8.1 Local features . . . 40
4.8.2 Global features . . . 41

5 Dataset 43
5.1 ArguE Unified Corpus Format . . . 43
5.2 AIF-DB Corpora Collection . . . 44
5.2.1 AraucariaDB . . . 44
5.3 Student Essay Corpus . . . 46
5.4 ArguE Corpus . . . 47

6 Research methodology 48
6.1 Data collection and preprocessing . . . 48
6.2 Feature selection . . . 49
6.2.1 ArguE features . . . 51
6.2.2 RST features . . . 52
6.2.3 Sentiment scores . . . 53
6.2.4 Word and sentence embedding vectors . . . 53
6.3 Classification problems . . . 53
6.4 Classification models . . . 54
6.4.1 ArguE based model using token- and sentence-level features . 54
6.4.2 Feed-Forward model using only sentence-level features . . . 56

6.4.3 Transfer learning using BERT 56
6.5 Combining corpora using the unified parser 58
6.6 Description of the tool 58

7 Evaluation **62**
7.1 Relation detection classifiers 62
7.2 Feature performance 65
7.3 Proposition type classifier 67
7.4 Unified corpus 68
7.5 Tool performance example 69

8 Analysis **72**
8.1 Data balance 72
8.2 Supporting features 73
8.3 General Argumentation Mining models 73

9 Future work **75**

10 Conclusion **76**

Bibliography **78**

A Student Essay Annotation: Original **89**

B Student Essay Annotation: Prediction **95**

Word Count: 14566

List of Tables

4.1 Clause level argument component identification 34
4.2 Discourse indicators: keywords used by Lawrence and Reed [LR15] and by Milz [Mil17] 39

6.1 Data collected from the Unified Corpus Format 49
6.2 Features of the models 50

7.1 Distribution of the relation database in the Student Essay Corpus. . . . 63
7.2 Relation detection models 63
7.3 Support-Attack identification 64
7.4 Non-related, support, attack identification 65
7.5 Relation detection performance with different features (Sentence Level model and ArguE based model) 65
7.6 Distribution of the proposition type database 67
7.7 Premise, Claim and Conclusion (Major Claim) classification models . 67
7.8 Comparison of the Sudent Essays Corpus and the Araucaria Corpus . 68
7.9 Relation detection models using unified corpus 68

List of Figures

2.1 A basic argument 13
2.2 Argument structures 13

3.1 Basics of feedforward neural networks 17
3.2 Basics of recurrent neural networks 19
3.3 The inside of an LSTM neuron 19
3.4 One-hot representation of the word *baby* in the two different way . . . 21
3.5 CBOW model example with the *The sky is blue* sentence 22
3.6 Word2vec vectors visualised in a 5x60 grayscale image 24
3.7 Words similar to $queen - woman + man$ 24
3.8 Sequence to sequence model architecture 26
3.9 The Tranformer - model architecture 28
3.10 Transfer learning with VGG16 30

4.1 Argument tree structure of a persuasive essay 37
4.2 Architecture of an general Argumentation Mining parser 38

5.1 AraucariaDB corpus Argument Map 16 representations 45

6.1 Original figure of ArguE's classifier architecture [Mil17, page 61] . . 55
6.2 Sentence-level network architecture with unit size 57
6.3 Argumentor tool pipeline 59

7.1 Relation graph of the original and the predicted annotation 71

Abstract

In this thesis, I investigate the possibilities of using contextualised word embedding vectors and Transformers in Argumentation Mining. I propose a model that uses only sentence-level embedding vectors. Therefore, eliminating the token level features, this model uses only sentence-level features, therefore, can be trained as a feed-forward network for argument classification problems. The experiment is tested on the argument component classification and argument relation detection subtasks of Argumentation Mining, achieving better performance on relation detection than the current state-of-the-art. Moreover, I show that Transfer Learning using Transformers can be applied in Argumentation Mining problems with competitive performance. Using the novel models described in the thesis, I built a pipeline architecture tool to perform Argumentation Mining tasks on the general text. The output of this tool is an annotation file in a format that is available for the most used Argumentation Mining corpora.

Declaration

No portion of the work referred to in this dissertation has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

i. The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the "Copyright") and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii. Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made **only** in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

iii. The ownership of certain Copyright, patents, designs, trade marks and other intellectual property (the "Intellectual Property") and any reproductions of copyright works in the thesis, for example graphs and tables ("Reproductions"), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv. Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see `http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=24420`), in any relevant Thesis restriction declarations deposited in the University Library, The University Library's regulations (see `http://www.library.manchester.ac.uk/about/regulations/`) and in The University's policy on presentation of Theses

Chapter 1

Introduction

Argumentation is an irreplaceable part of everyday human communication. It is part of the written and the spoken language. Argumentation can be a monologue, a written paper of reasons to convince general readers, or it can be a dialogue between arguing parts. The goal of the argumentation is to convince the opponent(s) to accept the arguer's view on a certain questionable statement. The method of argumentation is to derive a conclusion from premises that are acceptable by everyone.

Argumentation Mining (AM) is part of the research field of natural language processing (NLP) that focuses on identifying and analysing the arguments in a general text. Argumentation Mining is in close connection with other essential parts of Artificial Intelligence research, such as information retrieval and validation [CT15], legal counselling [PM09], policy-making [SKPK15] and debating technologies [LBH$^+$14, RDP$^+$15].

The argumentation mining process can be separated into several subtasks. The goal of *argument component identification* is to separate argumentative parts of the natural text from non-argumentative parts [MBPR07, FKKK13, LBH$^+$14, GLPK14, SG14b, SG17]. The *argument component classification* aims to decide whether a proposition is a premise or a claim [PM09, RWB12, SG14a]. The *argument connection detection* investigates the relation between argument components, identify if one supports or attacks the other or not even related to it [WEK12, SG14a, Mil17, LR15].

The high-level argumentation mining subtasks operate with multiple argument components and try to give a global overview of the argumentation. The *argumentative structure determination* aims to build an argument tree from the components [LRM$^+$14, LR15] and *scheme structure selection* aims to identify commonly used argumentation patterns in the argument structures [Wal12, FH11, LR15].

One of the key challenges of argumentation mining is that building an argumentation corpus requires well-trained annotators. Unlike many fields of science driven by machine learning, such as computer vision, argumentation mining does not have thousands of publicly available data. Building an argumentation corpus requires not only human power but a selection of annotation protocol. Unfortunately, there is no clear, generally accepted annotation for argumentation, therefore the publicly available corpora use several different formats.

Building general argumentation mining corpora faces not only technical difficulties but natural ones as well. As argumentation is a key part of human communication, it has many different ways to take place, from persuasive essays [SG17] to (presidential) debates [VKD+19], to internet forum argumentative posts [AEAW16]. The thesis does not engage with building a new corpus but uses existing corpora.

Recently, a study has attempted to bring closer the available corpora by making a Unified Format and a parser for several existing corpora [Mil17]. This study set out to investigate the usefulness of this Unified Corpus Format and the generalisability of the available corpus, models built on them.

This research also examines the possible role of Transformers [VSP+17, DCLT18, HR18] in the context of Argumentation Mining. Transformers are recent development of text mining and it is proven to have an emerging role in many fields of natural language processing.

This study is unable to encompass every subtask of argumentation mining and it focuses on argument component classification and argument connection detection.

The dissertation has been organised in the following way. The next chapter of this study summarises the theoretical background of argumentation. Chapter 3 provides a short overview of neural networks, the embedding vectors and entirely embedding based transfer learning models. Chapter 4 gives a brief history and overview of the related works. Chapter 5 introduces the corpora used in this experiment. Chapter 6 is concerned with the methodology used for this study. Chapter 7 describes the results of the experiments and Chapter 8 analyses them. Finally, Chapter 9 points out unanswered questions for future work.

Chapter 2

Background: Argumentation theory

Argumentation is part of the human language and maybe as old as it is. However, the investigation of arguments, good arguments, the argumentation theory is started with the ancient Greek philosophers from the 6th century B.C. Aristotle's logic is the first known work which defines the theory of logical reasoning and argumentation [Smi00].

Missimer describes argumentation as a method to prove a point to an opponent [Mis95]. The point called a *claim*. The process of the argumentation is to support the claim through logical reasoning, giving supporting pieces of evidence and inference relations to convince that the claim is a legitimate *conclusion* of the provided reasons.

2.1 Argument structures

On micro-level, arguments can be represented in a directed graph, where nodes are the propositions and the edges are the relations between them [Edw08, Gov13].

There are two types of propositions: a *claim* is the conclusion of an argument and a *premise* provides a reason to accept the relevant claim. More complex argumentation can contain multiple premises and claims connected in a tree. Some representation distinguish a third type for the final *conclusion* of the argumentation (also known as *major claim* [SG17]). Due to the multiple different argumentation models and representations, many different types and names can occur. Figure 2.1 shows an example of a basic argument containing two propositions.

There are two basic types of relations between propositions: in a *support* relation a supporting premise gives a reason to believe in the related claim; in a *attack* relation the premise contains evidence against the claim. More complex models can introduce different support and attack types based on the method used in the argument.

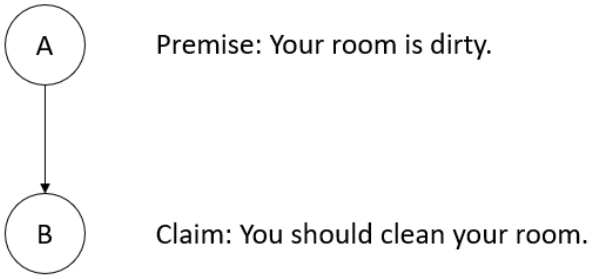


Figure 2.1: A basic argument

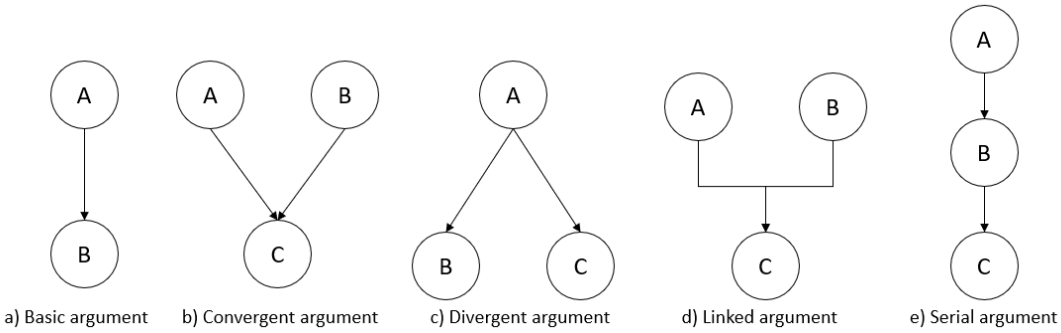


Figure 2.2: Argument structures

The simplest argument contains two connected nodes, however, there are more complex relations, represented in different ways in different argumentation theory models. Figure 2.2 shows basic argument structures. These structures are from different theories and still discussed among argumentation theorists [Mil17].

In the argument 2.2 (b), the convergent argument structure can be represented as two basic arguments ($A \rightarrow C$ and $B \rightarrow C$), simultaneously in the (c) argument, the complex divergent argument can be represented as two ($A \rightarrow B$ and $A \rightarrow C$). However, in the serial argument type (e), the proposition B has two different purposes if we try to represent it as a sum of basic arguments. In case of $A \rightarrow B$, it is a *claim*, meanwhile in $B \rightarrow C$, it is a *premise*.

This case is handled differently in many representations. For example, Monroe used multi-labelling (premise and claim at the same time) [Mon54], Govier used *main claim* for the final C node [Gov13], and *subclaim* for B and Cohen used only one claim for the final conclusion of the argumentation and handled everything else as a premise[Coh87]. In this paper, ArguE's representation is followed [Mil17], based on the model introduced by Freeman [Fre11]. In this model, every complex argument is

represented as a sum of basic 2-proposition argument, therefore the serial argument in the Figure 2.2 e) now has two arguments and the node *B* is a claim in the first one and a premise in the second one.

2.2 Argument Scheme Structures

Argumentation schemes are typical argument structures used in everyday life. These schemes contain patterns that can be helpful in argumentation theory. Identifying commonly used patterns can help when it comes to detecting the purpose of the arguments and the types of relations.

Walton et al. introduced 96 argumentation schemes aiming to cover most of the commonly used arguments of everyday communication [WRM08]. The following is an example scheme called *Argument from Expert Opinion*.

2.2.1 Argument from Expert Opinion

The Expert Opinion is a subtype of a more general scheme for the argument from position to know. The third-party expert has an opinion that the non-expert party of the dialogue wants to use as a piece of supporting evidence to the conclusion. This party refer to the source and usually the non-expert opponent respects the expert's opinion, however, Walton points out that the source may not be omniscient. The original description of the argument scheme from Walton et al:

Major Premise Source *E* is an expert in subject domain *S* containing proposition *A*.

Minor Premise *E* asserts that proposition *A* (in domain *S*) is true / false.

Conclusion *A* may plausibly be taken to be true / false.

A well-known example of the expert opinion is when in documentaries, the cast interview acknowledged scientist to talk about the topic: "(Interviewer): John Smith is a computer scientist in the University of Manchester. (John Smith): Understanding Argumentation Mining is necessary to build a General Artificial Intelligence. (Narrator): Argumentation Mining is important for Artifical Intelligence."

2.2.2 Argument from Sign

The Argument from Sign type of argument scheme is based on the Major Generalisation Premise. The generalisation says that if *A* statement is true, then usually, *B* is also true.

There could be another explanation, but in the absence of it, since the best explanation is the generalisation, the hypothesis is that it is true.

Minor Premise Given data represented as statement *A* is true in this situation.

Major (Generalisation) Premise Statement *B* is generally indicated as true when its sign, *A*, is true, in this kind of situation.

Conclusion Therefore, *B* is true in this situation.

An example of this argument is: "A metal is orange because it is melting. That metal is orange, therefore it is hot." The case might be that it is pained in orange but melting is the usual case.

2.2.3 Appeal to Popular Opinion

The idea of the Appeal to Popular Opinion scheme is that if a large majority of people accepts *A* as true, then we can assume that *A* is true.

A large majority accepts *A* as true

Therefore, there exists a presumption in favour of *A*.

A good example of this is the movie rating systems, like IMDb[1]. If a large majority of the viewers accept a movie as a good movie, then it is categorised as a good movie in the database of IMDb.

[1] `imdb.com`

Chapter 3

Background: Neural Networks in NLP

This chapter covers a summary of the basics of neural networks used in the experiment. For information about Transformers and BERT, see Section 3.4-3.6. The first sections covering the basics of neural networks are based on the *Deep learning in neural networks: An overview* by Schmidhuber [Sch15] and the course textbook of the class COMP61011 at the University of Manchester by Brown [1].

3.1 Feedforward neural network

The simplest feedforward neural network is the *perceptron*. The perceptron model uses the *discriminant function* to generate a decision boundary to solve classification problems with any number of features. The discriminant function is:

$$f(x) = \sum_{i=1}^{d} w_i x_i - b = w^T x - b$$

where i is the ith feature of the input, w is the weight of the model and d called dimension, the number of features. The perceptron decision rule is a binary classifier, for any given x, if $f(x) > 0$ the model predicts $\hat{y} = 1$ class, otherwise $\hat{y} = 0$ class. The aim of the algorithm called *learning algorithm* is to find the best w, b parameters for the function to fit the data samples.

While the perceptron model predicts a concrete class as an output based on the input features of the given sample, the *Logistic Regression* model predicts a probability

[1] `http://syllabus.cs.manchester.ac.uk/pgt/2018/COMP61011/`

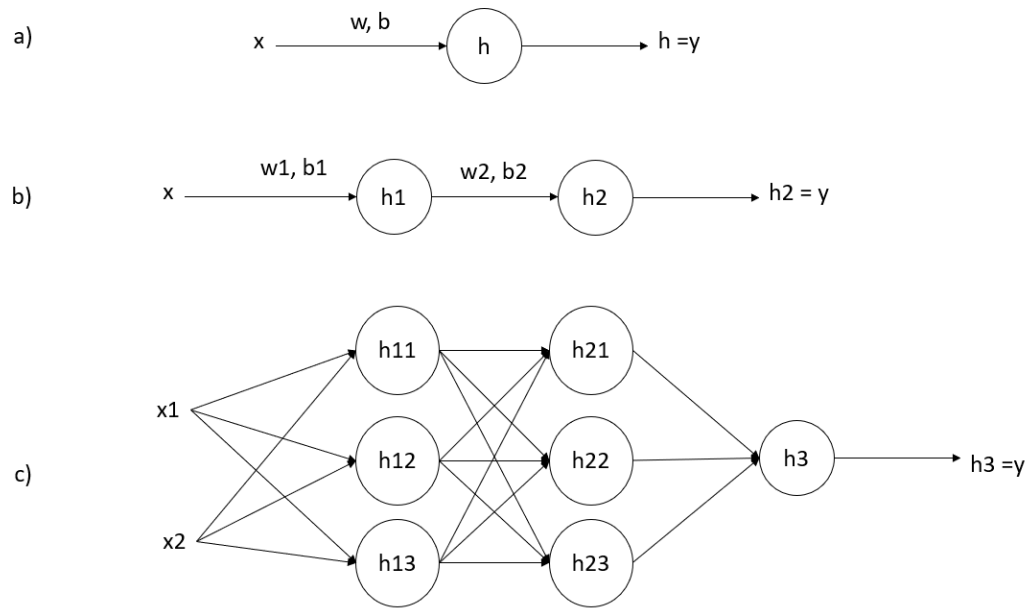


Figure 3.1: Basics of feedforward neural networks
a) perceptron, b) two-layer perceptron, c) multi-layer network

for the classes. For this, the model is redefined to have the *sigmoid* function:

$$f(x) = \frac{1}{1+e^{-(w^T x-b)}}$$

To measure the performance of the model, how good a given $\hat{y} = 0.912$ probability for an expected $y = 1$, the *cross-entropy* loss function is used:

$$L(f(x),y) = -\{y \log f(x) + (1-y)\log(1-f(x))\}$$

The E error function sums up the cross-entropy losses for every data sample to determine the model's performance, $E = \sum_i L(f(x_i),y_i)$. The *gradient descent* algorithm is a method to find the best w,b parameters by updating them in the direction of the negative gradient of the E error function [Had08].

A *neural network* is logistic regression units connected together. From one regression's output comes the input of the next one. The most well known learning algorithm to update the W,B parameters (where W contains the w parameter for every regression unit) is called *back-propogation*, which updates a multi-layer perceptron backwards [Kel60]. Figure 3.1 shows an example of logistic regression $a)$ and two more complex

neural networks.

3.1.1 Dropout

In many cases, we train a model using a training dataset (tune the w, b parameters), however, we want to use the model to predict y for a new input not represented in the training set. When a model performs well on the training set but predicts incorrectly the new inputs, we say that the model is *overfitting*, it learnt too much to lose the generality of itself.

Dropout is a technique to prevent neural networks from overfitting [SHK+14]. The idea is to exclude some random hidden units from every training step (e.g. remove $h21$ from Figure 3.1 in one step, $h23$ in another). The always-changing model reduces the chance of overfitting.

3.2 Recurrent neural network

A recurrent neural network (RNN) is suited for modeling sequential phenomena. Compared to the fully connected network, a RNN's h_t unit vector is calculated using its neighbour unit (h_{t-1}) and the corresponding input unit x_t using the formula:

$$h_t = f(Wx_t + Uh_{t-1} + b)$$

In theory, h_t can store all the information from the previous states, however, operating with long-range sequences can cause vanishing/exploding gradients [BSF94]. Figure 3.2 illustrates a simple RNN network using a sentence sequence separated by words.

Bidirectional recurrent neural networks use backward states as well as forwards [SP97]. They have a positive and negative direction update of the weights in the same step.

3.2.1 LSTM

Long short-term memory (LSTM) is an upgraded version of a recurrent neural network [HS97]. One step of a LSTM unit needs inputs x_t, h_{t-1}, c_{t-1} and produces h_t and c_t, where x and h represent input and hidden state respectively as in the previous section and c is a memory cell vector. To calculate h_t and c_t, a series of intermediate calculations takes place. i_t, f_t, o_t are referred to as *input*, *forget*, and *output* gates. See

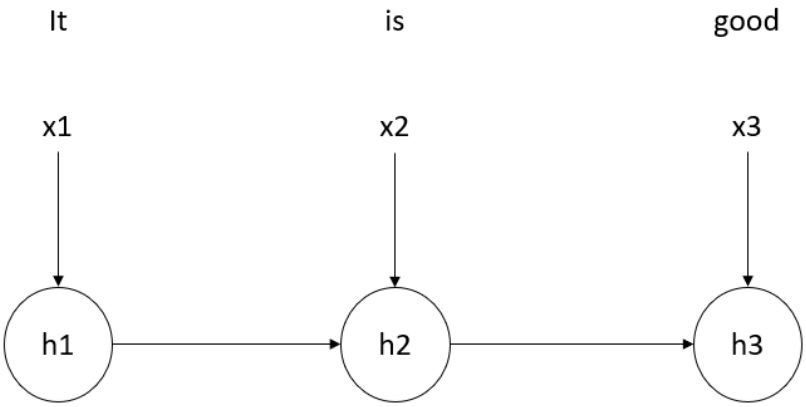


Figure 3.2: Basics of recurrent neural networks
3 units network with sequence input 'It is good'.

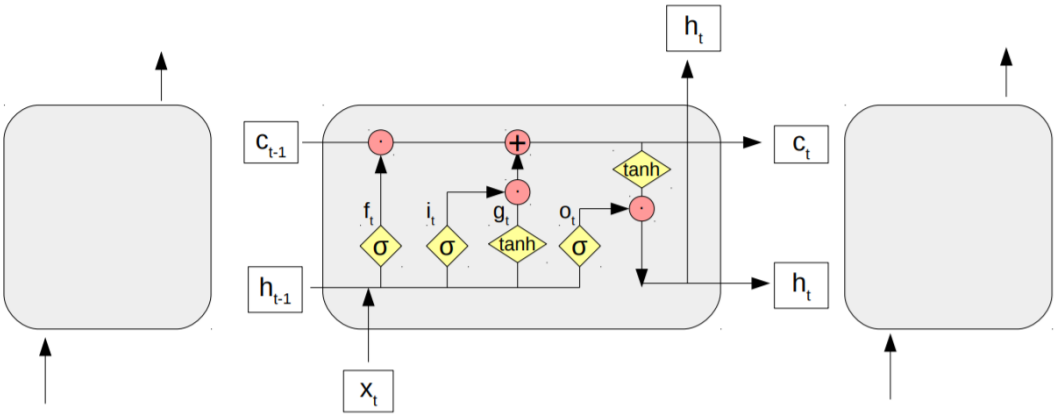


Figure 3.3: The inside of an LSTM neuron

visualisation and parameters in Figure 3.3[2]. The calculations are:

$$
\begin{aligned}
i_t &= \sigma(W^i x_t + U^i h_{t-1} + b_i) \\
f_t &= \sigma(W^f x_t + U^f h_{t-1} + b_f) \\
o_t &= \sigma(W^o x_t + U^o h_{t-1} + b_o) \\
g_t &= \tanh(W^g x_t + U^g h_{t-1} + b_g) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}
$$

where $\sigma(\cdot)$ and $\tanh(\cdot)$ are the element-wise sigmoid and hyperbolic tangent functions, $\odot$ is the element-wise multiplication operator. At $t = 1$, h_0 and c_0 are initialized to zero vectors. Learning parameters of the LSTM are W^j , U^j, b^j for $j \in i, f, o, g$.

3.3 Word representations in NLP

The year 2018 is referred as an ImageNet moment in Natural Language Processing (NLP) because in this year multiple large language modelling networks were published and researchers start to use them in transfer learning methods such as ImageNet is used in computer vision.

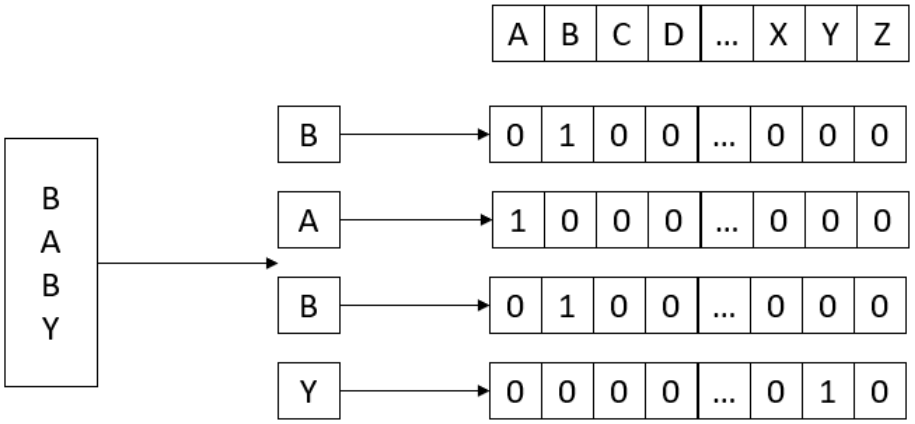
3.3.1 One-hot word representation

To process a word in a machine learning model it has to be represented numerically. The first approach of this is to represent every character in a one-hot vector of the alphabet or to represent every word as a one-hot vector of the dictionary.

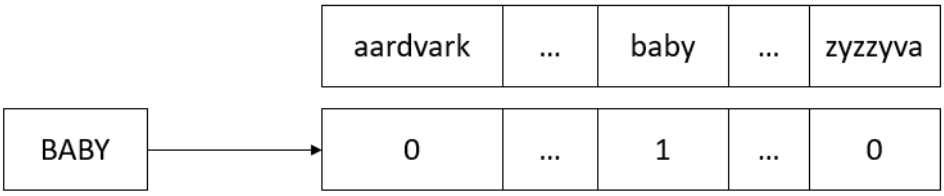
One the one hand, using the one-hot representation of the characters with the English alphabet, every character is represented in a 26 long vector where every element is 0 except the character, where it is a 1. This representation assigns a numerical vector to every word, however, the length of the words are different and many machine learning algorithm cannot handle it either.

On the other hand, the one-hot representation of the dictionary assigns the same length vector to every word. But, if we use a large English word dictionary, the representation of a single word will be a large vector. However, if we use a smaller

[2]Visualisation and more: `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`



(a) Character-level One-hot



(b) Word-level One-hot

Figure 3.4: One-hot representation of the word *baby* in the two different way

dictionary, there is a possibility that a future word will not be in the dictionary, thus we cannot assign a vector to it.

Figure 3.4 shows the difference between the two way of one-hot representation of the words in the English alphabet.

3.3.2 Word embedding

Alongside with the trivial problems stated in the previous section, these word representations do not have special nature scientists require from the model, whereas, the word representation should represent *semantically* close words close to each other in the vector space. Mikolov et al. showed that this can be achieved and introduced word2vec a word embedding that can represent the relation between similar words [MSC+13].

Word2vec is trained using one of two methods to produce embedding vectors: *Common Bag Of Words (CBOW)* and *Skip Gram*. CBOW uses the context of a word as input and tries to predict the right word according to the context. For example, if we have the sentence *The sky is blue.”*, we want to predict the word *sky* in context of the words *the, is, blue*. A similar sentence is *”The sea is blue.”*, therefore, the embedding

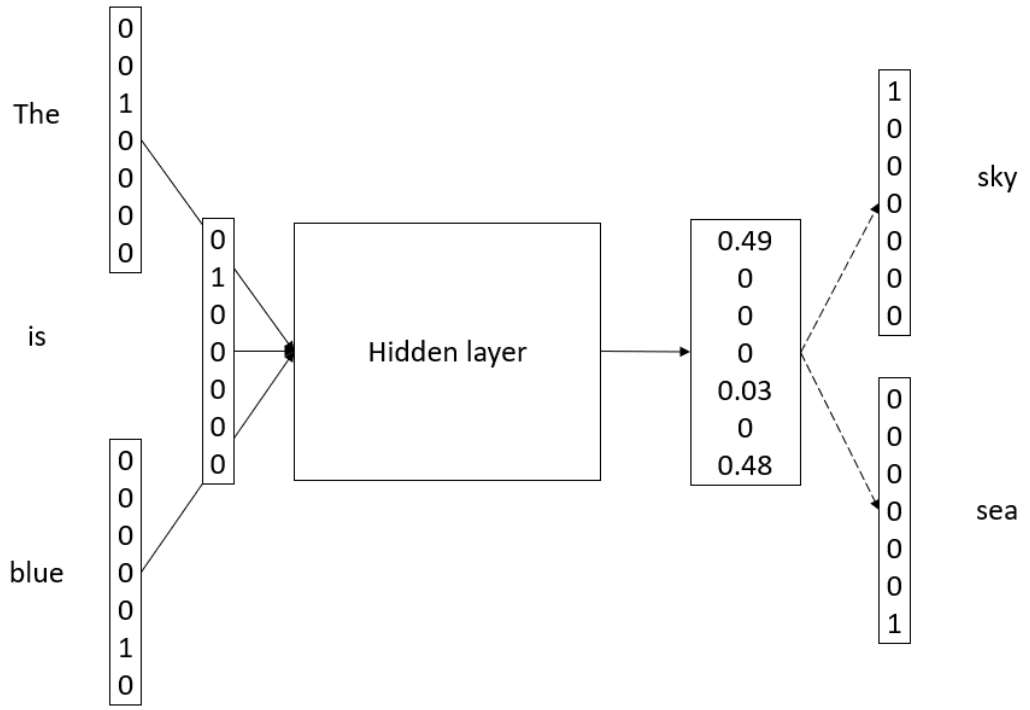


Figure 3.5: CBOW model example with the *The sky is blue* sentence

vectors of the two words *sky* and *sea* are relatively close to each other. The Skip Gram method is the reverse of CBOW. Here, the input is a single word and the target is to predict the relevant context.

The training of a word2vec model requires a large corpus with multiple occurrences of every word to represent them correctly. Using CBOW with c neighbour words, the input of the model is the c words one-hot encoded representation and the output is a softmax prediction of the most probable words. Where is the embedding vector? The Skip Gram method's hidden layer's numerical values represent the word. Figure 3.5 shows the CBOW architecture, the Skip Gram's architecture is the reverse of it.

A word2vec model represents a word with a single set of vectors, therefore, it does not have different values depending on the context. It finds the best vector to fit all the occurrences of the word. See the next section for contextualised embedding vectors.

Jay Alammar wrote a great summary about illustrated word2vec[3]. The following paragraph is based on his work. The model used here is from the official word2vec website[4], pre-trained in a 3 billion words Google News corpus where each word is

[3]See at (Accessed: 19th April, 2019): `https://jalammar.github.io/illustrated-word2vec`
[4]See at `https://code.google.com/archive/p/word2vec/`

represented in a 300 dimension vector.

If we try to visualise a single word2vec vector, we cannot see any features, but if we compare multiple words, we can observe similarities. In 3.6, we can see an image representation of the vectors for 8 words. The first word, *apple* is different from the other seven, which represent humans. If we look at the images, we can see a lighter line at the 42th column. Could it be the representation of human? The *queen* and *king* words share a similar dark cell at 2x46, does this show royalty?

If we try to reconstruct the word *king* from *queen* by changing its *woman*-likeliness to *man*, we get the 8th image. It is similar to the original *king*, but not the same. Is it the most similar word in the dictionary? The gensim[5] python package, used to work with the word2vec data comes with a function, that answers this question. Figure 3.7 shows that the king is truly the most similar word, as we expected (from the 500000 most frequent word in the dictionary). Another good example, if we want to know, what has the same relationship as *police* and *policeman*, if we use the word *ambulance*. The answer is *paramedic*.

These examples prove that word2vec embedding stores the relationship between the words.

3.3.3 Contextualised word embedding

However, a word itself does not always have the same meaning. For example *lap* can be a noun or a verb, with multiple meanings. Do decide which meaning has to be used, humans use the context of the word. Hence, the idea of contextualised word embedding.

In 2017, NLP researchers started to work on a word embedding, which uses not only the word but the sentence to determine the meaning of the word. [PABP17, MBXS17, PNI$^+$18].

The first contextualised embedding model, ELMo (Embeddings from Language Models) uses bi-directional LSTM models pre-trained to solve various tasks using sentence-level context [PNI$^+$18]. The idea behind ELMo is that it is trained to solve a problem called *Language Modelling*. Language Modelling aims to predict the next word in the sentence using the previous words. The bi-directional model means that it not only tries to predict the next word but the previous word as well. ELMo can be trained using a large number of general text corpora and the authors showed that

[5]See at `https://radimrehurek.com/gensim/`

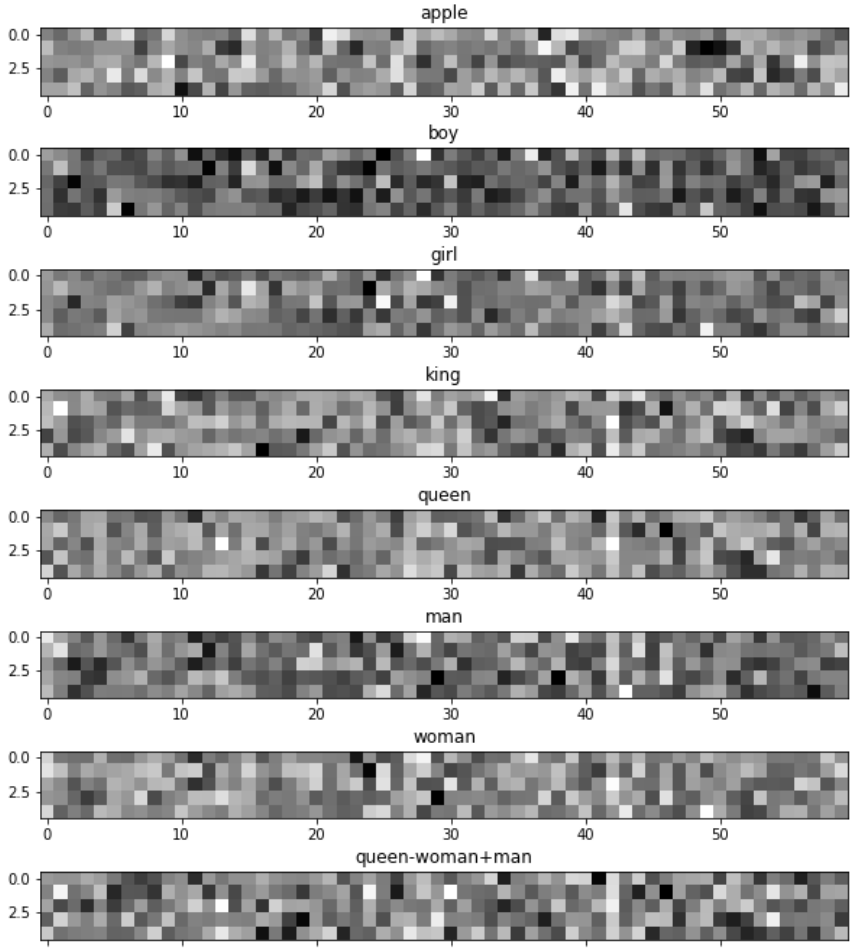


Figure 3.6: Word2vec vectors visualised in a 5x60 grayscale image

```
word2vec.most_similar(positive=["queen", "man"], negative=["woman"])
```

```
[('king', 0.6958590149879456),
 ('kings', 0.5950952768325806),
 ('queens', 0.5838501453399658),
 ('monarch', 0.5398427248001099),
 ('prince', 0.5223615169525146),
 ('princess', 0.5175285935401917),
 ('princes', 0.49844634532928467),
 ('royal', 0.4924592971801758),
 ('joker', 0.47121769189834595),
 ('Queen', 0.4703065752983093)]
```

Figure 3.7: Words similar to $queen - woman + man$

it can be used as a pre-trained component in various NLP problems such as question answering, textual entailment, semantic role labelling, coreference resolution, named entity recognition and sentiment analysis.

3.4 Transformers

In 2017, sequence-to-sequence architecture was introduced and based on it, transformers became popular models in NLP [VSP+17].

3.4.1 Sequence-to-sequence models

A sequence-to-sequence (seq2seq) architecture transforms a given sequence input to another sequence output. It is used in machine translation to translate text from a language to another [CVMBB14, SVL14], in question answering [YHG+16]. It also performs comparably well to RNN's in simpler problems such as hyphenation [Nem18]. As sequence data has no discrete length, the common practice to mark the beginning and the ending of the sequence with additional characters, tokens to notify the machine to terminate the program. The terminology used here is [CLS] token signals the start of a sequence and [SEP] signals the end of a sequence. Therefore the sentence *This is a nice sentence.* modifies to *[CLS] This is a nice sentence. [SEP]* in the preprocessing phase of the work. Other works also use [START] and [STOP] tokens or ^ and $ characters.

A sequence-to-sequence model consists of two separate LSTM based models. These are called Encoder and Decoder. Both LSTM networks take a sequence as input and another sequence as output. The *encoder* LTSM network's input sequence is the input sentence. Its output sequence is not used, only the hidden states of the LSTM vector. They are fed into the *decoder* network as its hidden states. The encoder network's tasks are different in the training phase and the prediction phase.

In the training phase, the decoder network's input is the target sequence, starting with the [CLS] token, and the output is the target sequence as well, ending with a [SEP] token. In the prediction phase, using the encoder's hidden states as *context*, the decoder generates a sequence starting from the [CLS] token, adding the predicted tokens to its input. This token generating step is repeated until a [SEP] token signals the end of the generated sequence (or it reaches the defined token limit).

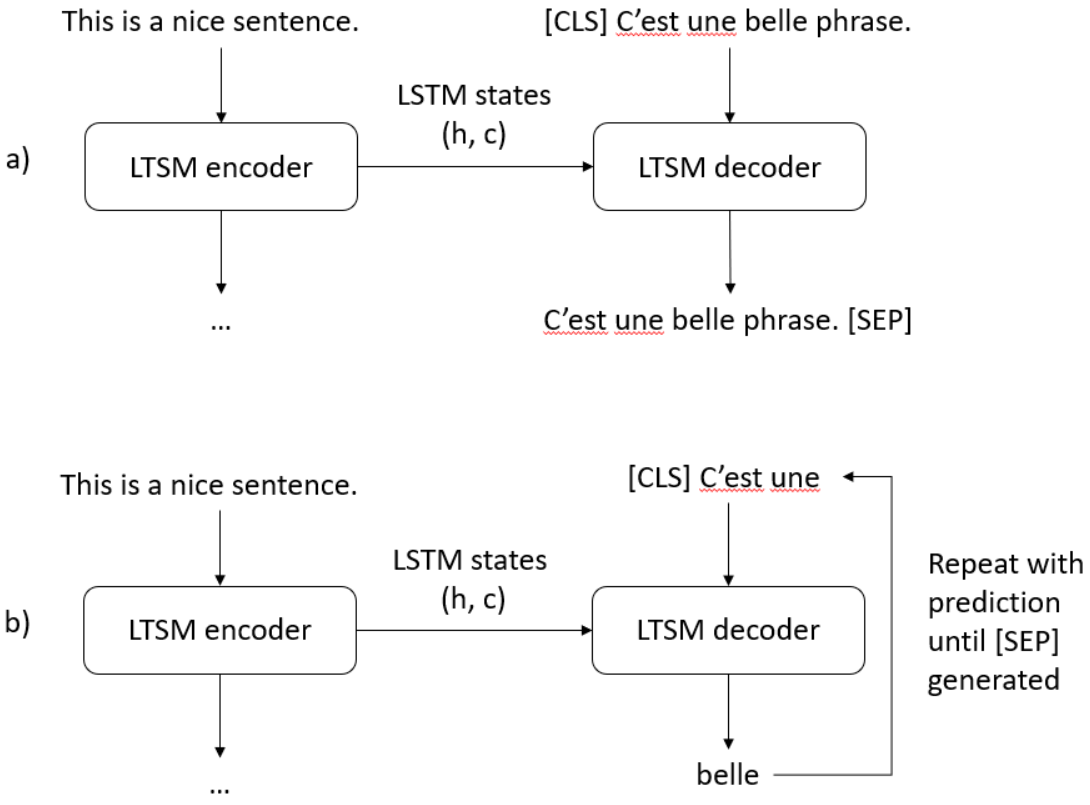


Figure 3.8: Sequence to sequence model architecture
a) training phase, b) prediction phase

3.4.2 Attention

A Transformer model differs from a sequence-to-sequence model in a way that the encoder and decoder networks use LSTM modules but attention based ones [VSP+17]. While the LSTM unit only sees the neighbours' states, the attention has a global function to reflect every token of the sequence:

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

Where Q is the query matrix, a vector representation of one word in the sequence, K is a matrix of all the keys (a vector representation of all the words in the sequence). And V is the value, a representation of a word. d_k is the dimension of the keys. Here, the $q = softmax(\frac{QK^T}{\sqrt{d_k}})$ is how the Q word affects all the keys. And the attention qV is how Q in the context of K modifies the word V.

In the original paper, there are three different attention layer. The *encoder-decoder attention* Q queries come from the decoder while the K and V keys and values from the encoder. This shifts the information from the encoder to the decoder in the same way as the LSTM network's hidden states in the previous seq2seq model.

Both the encoder and the decoder has a *self-attention* layer. It has all the Q,K,V data from the same encoder or decoder unit and has the same word as Q and V.

This *Multi-Head Attention* layer is the key part of the Transformer model's encoder and decoder, built with additional feed-forward layers [VSP+17]. Figure 3.9 is the original image of the Transformer architecture described by Vaswani et al.

3.5 BERT

BERT: Bidirectional Encoder Representations from Transformers is a language representation model based on the architecture described in the previous section [DCLT18]. The parameters of BERT are L, the number of Transformer blocks, H the hidden size (d_k in the previous section) and the number of self-attention heads, A. There are two model sizes used in the original paper, $\text{BERT}_{\text{BASE}}$: $L = 12$, $H = 768$, $A = 12$, Total Parameters: 110M and $\text{BERT}_{\text{LARGE}}$: $L = 24$, $H = 1024$, $A = 16$, Total Parameters: 340M. In this experiment, the $\text{BERT}_{\text{BASE}}$ is used, therefore, using the BERT embedding vectors, a word is represented in a 768 length vector.

BERT is designed to be a pre-trained model that can be fine-tuned with just one additional output layer for a specific task such as question answering.

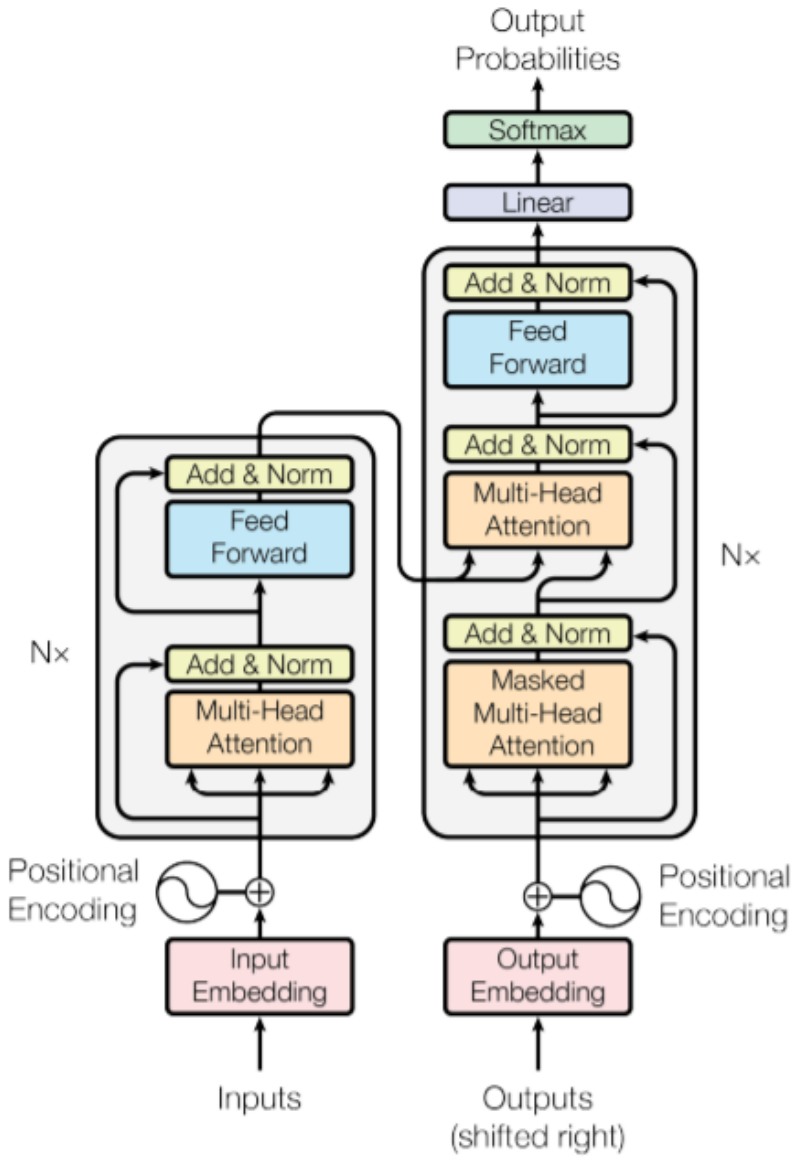


Figure 3.9: The Tranformer - model architecture

3.5.1 Masked LM

For pre-training BERT, the authors introduced two novel unsupervised tasks. The first one is called Masked Language Modelling (MLM). During the preparation, a percentage (15%) of the tokens are masked out randomly, and the goal of the algorithm is to predict only those masked tokens. This method is also called as *Cloze* [Tay53]. The final hidden vectors of the masked tokens are fed into an output softmax and becomes embedding vectors. For example, if we mask out the word *nice* in the sentence: *This is a nice sentence.*, the masked sentence is: *This is a [MASK] sentence.*

3.5.2 Next Sentence Prediction

The second task of the pre-training of BERT is the next sentence prediction. This is important for many natural language processing problems, where the task is to determine the relationship between sequences, such as question answering or the argument connection detection problem. The next sentence task, like the MLM task, is easily generated from a natural text corpus. For every sentence pair A and the following sentence B, the preparation algorithm creates an $AB-YES$ and an $AC-NO$ data sample. where the logical unit answers the question 'Is B following A?'. For example:

[CLS] My name is Gergely. [SEP] What is your name? [SEP]

has the label *IsNext* and

[CLS] My name is Gergely. [SEP] This is a nice sentence.[SEP]

has the label *NotNext*. Of course, in the training process, some of the words are masked out because of the other task.

The final pre-trained models achieve a 97%-98% accuracy on this simple task.

3.5.3 Pre-Training

The BERT models were trained on a concatenated corpora of the BooksCorpus (800M words [ZKZ+15] and the English Wikipedia (2,500M words). BERT was trained on 4 Cloud TPUs and took 4 days to complete.

3.6 Transfer Learning

Transfer Learning is a machine learning technique where a model trained for one purpose is reused for a second, related, task. Transfer learning is a widely used technique

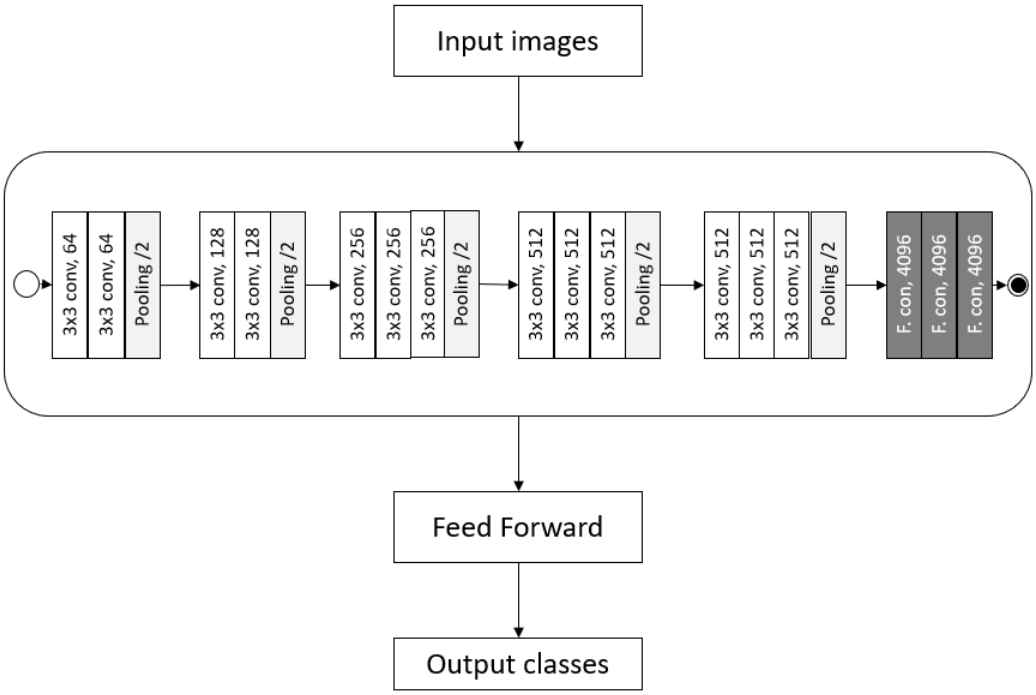


Figure 3.10: Transfer learning with VGG16

in Computer Vision due to the large available data and well-performing pre-trained models. Recently, natural language processing also achieved a state where models pre-trained in general data are useful for specific tasks.

These pre-trained models usually trained on a large dataset. The training may take days or weeks on a high-end machine but the final model can be used by users with less computational power.

A good example of Transfer Learning is using the ImageNet trained models for a specific image classification problem. ImageNet is a hierarchically structured image database containing more than fourteen million images classified into almost twenty-two thousand category [DDS$^+$09].

As ImageNet became a commonly used dataset to train and evaluate image recognition models, most state-of-the-art model's publication comes with pre-trained ImageNet weights and its scores in the 1000-class ImageNet challenge [RDS$^+$15]. Therefore, if someone wants to use a pre-trained image classification model for transfer

learning, one has many options, such as ResNet [HZRS16], VGG-16 [SZ14] and MobileNet [HZC+17]. As of today, Keras contains 20 different image classification models with pre-trained ImageNet weights[6].

Figure 3.10 illustrates a standard method of transfer learning is using a pre-trained image classification model. The input images are fed into a VGG model, a few feed-forward layers are added at the end of the network and it is trained with the specific task's image dataset. Image Transfer Learning is used in medicine for detecting brain or skin diseases [KBH16, MFP+17], to build self-driving cars [KP17, MLG+18] and many other applications.

However, due to the Internet, a lot of textual data is available, transfer learning for natural language processing has become popular just recently. The main reason for this is that NLP had no such a general task as image classification for Computer Vision. NLP models solved their specific problems and they were not easily reusable.

ULM-FiT (Universal Language Model Fine-tuning) has not only introduced a language model but it showed how the model can be fine-tuned for specific NLP tasks [HR18]. The ULM-FiT has three steps to solve a classification problem: a) general LM training, b) fine-tuning the LM model on the target task's data, c) training target task classifier. For example, let's look at the argument proposition type classification problem! For the general LM training, the model is trained on a large language corpus, like Wikipedia articles. Then, the fine-tuning of the language model requires argument propositions. Finally, the classification model uses the LM model as a basis to build a classifier over it that decides whether the proposition is a premise or a claim.

Both the ULM-FiT paper and BERT paper describes some examples the model fine-tuning was experimented on. Here is the list of tasks for the two LMs' fine-tuning experiments:

- Imdb Sentiment Analysis (ULM-FiT). This problem uses the IMDb binary movie review dataset [MDP+11] and the five-class Yelp review dataset [ZZL15] to build sentiment analysis based on the review scores.

- TREC: Question Classification (ULM-FiT). Six class semantic classification of open-domain, fact-based questions from the TREC database [VT99].

- Topic classification (ULM-FiT). Classifying text into topics using AG news and DBpedia ontology datasets [ZZL15]

[6]Keras Image Models, Accessed: 21st August, 2019: https://keras.io/applications/

- GLUE: General Language Understanding Evaluation (BERT). This benchmark is a collection of natural language understanding tasks [WSM+18].
  - MNLI: Multi-Genre Natural Language Inference is a sentence relation classification task where the aim is to determine the second sentence's entailment, contradiction, or neutral relation towards the first sentence [WNB17].
  - QQP: Quora Question Pairs is a sentence relation classification task where the aim is to identify semantically equivalent questions [CZZZ18].
  - QNLI: Question Natural Language Inference is a binary classification task [WSM+18] based on the Stanford Question Answering Dataset [RZLL16] where the positive samples are question-answer pairs where the answer is correct.
  - SST-2: The Stanford Sentiment Treebank is a binary sentiment analysis task [SPW+13].
  - CoLA: The Corpus of Linguistic Acceptability is a binary single-sentence classification task where the goal is to determine whether an English sentence is linguistically "acceptable" or not [WSB18].
  - STS-B: The Semantic Textual Similarity Benchmark is a sentence pair relation classification problem based on news headline similarity pairs [CDA+17].
  - MRPC: Microsoft Research Paraphrase Corpus is a semantic equivalence sentence pair classification corpus [DB05].
  - RTE: Recognizing Textual Entailment is a sentence relation classification task where the goal is the same as it is in the MNLI task [BCDG09].
- SQuAD: Stanford Question Answering Dataset (BERT) is a collection of 100k question-answer pairs where the task is to determine for a given question and a corresponding Wikipedia text containing the answer from the pair where is the answer passage in the text [RZLL16]. The training's loss function is the sum of the likelihood of the correct start and end token classified as start and end token of the answer passage. The second version of this task allows the possibility of missing answer passages.
- SWAG: Situations With Adversarial Generations (BERT) is a sentence-pair dataset of 113k pairs [ZBSC18]. For a given sentence the task is to select the most likely continuation from four options. Therefore this is a classification problem as well as the others.

Chapter 4

Related Work: Argumentation Mining techniques

There are several argument mining approaches as argumentation mining is a general task, therefore each approach addresses a variety of different goals to understand the arguments in the general text better. This is a summary of approaches in recent literature. Some of the subtasks described here can be used together to achieve a more general argumentation mining project and some of them help us understand the argumentation better.

In the recent years, argumentation mining become a popular field of the natural language processing. Starting in 2014 the annual meeting of the Association of Computer Linguistics (ACL conference) has held a Workshop in Argumentation Mining. This year's workshop was in early August, therefore, this thesis is not based on that. However, it contains some techniques using contextualised embedding vectors and this section covers them [Pet19, SKH19].

4.1 Argument Component Identification

The argument component identification (or argument component segmentation) is the very first step of argument mining. It is the process of selecting relevant text in the general text which can be part of an argumentation. It can be interpreted as a classification problem as a sentence (or sentence segment) is argumentative or non-argumentative. Moens et al. identified argumentation sentences using word pairs, text statistics, verbs and keyword feature [MBPR07]. Florou et al. used discourse markers and features extracted from the tense and mood of verbs [FKKK13].

Hence	it	is	always	said	that	competition	makes	the	society	more	effective	.
O	O	O	O	O	O	B	I	I	I	I	I	O

Table 4.1: Clause level argument component identification

The task can be approached as a sentence-level classification. In this case, the assumption is that every sentence is either an argumentative one or not. However, others may argue that argumentative parts of the text cannot be identified by sentences. In the other case, the task is more complex than a simple classification. It requires a sequence identification in the text to determine the argument discourse units (ADUs, argument propositions). The majority of the works applies classic machine learning models and focuses on collecting relevant features from the text [LT16]. The list of machine learning models used in the sentence segmentation problems includes Naive Bayes [MBPR07], Support Vector Machines (SVMs) [RWB12, SG14b], Logistic Regression [GLPK14, LBH$^+$14], Decision Trees and Random Forests [GLPK14, SG14b].

Several approaches use pipelines for clause identification: one step is the argumentative part classification and another step is the identification of the boundaries [GLPK14, SG17]. Argumentation clause identification can be applied as token-level labelling: for every token in the text, the classifier choose from classes *B*, *I* and *O* refering to the **B**eginning, **I**nside and **O**utside of an argumentative clause. To restore argumentative parts, one has to merge the *B* and *I* labelled tokens.

Conditional Random Fields (CRFs) are applied in sequence level chains: from sequence input the classifiers generates sequence-labelled output. Stab and Gurevych built an annotated corpus called Student Essay Corpus and achieved a F1-score of 0.86 on this task [SG17]. One example to build this classifier is to use LSTM networks. Ajjour et al. achieved a F1-score of 0.88 using bidirectional LSTMs [ACK$^+$17]. Eger et al. built an end-to-end biLSTM network for multiple supervised argumentation mining tasks [EDG17]. Their model scored 0.69 on the Argumentative Essay Corpus. Petasis used contextual word embedding (Flair [ABV18] and BERT [DCLT18]) as token-level features to achieve a 0.90 score [Pet19]. Using a rule-based approach Persing and Ng achieved a F1-score of 0.92 [PN16].

An example of the sentence *"Hence it is always said that competition makes the society more effective."* in the argumentative essay of Appendix A (example from the

manually annotated Student Essays Corpus [SG17]) would be identified as an argumentative sentence in the sentence-level approach. However, using token level approach (words as tokens, punctuation character is a different token), the sequence labels are described in table 4.1. The annotated argument component part of the sentence is *"competition makes the society more effective"*.

The clause-level argument structures allow multiple argumentative parts in the same sentence. An example of this situation from the Student Essays Corpus is the following sentence: *"To some extent,* ***the gap between rich people and poor is widen by technology****, because only the rich people have ability to afford the expensive high-tech products, and* ***these products can help them to earn more in return****."* It contains two argument proposition in the corresponding annotation. The argumentative parts of the sentence are bold.

4.2 Argument Component Classification

In this subtask, the goal is to determine the type of argument proposition. Whether it is a *premise*, *claim* or *conclusion*. Or, if the used Argument Theory uses different types, then classified to the used types. It can be part of an argument component identification classifier using the *non-argumentative* class as an additional class.

Burstein and Marcu implemented models to classify argument thesis and conclusion propositions[BM03]. Kwon et al. used a two step pipeline to achieve the task in online comments [KZHS07]. In the first step, they classified the sentences as claims, then the claims into support, oppose or propose types. Their claim F1 score was 0.55 and second-type classification F1-score was 0.67. Rooney et al. merged the previous and this task together using a premise, claim, non-arguentative classifer [RWB12] achieving a 0.65 score. Mochales-Palau and Moens [PM09] and Stab and Gurevych [SG14b] used SVMs for premise-claim classification and Lippi and Torroni used partial tree kernels [LT15]. The latest state-of-the-art premise, claim, major claim classifer from Stab and Gurevych achieves 0.77 F1 score [SG17].

The persuasive essay example in Appendix A contains one conclusion, 5 claims and 8 premises. Examples of each type:

Conclusion *"we should attach more importance to cooperation"*

Claim *"without the cooperation, there would be no victory of competition"*

Premise *"The winner is the athlete but the success belongs to the whole team"*

4.3 Argument Connection Detection

In this subtask, the goal is to determine whether two arguments are connected in the current text or not. Argument pairs are connected with a directed link. One proposition (a premise) can support or attack the other proposition (claim or conclusion). Therefore, the task is to classify between five options: the proposition A and B are non-related, they are related and A supports B, B proposition supports A, A statement attacks B or B attacks A. The related works are usually separates them to two different tasks: identifying the connection (related or non-related pairs) and determine the type of connection (support or attack).

To determine the relation between propositions, two approaches can be applied. Firstly, depending only the proposition pairs to determine the connection type between them (described here) or using the whole argumentative text to build a structure (next section). In the first version, the task is a simple classification for every pair of propositions. However, the second version uses full text level restrictions. For example, in an Argumentation Theory using tree models to describe an argument, there must be only $n-1$ link in an n proposition argument without any linked circle (this restriction comes from the definition of a tree). Meanwhile, if one uses only proposition pairs, these type of restrictions are not applied.

Stab and Gurevych built a proposition pair binary classifier (related or non-related pair) using SVMs and achieved a 0.72 F1 score on student essays [SG14b]. Their continuous work uses a two-step approach [SG17]. Firstly, it classify as related or non-related (F1 score of 0.73). Secondly, it classifies as support or attack relation (F1 score of 0.70). Peldszus introduced a proposition pair relation detection model using global information from the other propositions as well [Pel14].

An example of a support relation from the Appendix A: *"without the cooperation, there would be no victory of competition"* **supports** *"we should attach more importance to cooperation"*.

An example of an attack relation: *"competition can effectively promote the development of economy"* **attacks** *"we should attach more importance to cooperation"*

4.4 Argumentative Structure Determination

In this task, we assume that the argumentative structure can be represented as a tree with a conclusion root and supporting arguments as branches. With this assumption,

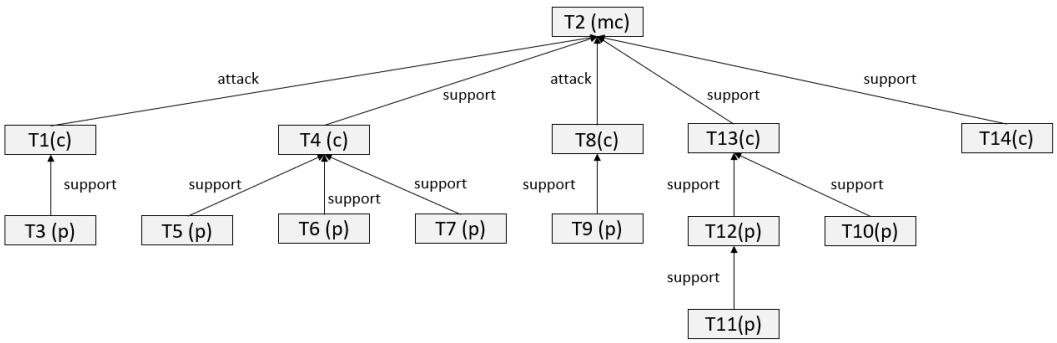


Figure 4.1: Argument tree structure of a persuasive essay

the task is to determine this tree. One way to solve this task is based on the previous task: if we have the correct connections between the propositions, we can build a tree from that. The tool built in this experiment uses this to build a argument structure.

Mochales-Paulu and Moens introduced a argument structure tree modelling approach [PM09], however, their approach only recognises arguments with discourse indicators. Peldszus and Stede implemented a Minimum Spanning Tree searching model that globally optimises argumentative relations [PS15].

Figure 4.1 is an example of a tree structure based on the argumentative essay described in Appendix A. There are several argument illustration tools but this thesis does not cover them. The figure contains the argument propositions of the essay (represented by their ID and their proposition type), and the relations between them, labelled with the type of the relation.

4.4.1 Topic Similarity

A possible way to solve this task is by using the similarity of the propositions. If we assume that the arguments are following each other depth-first, the algorithm is stated this way: if a proposition is topically similar to the previous one, it supports the predecessor. Otherwise, moving up the tree we select the most similar proposition and make the new one as a supporting branch of that one. If we do not find any similar topic, the proposition is determined to be irrelevant to the existing structure [LRM$^+$14, LR15].

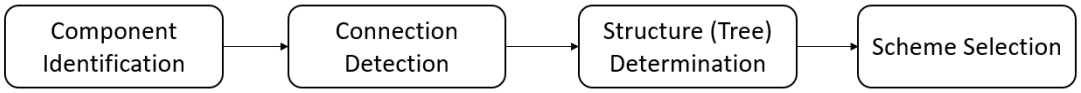


Figure 4.2: Architecture of an general Argumentation Mining parser

Lawrence et al. uses Latent Dirichlet Allocation (LDA) topic model for the comparisons [LRM+14], Lawrence and Reeds uses WordNet's similarity function[1]. WordNet as part of the NLTK python package can be used to determine sentence similarity[2], but spaCy[3] has its own sentence level similarity as well.

4.5 Scheme Structure Selection

In Chapter 2, a brief introduction of the argument schemes is described. This subtask aims to determine the best matching scheme structures for the arguments. The first discussion of this task is from Walton who proposes a six-stage process to identify the arguments and their stages. It starts with an argument component identification and then tries to fit the arguments to an argument scheme from a list [Wal12]. Feng and Hirst propose a similar algorithm with classifiers for the known argumentation schemes [FH11]. Lawrence and Reeds use the scheme structure selection as an independent task as well as part of a combined method [LR15].

4.6 Combined methods

To achieve a more general argument mining model, we have to combine these steps with as accurate predictions as possible. Figure 4.2 summarises the architecture described in this chapter. The relevant literature may use different notation.

There are several existing combined methods, mentioned in the previous section [Wal12, LR15, SG17]

[1] See at: `http://wordnet.princeton.edu/`

[2] See a blog post for the required steps (Accessed: 17th April, 2019): `https://nlpforhackers.io/wordnet-sentence-similarity/`

[3] `https://spacy.io/`

Lawrence and Reed		Milz
Support	Attack	Keywords (overall 195 indicators)
because, therefore, after, for, since, when, assuming, so, accordingly, thus, hence, then, consequently	however, but, though, except, not, never, no, whereas, nonetheless, yet, despite	above all, accordingly, actually, by the way, certainly, clearly, consequently, further, given that, hence, however, similarly, simply because, since, so that, wherever, yet

Table 4.2: Discourse indicators: keywords used by Lawrence and Reed [LR15] and by Milz [Mil17]

4.7 Discourse Indicators

Discourse indicators (discourse markers) are linguistic expressions to connect statements [WEK12]. If present, they can determine the relationship between statements, indicating the argumentative structure. For example, in the original text of Figure 5.1, the statements ”We can create all the legal issues, laws and strict gun control we want.” and ”if someone wants to kill, they could kill someone with a pen, pencil, knife, baseball bat or even a slingshot” are connected with the discourse indicator *but*, therefore we can assume that it is a conflict and the second statement attacks the first one.

Discourse indicators can be used as features of argument mining classification [SG14b, Mil17], or in their [LR15] to determine connections between arguments.

Either complex parser can be trained to identify and categorise discourse indicators [LNK14], or they can be used with simple keyword search [LR15, Mil17]. Examples of discourse indicator keywords are in Table 4.2.

4.8 Feature Selection

As we saw in the previous section, discourse indicators can be used as features of an argumentation mining model. But what are the relevant features of a general text? Many works of literature aim to answer this question [LJN10, SEW15].

There is two way of extracting features from the argumentative text: a) local word-by-word features (e.g. word2vec embeddings [MSC$^+$13] or Part-of-Speech tagging) and b) global argument-wise features which can be sentence-related or paragraph-related (e.g. sentence similarity, keywords, similar verbs). The following is a list of the features used in the related literature.

4.8.1 Local features

The local features are usually applied as CRFs (Conditional Random Fields). A linear chain CRF maps a sequence of tokens to a sequence of labels.

Part-of-speech tags

The part-of-speech tag of every token (nouns, pronouns, adjectives, verbs, adverbs, prepositions, conjunctions and interjections) [LR15, SG17, Mil17, SEW15]. Some experiment only uses verbs and adverbs [MBPR07] while others use every PoS tag.

Unigrams, bigrams

Unigram words and two-length word sequences. Can be part of the feature space as it is in the argument [LR15] or preprocessed (e.g. lemmatised) [SG14b, SEW15, SG17]. Sometimes trigrams also used [MBPR07].

Token positioning

The starting position (offset) of the tokens. Features can include every token in the text or only relevant tokens (e.g. verbs). Both the position in the sentence and the whole argumentative text can be relevant [SG17].

Lowest common ancestor (LCA)

The dependency tree of a sentence is a structure that aims to determine the connections between the words of the sentence. As of today, most Natural Language Processing Toolkit provides a dependency tree between the tokens of the sentences. Both local and global features are extracted from the dependency tree and used in modelling. The first common ancestor of two tokens in the dependency tree is a local feature of every word pair in the sentence. Additionally, the types of LCAs and the number of LCA in every type are global features [SG17].

Embedding vectors

Word2vec [MSC+13] is a model used to generate embeddings for words. It generates a mapping function from a large corpus in a way that every word in the corpus has a corresponding vector in the high-dimensional vector space output such as words sharing similar linguistic contexts are located in close to each other. The usage of

word2vec in argumentation mining is relatively new, but it can be recognised as a local feature in the same way as PoS for every word (however a word2vec vector is comparably larger than a PoS tag) [Mil17, SG17]. Stab and Gurevych uses the sum of the word vectors to minimise the size of the vectors.

Contextualised embedding vectors such as BERT were used previously, only as a token-level replacement of the word2vec vectors [Pet19, SKH19]. To the author's knowledge, BERT as sentence level embedding is a novel feature in this experiment.

4.8.2 Global features

Length and average sentence length

The total length of the argument and the average length of each sentence in the argument are commonly used features of the Argumentation Mining models [MBPR07, SG14b, LR15, SEW15, Mil17].

Keyword and punctuation presence

The discourse indicators were already introduced as important keyword features. There are other used keywords in several papers as well as punctuation characters used as features. The number, type and positioning of the punctuations can be helpful in these classification problems [SG14b, SG17, LR15, Mil17, MBPR07, FKKK13, SEW15]. Moens et al. refer to the discourse indicators with the term modal auxiliary.

Similarity

The similarity of the arguments (sentences) can be used to determine the connection between the sentences (see section 4.4.1). The n-grams can be similar to a predetermined proposition type. For example, the statement "his/her opinion" can be used as an indicator of the scheme type Expert opinion [LR15]. Also, words that are similar to the keywords in a word2vec vector space can be potential discourse indicators. A simple sentence similarity (same sentence binary feature) is used in Milz's work as well as [Mil17].

Milz uses a shared noun feature to count the number of shared words of the propositions [Mil17]. An expanded version of this, a shared, stemmed word counter is included in this experiment's feature selection process.

Dependency tree properties

Depth of the tree for each sentence, number of subclauses [MBPR07, SG14b]. Production rules [LKN09] are function tag connections (e.g. $S \rightarrow NP$) can be collected from the dependency tree. Lin et al. use a binary list for the occurrences of every possible pairs [LKN09, SG14b].

Chapter 5

Dataset

One of the challenges of the current state of argumentation mining is the lack of large quantities of properly annotated arguments. There are several approaches of argument definition, description and argumentation schemes (see section 4.5) and the scientists building argumentation corpus do not have a consensus to use therefore the available corpora lack unified structure.

Tobias Milz proposes an argumentation corpus parser[1] to parse certain argumentation corpora into a unified model and format. His ArguE classifier is based on these corpora and shows results comparable with methods using the original corpora and scheme model [Mil17]. However, this parser cannot solve several of the problems with the publicly available corpora, e.g. the lack of original text.

5.1 ArguE Unified Corpus Format

The ArguE corpus format is an XML file. Each argument component is listed as a ”Proposition” with the following child nodes:

- ADU: The Argumentative Discourse Unit tag contains the type of the argument proposition: premise, claim or conclusion
- text: The text of the argument proposition.
- textPosition: The starting and ending offset of the proposition in the original text. If the original text is unknown, it is set to -1.

[1] Available at `https://github.com/Milzi/arguEParser`

- Relation: Connection with other propositions. It has a unique identifier and a pointer to the partner. It has a relation type which can be ”Default Inference” (support), ”Default Conflict” (attack), or any specific type of discourse relation such as ”Analogy” or ”Expert Opinion”. These descriptors are defined by the AIF standard. The relation also contains a binary relation indicator which defines whether it is any kind of attack or support.

The original text is in a separate XML node.

Figure 5.1 is an example of the unified corpus from the Araucaria corpus. Appendix A is an example of a Student Corpus Essay annotation. A quick observation of the figure can show the issue with the Araucaria database: some of the propositions are not in the Original Text (e.g. proposition id 309).

5.2 AIF-DB Corpora Collection

The AIF-DB [LR14] is the largest collection of publicly available argumentation annotations and argument maps [2]. It contains several corpora, most famously the AraucariaDB [RPR+08] which is used in multiple argumentation mining literature. The project proposes an Argument Interchange Format (AIF) and the argument map creation tool OVA[3]. The AIF is designed for the collection of argument maps and not necessarily argumentation annotations, therefore, its format is not fitting the argumentation classification problems as well.

Most of the corpora in the collection offers only arguments and lacks the original text, making the corpus unusable for several argumentation mining subtasks. Another drawback of the datasets in the AIF-DB collection are containing a relatively small amount of argumentation structures and these structures contain only a few arguments, e.g. only a premise and a supporting claim.

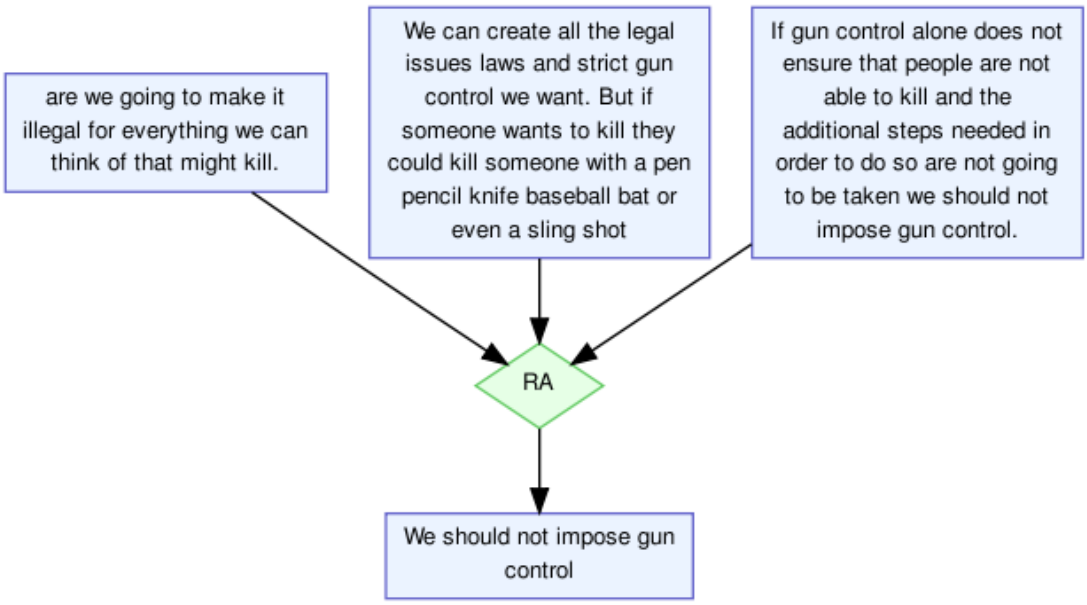
5.2.1 AraucariaDB

The Araucaria project is a software tool for analysing arguments and the AraucariaDB is an argument corpus based on this tool [RPR+08]. The collection of the corpus is commenced in 2003 and contains diverse sources (newspaper editorials, parliamentary

[2] See at: `http://corpora.aifdb.org/`
[3] See at `http://ova.arg-tech.org/`

```
<Annotation corpus="araucaria">
  <Proposition id="309">
    <ADU type="conclusion"/>
    <text>We should not impose gun control</text>
    <TextPosition start="-1" end="-1"/>
  </Proposition>
  <Proposition id="310">
    <ADU type="premise"/>
    <text>are we going to make it illegal for everything we can think of that might
kill.</text>
    <TextPosition start="192" end="271"/>
    <Relation relationID="313" type="Default Inference" typeBinary="0" partnerID="309"/>
  </Proposition>
  <Proposition id="312">
    <ADU type="premise"/>
    <text>If gun control alone does not ensure that people are not able to kill and the
additional steps needed in order to do so are not going to be taken we should not impose
gun control.</text>
    <TextPosition start="-1" end="-1"/>
    <Relation relationID="313" type="Default Inference" typeBinary="0" partnerID="309"/>
  </Proposition>
  <Proposition id="311">
    <ADU type="premise"/>
    <text>We can create all the legal issues, laws and strict gun control we want. But
if someone wants to kill, they could kill someone with a pen, pencil, knife, baseball
bat or even a sling shot</text>
    <TextPosition start="0" end="187"/>
    <Relation relationID="313" type="Default Inference" typeBinary="0" partnerID="309"/>
  </Proposition>
  <OriginalText>We can create all the legal issues, laws and strict gun control we want.
But if someone wants to kill, they could kill someone with a pen, pencil, knife,
baseball bat or even a sling shot. So are we going to make it illegal for everything we
can think of that might kill. Musa , USA</OriginalText>
</Annotation>
```

(a) Unified corpus format using Milz's parser



(b) OVA representation at AIF-DB

Figure 5.1: AraucariaDB corpus Argument Map 16 representations

records, judicial summaries, discussion boards). Originally, it was built in an Argument Markup Language, but it went under several changes as it becomes part of the AIF-DB collection and included in Milz's Unified Corpora. Currently, the corpus contains 662 arguments with annotated premises and claims in a tree-like structure. A few arguments even have argumentation scheme too. However, the database does not include the original text only the argumentation relevant parts.

The Araucaria database contains original text information, however, it does not include the full text every time. For example, in the argument described in Figure 5.1, the argument conclusion *We should not impose gun control* is not in the corresponding OriginalText element of the XML file. To solve this issue, in the preprocessing phase of model training with full-textual features, the argument propositions without correct position field are removed.

5.3 Student Essay Corpus

The Student Essay Corpus v2[4] contains essays from an online forum collected and annotated by Stab and Gurevych [SG17]. The arguments contain premises, claims and major claims. The major claim is the overall conclusion of an essay, thus, in the Unified Corpus Format, they are conclusion type propositions. The corpus annotates support and attack relations and the original essays are available therefore it is useful for argumentation part identification. It was annotated with the BRAT annotation tool[5].

This corpus contains the original text of the argumentation, therefore the position of the arguments and the original sentences are available. These additional features give more precision to the models, however, cannot be used with corpora without this information.

The original files of the Student Essay Corpus contain paragraph-level information. Every paragraph has an annotation of For or Against type. These are similar to the support and attack relations of proposition pairs, just paragraph-level links. The Unified Corpus Format does not include this information, therefore, the models of this experiment are not supported with these features.

Similarly to the Araucaria, the Student Essays Corpus' Argument Theory is based on a tree structure, therefore every argument with n proposition contains $n-1$ relation between the propositions. Each essay has exactly one conclusion and several premises

[4] `https://www.informatik.tu-darmstadt.de/ukp/research_6/data/index.en.jsp`
[5] See at `http://brat.nlplab.org/`

and claims. Table 7.1 and 7.6 contains information about the data distribution of the final train and test sets built from the corpus.

5.4 ArguE Corpus

The ArguE Corpus is an argument annotation created by Milz and designed for the unified corpus [Mil17]. It contains 8 debates and it was annotated using the BRAT tool.

This corpus was built with the Unified Corpus Format so it can support the validation of the protocol. However, the corpus was available during the experiment, the results of the other, larger corpora suggested that the focus of the experiment should involve the two other corpora, therefore, this corpus was not used in the experiment.

Chapter 6

Research methodology

The following chapter describes the algorithms and models used in the experiment to classify argument and argument relation types. The last section contains a handful description of the downloadable tool available on GitHub.

The *proposition type classification* contains information about the proposition and aims to classify whether it is a premise or claim based using machine learning.

The *argument relation classification* pairs arguments and aims to determine whether there is a connection between the propositions and if so, it is a supporting or attacking relation.

6.1 Data collection and preprocessing

Compared to the AIF-DB collection, the Student Essay Corpus v2 [SG17] contains additional information because it has the original texts of the arguments. Therefore, additional features can be extracted from this corpus that may not be extracted from other ones.

The ArguE Unified Corpus Format contains the arguments and argument relations in an argumentation grouped way. For the proposition type classification and relation classification problems, the data needs to be transferred to an argument-based database.

To classify connections between propositions, proposition-pairs are linked together. To be able to decide between related and unrelated propositions, there must be unrelated pairs in the dataset. Therefore, during the preparation, every proposition is connected to every other one. This is a complete graph regarding the propositions as the nodes. As Chapter 2 describes, an argument structure usually represented as a tree. Therefore, the number of proposition-pairs in an argument containing n propositions

Name	Description
arg1, arg2	The text of the two propositions
originalArg1, originalArg2	The original sentence of the propositions.
label	Either the proposition type (ADU) or the argument relation type.
fullText1 *	The full text of the argumentation.
positionDiff *	Difference between the starting character of the two proposition in the full text divided by the length of the full text.
sentenceDiff *	Distance between the original sentences divided by the number of sentences in the full text.
positArg *!	Distance from the beginning of the full text.

Table 6.1: Data collected from the Unified Corpus Format
(*): only available if the original text exists (!): collected only for the argument type detection

is $\binom{n}{2}$, while the number of related proposition pairs is $n-1$. Regarding the difference between the size of the related and unrelated proposition-pair groups, data balancing is required during the preparation.

The preparation can also include classic text mining preprocessing tasks, such as lowering, however, it may include additional information for feature selection. Table 6.1 shows the features collected from the Unified Corpus Format after using Milz's parser [Mil17].

6.2 Feature selection

Table 6.2 summarises all the features used in this experiment. Some of these features are extracted from a single proposition (e.g. keyword features, length), some of them aim to capture the connection between the two propositions (e.g. shared words) and some of them describes the position of the propositions in the full text (e.g. positionDiff).

Section 4.8 summarised the features used in related projects, this section introduces the features used in this experiment.

For the majority of token dependent features, NLTK's tokenizer is used [LB02], however, BERT uses it's own tokenizer [DCLT18].

Name	Data Type	Description
positionDiff *	0-1 float	Difference between the starting character of the two proposition in the full text divided by the length of the full text.
sentenceDiff *	0-1 float	Distance between the original sentences divided by the number of sentences in the full text.
rstCon *	logic	The two propositions' nodes shares a link in the RST tree.
rstConParent *	logic	The two propositions' nodes shares a parent in the RST tree.
vector -2	300xTL	Word2vec vectors of every token in the propositions, original sentences.
pos -2	35xTL	Part-of-Speech tags of every token in the propositions, sentences
premiseIndicator, claimIndicator 2	int	Number of keywords in the propositions.
tokens 2	int	Number of tokens in each proposition.
sharedWords 2(*)	logic & int	Shared words, nouns, verbs between the propositions, sentences.
sameSentence	logic	True if the two proposition has the same original sentence.
bert 2	768 floats	Sentence level BERT embedding for the propositions, original sentences
bertVector -2	768xTL	Token level BERT embedding vectors.
sentCompound, sentNeg, sentNeu, sentPos 2(*)	0-1 float	Sentinent scores extracted from the propositions, sentences and full texts if available.

Table 6.2: Features of the models
(*): only available if the original text exists, (-): token-level vectors, (2): extracted from both proposition simultaneously. TL: token length

6.2.1 ArguE features

These features originally used in Milz's work [Mil17], available at GitHub [1].

Word2vec vectors

For every token in the proposition's token list, ArguE generates a 300 length vector embedding using Mikolov's Word2Vec representation [MSC+13]. For proposition pairs, both propositions have their sequence. To remove problems caused by the different proposition token lengths, the algorithm uses a method called padding. Padding fills up the shorter propositions with zero value vectors to match the longer ones. For *padLength* padding size, the minimum value of the padding size has to be at least as big as the longest proposition's token length $padLength \leq \max(tokenLength(p)); p \in propositions$. Therefore, using $padLength = 30$, this method generates a 300×30 feature matrix for every proposition.

Part-of-Speech vectors

Using the NLTK toolkit's Part-of-Speech tagging [LB02], every token gets a PoS tag. For example: *NN* noun, singular 'desk', *NNS* noun plural 'desks', *VB* verb, base form 'take', *VBD* verb, past tense 'took'. The complete list of the NLTK PoS tags contains 35 different labels [2]. For computer accessibility, the labels are stored using one hot encoding. Padding for the different token lengths also applied here.

Premise and claim Indicators

This feature searches for matching premise or claim keywords. The original code checked for exact matching, but in this experiment, the text is lowered before the matching. Table 4.2 shows examples of the keyword indicators.

Tokens

Simple integer number of the tokens in each proposition.

[1] Available at: `https://github.com/Milzi/ArguE`, updated version: `https://github.com/negedng/ArguE`

[2] An example of the list (accessed at 16th August, 2019): `https://pythonprogramming.net/natural-language-toolkit-nltk-part-speech-tagging/`

Shared words

The original version only checked for exact noun matches. In this experiment, shared words, nouns and verbs are used with a few additional parameters. A length threshold is applied to eliminate the common words like *a* or *the*. Also, there is an option to use word stemming to match words with the same origins (e.g. *beginning* and *began* shares the same origin *begin*). To collect stemming information, NLTK's PorterStemmer [3] is used [LB02].

Same sentence

In most cases, the propositions of an argument are not complete sentences. In these cases, the propositions sharing the same original sentences are usually about a related topic. Therefore, this might be an important feature of the model.

6.2.2 RST features

Discourse structures describe the high-level connections between parts of a general text or speech. Rhetorical Structure Theory (RST) is a hierarchical discourse structure representation [TM06]. Discourse structures are used in many applications, such as text summarisation [LJN10], question answering [FBCC+10] or sentiment analysis [VT07]. However, the performance of discourse parsing is weak. The F-score for text-level relation detection in Ji and Eisenstein's state-of-the-art model's performance is only at 61% [JE14].

In this experiment, Ji and Eisenstein's parser is used to generate a text-level relation for the full texts of the Student Corpus Essays [JE14]. The parser is available at GitHub[4]. Unfortunately, these features are not available for the AIF-DB collections. This parser creates a tree-like structure for the text propositions with labelled relation types.

After matching the argument propositions with the ones generated by the parser, this experiment uses two features: *rstCon* is a logical variable, signalling true if the two-argument propositions are connected in the RST parser's representation and *rstConParent* signalling if the two share the same parent node.

[3] NLTK Stemmer: `http://www.nltk.org/howto/stem.html`

[4] RST parser (accessed at 18th July, 2019): `https://github.com/jiyfeng/DPLP`

6.2.3 Sentiment scores

Sentiment analysis or opinion mining is an area of natural language processing. Its goal is to analyse people's opinions, sentiments and emotions via their text using computational algorithms. Sentiment lexicon-based approaches use positively and negatively labelled word lexicons and determine the text's sentiment score using the word occurrences [Liu10]. *Semantic orientation (polarity-based) lexicons* uses only positive and negative scores, but many applications can benefit from weighing the words' scores ('horrible' is a worse word than 'bad'). *Sentiment intensity (valence-based) lexicons* aims to resolve this issue by giving intensity scores to the words [WWH04]. Furthermore, context-awareness is another important aspect of field [AWM09]. For example, the word 'bad' has a negative meaning, however, using in the phrase 'not bad', it means something positive.

This experiment uses a module called VADER[5] to determine sentiment scores for both propositions, the original sentences and the full text. VADER (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media [HG14].

6.2.4 Word and sentence embedding vectors

Section 3.3 introduced the word embedding vectors as word representations. ArguE uses a model where the word embedding of the two propositions is fed into an LSTM network [Mil17]. This project included the same LTSM network using both the propositions and the original sentences. Both word2vec [MSC+13] and BERT [DCLT18] embedding vectors can be used similarly.

However, BERT can be used as sentence-level embedding using the [CLS] starting token's vectors to represent the whole sentence [DCLT18]. This way, the LTSM network is not necessary and the BERT embedding can be used as features of a simple feed-forward network the same way as the other features.

6.3 Classification problems

This experiment includes two classification problems.

The *proposition type classification* aims to identify the type of proposition in the argument. The Student Essay Corpus v2 uses *premise*, *claim* and *conclusion* types.

[5] Available at `https://github.com/cjhutto/vaderSentiment`

The conclusion type is the final claim of the essay, therefore each essay contains only one conclusion. This makes the dataset heavily unbalanced regarding the conclusion type.

The *argument relation classification* aims to identify the relations between proposition pairs. The simplest task of this classification is to choose between *related* and *non-related* types. Using different *support* and *attack* classes, the problem becomes harder. It can be a one-step three-way classification or a two-steps chain (first identify the relation, later the support-attack type). The connection between the propositions can be directed or non-directed (*A* supports *B* or *B* supports *A*).

6.4 Classification models

The models of this experiment are built-in Keras version 2.2.4 using Tensorflow [C+15]. Keras is a high-level neural networks API, developed with a focus on enabling fast experimentation.

6.4.1 ArguE based model using token- and sentence-level features

This model based on the one introduced by Milz using LSTM networks of token-level (local) features connected with sentence-level (global) features in a fully connected network. Figure 6.1 describes the original ArguE classifier.

The ArguE classifier uses LSTM (Long Short-Term Memory) recurrent layers to capture and feed token level information into the network. The word2vec, PoS and token-level BERT vectors are the inputs of the LSTM vectors for both proposition respectively. Then, the output of the two LSTM networks are concatenated with the shared features and a standard fully connected layer merges the information. The output layer uses softmax activation to serve the probability format output.

Because of the different tokenizer used in BERT and NLTK, the tokens of a sentence are different for the token-level BERT and the word2vec, PoS tokens. Therefore, the models of this experiment are not using the three of them simultaneously. However, global features can be added to the original ArguE architecture.

The original ArguE classifier uses adam optimizer [KB14] with binary crossentropy loss function.

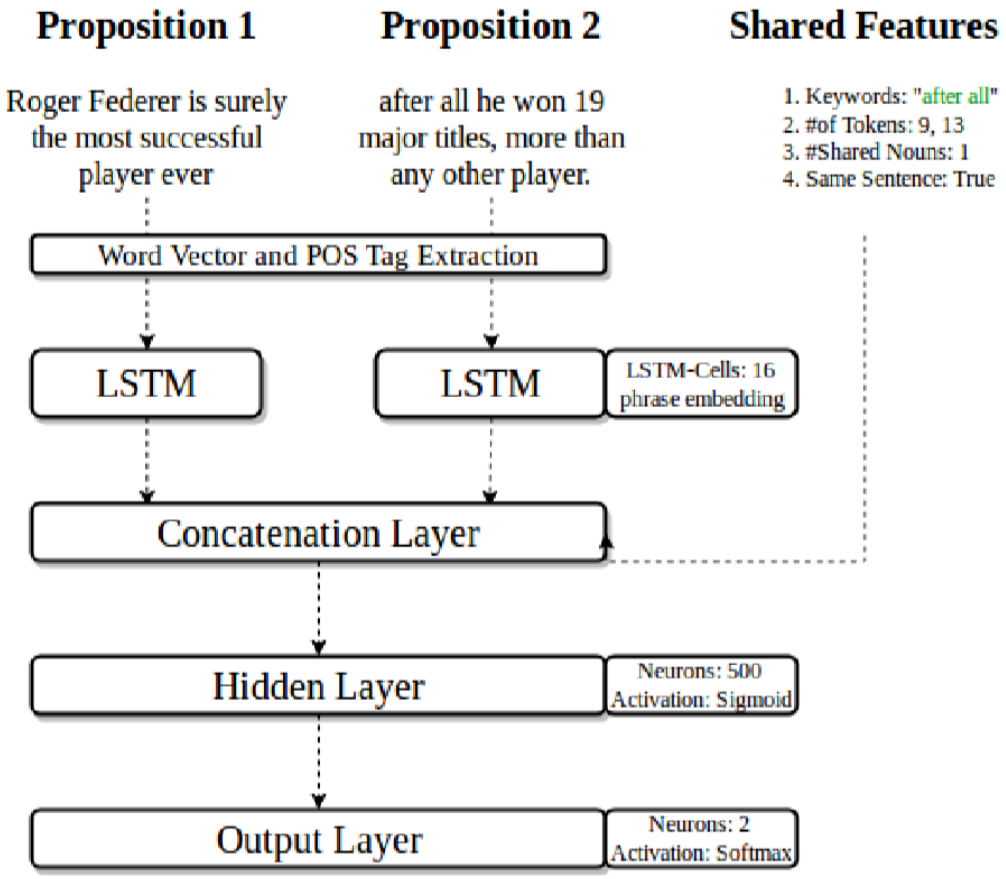


Figure 6.1: Original figure of ArguE's classifier architecture [Mil17, page 61]

6.4.2 Feed-Forward model using only sentence-level features

This model uses BERT sentence-level embedding vectors replacing the previous model's token-level representations (word2vec embedding and PoS tags). Therefore, this model uses only feed-forward hidden layers and no LSTM layers.

To balance the weights of the sentence-level embedding vectors (768 input features) with the shared global features (at most 43 features), the architecture includes hidden layers of fully connected neurons before the concatenation. Figure 6.2 layer level 1-2.

For additional information, not only the propositions' BERT sentence-level embedding vectors are inputs of the network, but the original sentences of the propositions are stored as well. The proposition's layers are connected with the corresponding sentence's layers before feeding into the merger concatenation layer (layer level 3). Additional layers are included to merge the information of the proposition's features and original sentence's features before concatenating with the other proposition (layer level 4-5).

After concatenating the proposition specific layers with the global shared features, the architecture is the same as the ArguE model's (layer level 6+).

A Dropout layer is available after every hidden layer to prevent overfitting. Unlike ArguE, this architecture uses ReLU activation function.

A stored model in .h5 format requires less than 20 MB.

6.4.3 Transfer learning using BERT

This model uses a pre-trained BERT model fine-tuned to fit the relation detection task.

The original, pre-trained BERT model was trained in two way. To guess [MASK] masked words based on the context and to predict follow-up sentences. Both method's training data can be generated from large text corpora with self-supervised learning. The preprocessing algorithm can mask out words or mix sentences to generate the data.

In this section, the BERT sentence prediction is used. The original method expects a sentence *A* and a sentence *B* and gives a percentage of *P* that tells us, how likely that *B* follows *A*.

Changing the method's parameters with Transfer Learning to fit the relation detection task, the method's modified task is the following: with input proposition *A* and proposition *B* what percentage *P* is the chance that *A* and *B* are related.

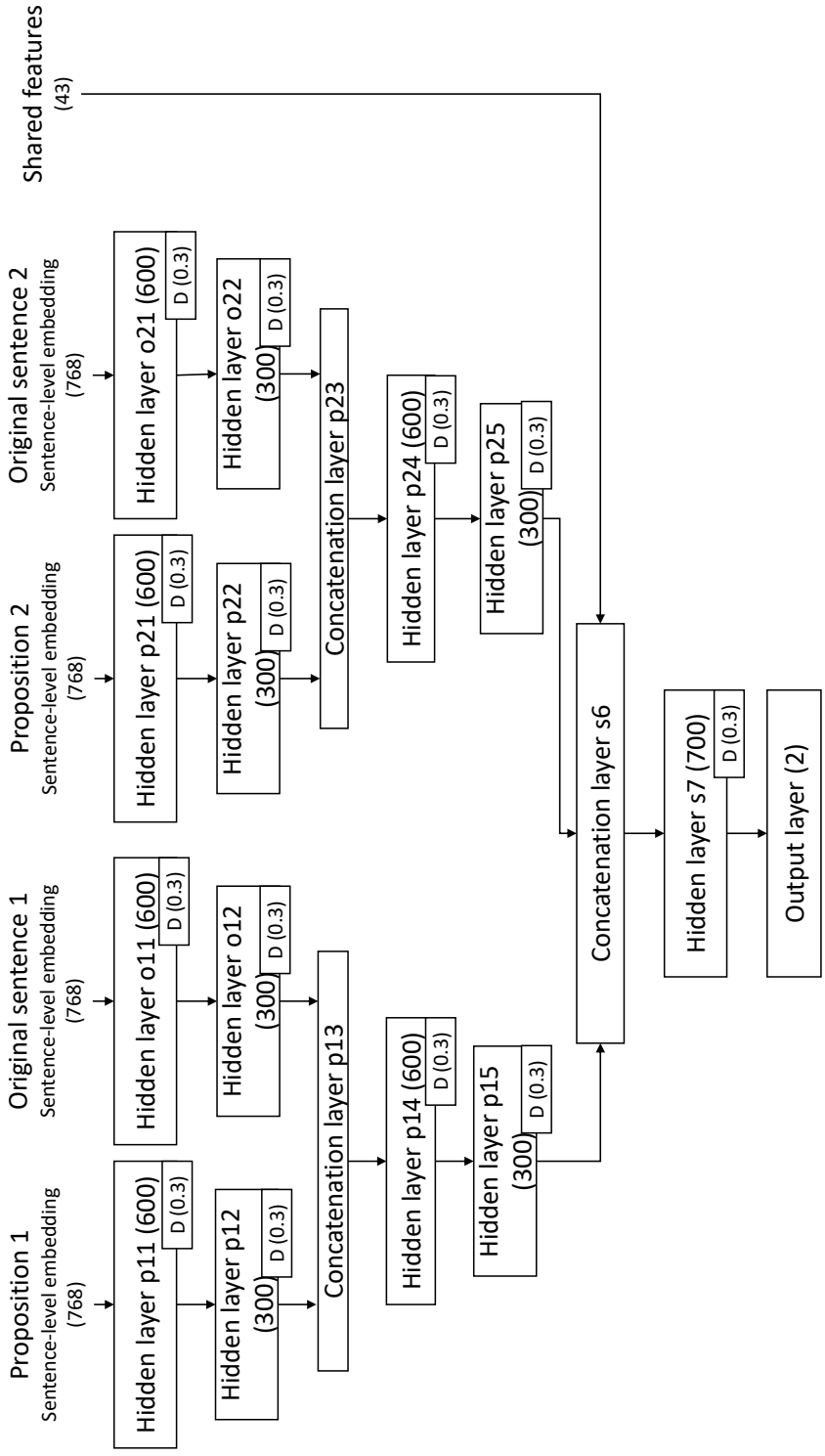


Figure 6.2: Sentence-level network architecture with unit size
Dropout rate: 0.3, hidden layers per step: 2, hidden layer decrease rate: 0.5

For this experiment, an online available example[6] is used without major modifications, therefore the experiment's codes are not included in the project's source code files.

6.5 Combining corpora using the unified parser

Each unique corpus contains only a handful of arguments, therefore combining the corpora to generate a larger dataset using the Unified Corpus Format is a reasonable goal. However, the differences between the corpora make some features unusable. For example, the AIF-DB collection does not store the original text, therefore, the distance difference features cannot be used in a general model. These features are noted with * character at the Table 6.2.

6.6 Description of the tool

The downloadable tool available on GitHub [7] contains a trainer method and a predictor method.

The trainer needs a data collection in the Unified Corpus Format, with possible additional RST files. It can be trained to predict proposition types or relations.

The trainer's data preprocessing is prepared for both the Student Essay Corpus and the AIF-DB Collection files using the Unified Corpus Format. It can be trained for either with every available feature or only with generable ones. In the second version, the RST files are not included as they are produced with a separate parser. The default mode uses the features described in the previous sections. The trainer stores the Keras model in .h5 format.

The predictor is a pipeline that constructs Unified Corpus Format from an argumentation essay. It uses a trained proposition type classifier and a relation detection classifier. See Figure 6.3 for the summary of the pipeline.

The required Argumentation Mining steps to reconstruct a Unified Format file from the full argumentation text are 1) proposition identification, 2) proposition type classification, 3) relation detection.

[6] https://colab.research.google.com/GitHub/google-research/bert/blob/master/predicting_movie_reviews_with_bert_on_tf_hub.ipynb

[7] https://github.com/negedng/argument_BERT

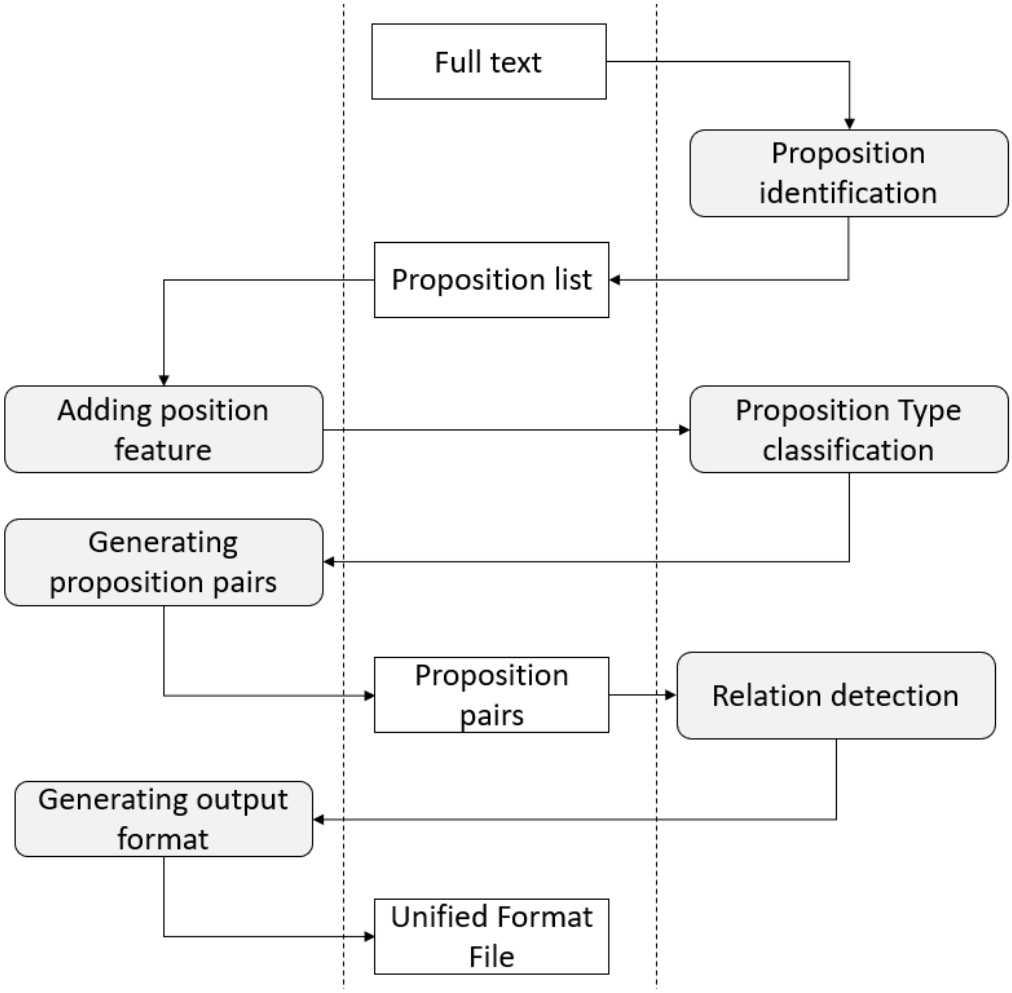


Figure 6.3: Argumentor tool pipeline
Right side: Argumentation Mining tasks, Left side: additional tasks

As the proposition identification is not part of the current experiment, in this part we have to make a concession. Instead of using a proper proposition identification method, here the tool uses a simple sentence segmentation. This means that in some cases, the correct argument propositions are in the same sentence for the tool. Also, non-argumentative sentences are included in the argument structure. Future work is required to solve this task of the pipeline.

For example, in the *essay01* from the Student Essays Corpus, the sentence *"Hence it is always said that competition makes the society more effective."* originally has a proposition *"competition makes the society more effective"*. Sometimes, the difference is only a few words but sometimes there are more propositions in the same sentence.

The proposition type classification is a three-class classification, described in the previous section. The classification uses only the current proposition for the decision, thus it is possible to have multiple *conclusion* in the same argumentation although the Student Essay Corpus is based on a model that has only one conclusion in every argumentation.

The Unified Corpus Format's ADU element stores the most probable type of the proposition. For additional information, the prediction's softmax score (the probability of the class) is stored in the confidence field. An example of an ADU element is:

```
<ADU type="claim" confidence="0.51830024"/>
```

For the relation detection, we have two options. Using a *related*, *non-related* classifier and a *support*, *attack* classifier as a chain or a three-way *non-related*, *support* or *attack* classifier. This tool uses the second option.

The tool pairs every proposition with every other proposition, making a complete graph of propositions and stores the result of the related pairs. However, the original data stores directed connections in the head of the directed edge, this tool uses non-directed connections, therefore, it cannot store the relation in the head proposition. Instead, the tool uses the first come first served rule, the relation is stored in the proposition that is earlier in the proposition list, thus, in the text.

The original Student Essay Corpus files have relations with special IDs. Also, the Unified Corpus Format makes it possible to add special argument scheme related connections, e.g. Expert Opinion support. This tool cannot reproduce these information, therefore, every connection is either a *Default Conflict* (attack) or a *Default Inference* (support). The ID of the relations is based on the IDs of the propositions.

The tool uses only the information extracted from the proposition pair and no argumentation level information. Even though the Student Essay Corpus arguments are

built on a tree model, this tool can produce arguments that cannot be represented as a tree. A tree model ensures that every proposition has at least one connection and there are no circles in the tree.

An example of a relation element stored in the proposition T4is stated below using the Unified Corpus Format. The relation detection model's maximum probability class is stored as the relation type and the probability is the confidence. Non-related pair relations are not stored.

```
<Relation relationID="RT4-T12" type="Default Inference" confidence=
    "0.4926278" typeBinary="0" partnerID="T12"/>
```

Appendix B is an example of a generated Unified Corpus Format file of the Student Essay Corpus *essay01*. The original annotation can be found in Appendix A.

Chapter 7

Evaluation

The experiments are tested in a Google Colab environment using the Python 3 Google Compute Engine backend with 12.72 GB RAM and 48.97 GB disk capacity. See more about Google Colab Research at `https://colab.research.google.com`.

For each test, measurements were taken three times, average and the standard deviation is included in the result tables. To calculate average scores for the metrics in the multi-class tasks, the macro average is used. $Avg_\alpha = \frac{1}{N}\sum_i^N \alpha_i$. This does not apply to the Transfer Learning models due to its length (unfortunately, Google Colab is not suited for long training).

The used metrics in the measurements: precision (P) is the relevant, correct information among the identified instances $P = \frac{TP}{TP+FP}$, recall (R) is the found instances of the relevant information among all the instances supposed to be identified $R = \frac{TP}{TP+FN}$. F1-score is the harmonic mean of the precision and the recall $F_1 = 2\frac{P \cdot R}{P+R}$.

7.1 Relation detection classifiers

The first test is a two-class classification problem. The task is to determine whether the proposition pair is related to each other or not. Support and attack relations are both included. The direction of the proposition is not important. To balance the dataset, the number of non-related proposition pairs is reduced to the number of related proposition pairs, unless it is noted otherwise. Table 7.1 summarises the distribution of the Student Essays Corpus database [SG17].

The balanced dataset contains the same amount of related proposition pairs as it contains unrelated pairs. The total number of related pairs is 5696. The related / non-related rate of the unbalanced dataset is 0.10. However, the dataset is balanced to

	Balanced		Unbalanced	
	Train set	Test set	Train set	Test set
Sum	5126	570	25045	2783
Non-related	2538	310	22491	2489
Related	2588	260	2554	294
Support	2306	226		
Attack	282	34		

Table 7.1: Distribution of the relation database in the Student Essay Corpus.

Name	P	R	F1	F1 Non-related	F1 Related
[SG17] baseline majority	0.42	0.5	0.46	0.91	0
[SG17] baseline heuristic	0.66	0.66	0.66	0.89	0.44
[SG17] human upper bound	-	-	0.85	0.95	0.75
[SG17] SVM all features	0.76	0.71	0.73	0.92	0.54
[Mil17] ArguE	0.79	0.63	0.68	0.75	0.27
ArguE restored, balanced	0.63	0.63	0.63	0.64	0.61
SL all features, balanced	0.76	0.76	0.76	0.78	0.75
SL all features, unbalanced	**0.80**	0.73	0.76	**0.95**	0.56
TL with BERT	0.78	**0.77**	**0.77**	0.76	**0.77**

Table 7.2: Relation detection models

have the same amount of related and non-related pairs, the support-attack ratio is still unbalanced. Due to the lack of enough attack relations, there was no attack-support balanced dataset. The attack-support ratio in the balanced dataset is 0.12.

Table 7.2 contains a comparison of the related works tested in this dataset with the best performing models of this experiment.

Stab and Gurevych used two baseline function for the tests and a human upper bound by averaging the results of three annotators on their test data [SG17]. This table includes the same baselines.

The *majority baseline* labels each instance with the majority class. As their original training set had 17.5% related pairs, the majority baseline signals non-related case every time.

The *heuristic baseline* is motivated by the common structure of persuasive essays [Whi09, Per10]. The heuristic baseline for relation detection signals relation label if the target proposition is in the first paragraph of the essay.

The first model is their best performing SVM model with all of their features included [SG17].

The second model is Milz's ArguE classifier, using the architecture described in

Name	P	R	F1	F1 Support	F1 Attack
[SG17] baseline majority	0.45	0.5	0.48	0.95	0
[SG17] baseline heuristic	0.51	0.53	0.52	0.77	0.17
[SG17] human upper bound	-	-	0.84	0.96	0.70
[SG17] SVM all features	**0.71**	**0.69**	**0.70**	**0.95**	**0.46**
SL all features	0.61	0.58	0.59	0.91	0.27

Table 7.3: Support-Attack identification

Section 6.4.1. As the source code of the architecture is available, the was implemented in this experiment as well and the results are included in the table.

The table contains two sentence-level feed-forward neural networks described in this paper. The first one is trained and tested on the balanced dataset and the second one is on the unbalanced dataset. The balanced ratio of Stab and Gurevych's original test set is 0.165 while the balance ratios in these two cases are 0.5 and 0.102 respectively.

Finally, the Transfer Learning model based on the pre-trained BERT performed in a different environment. It used a different balanced distribution of the dataset.

The unbalanced sentence-level model outperformed both the SVM model and the ArguE model with a 0.76 average F1 score and a human-level 0.95 F1 score on non-related proposition pairs while keeping a fine 0.56 F1 score on the related pairs.

The balanced sentence-level model and the transfer learning model both performed a balanced $0.75 - 0.78$ F1 score on both the related and the non-related pairs. This result outperforms the human upper bound of Stab and Gurevych's experiment on related pairs. However, the test sets are different here, therefore the comparison may not be relevant.

The performance of the restored ArguE model shows differences from the original one but it comes from the difference between the training and testing data balance.

Table 7.3 contains results of a support-attack identification. In this case, only the related pairs of the dataset are included in both the training and the testing phase. Therefore, this experiment is unbalanced as well as Stab and Gurevych's original one. Their attack-support ratio for the test set were 0.08 and here, it was 0.12. The best performing sentence level model is compared to its SVM model.

The model performed with an average F1 score of 0.59, falling behind the SVM model by 10% but above the baselines. It performed well on support relations with an F1 score of 0.91, just below the human upper bound and the SVM performance. The attack relation identification was not satisfactory with a 0.27 F1 score, above the baseline but falling behind the SVM model.

Precision		Recall		F1-score							
				Average		Support		Attack		Non-related	
Avg	SD	Avg	SD	Avg	SD	Avg	SD	Avg	SD	Avg	SD
0.55	0.04	0.53	0.03	0.53	0.04	0.66	0.07	0.17	0.10	0.76	0.02

Table 7.4: Non-related, support, attack identification

Name	Time	Epochs	Precision		Recall		F1-score					
							Average		Non-related		Related	
	Avg(s)	Avg	Avg	SD	Avg	SD	Avg	SD	Avg	SD	Avg	SD
AllFeat	134.43	160.00	**0.76**	0.01	**0.76**	0.01	**0.76**	0.01	**0.77**	0.01	**0.75**	0.01
NOrigBert	**59.78**	155.67	0.73	0.01	0.73	0.01	0.73	0.01	0.74	0.01	0.72	0.02
NFullText	101.27	113.33	0.68	0.01	0.68	0.01	0.68	0.01	0.70	0.01	0.66	0.01
OnlyBert	107.76	119.33	0.64	0.01	0.64	0.01	0.64	0.01	0.63	0.02	0.65	0.02
ArguE	208.25	101.33	0.63	0.02	0.63	0.02	0.63	0.02	0.64	0.02	0.61	0.02
ArgueAll	193.01	92.33	0.66	0.01	0.66	0.01	0.66	0.01	0.68	0.02	0.65	0.01
ArgueBert	282.51	**71.00**	0.64	0.01	0.64	0.01	0.64	0.01	0.66	0.02	0.62	0.01

Table 7.5: Relation detection performance with different features (Sentence Level model and ArguE based model)

Table 7.4 includes performances of a multi-class classification where the support, attack and non-related pairs are together and the task is to select the right class from the three. The model performs with a 0.53 average F1 score. It works the best on the non-related pairs with an average of 0.76 F1 score, following on support relations with an average of 0.66 F1 score and it performs the worst on attack relations with an average F1 score of 0.17. However, the standard deviation of the attack relation performance is high because it varies between 0.02 and 0.25.

7.2 Feature performance

In this experiment, different feature lists and models are tested. All test are measured on the balanced related non-related dataset described in the previous section. The models are the followings:

The model named *AllFeat* is the one described in Section 6.4.2. It contains all the features described in Section 6.2. It has BERT sentence-level embedding vectors for both proposition and their original sentences and all the position related, ArguE based, sentiment and RST features. It took a little bit over 2 minutes to train a network and it had a 0.76 macro-average F1 score.

The model named *NOriginBert* has no BERT sentence-level embedding vectors for

the original sentences, only for the prepositions. This has sped up the training from the average 134s to an average of 60s, making the training more than twice as fast as the previous model's. On the other hand, this reduced the performance by 0.03 for every metrics, however, it had the same average F1 score, as the SVM from Stab and Gurevych.

The next model is called *NFullText*. It has no full-text dependent features like sentence position difference or shared words with the full text. See 6.2 for the full list. This is important because a lot of available corpora lacks the original full text of the argumentation and only has the annotated propositions. Also, the tool described in Section 6.6 uses only the full-text independent features. It had a 0.68 average F1 score with a 0.70 F1 score for non-related pairs and 0.66 F1 score for related pairs making it just above the heuristic baseline. Interestingly, the heuristic baseline is calculated from the full text and had a 0.66 average F1 score.

The final sentence-level feedforward network-based model uses only BERT embedding vectors (*OnlyBert*). It uses zero additional global shared features, only the embedding vectors of the two propositions and original sentences. The model felt just below the heuristic baseline with a 0.64 average F1 score.

The ArguE named models are based on Milz's ArguE model [Mil17]. They have a token-level feature-based LSTM network linked together with the global shared features. The first model named *ArguE* is the same as it is described in the original paper and Section 6.4.1. It had a 0.63 average F1 score making it worse than the least well-performing one of sentence-level models.

The next ArguE based model is called *ArgueAll*. It uses the ArguE defined token-level features with all the available sentence-level global features (not the sentence-level BERT features). This model achieved the heuristic baseline's F1 score with 0.66, however, still far from the sentence-level models.

Lastly, *ArgueBert* uses token-level BERT embedding vectors. Because of the difference between the tokenizers used in the two methods, it cannot be used with the word2vec and PoS token-level features. This model uses all the shared features as the previous one and performed slightly worse than the ArgueAll model with an F1 average score of 0.64.

	Unbalanced		Balanced	
	Train set	Test set	Train set	Test set
Sum	1241	311	2206	736
Premise (Pr)	827	206	749	251
Claim (Cl)	343	86	898	292
Conclusion (MC)	71	19	559	193

Table 7.6: Distribution of the proposition type database
The Student Essay Corpus and a balanced Unified Corpus based on the Student Essays and Araucaria corpora.

Name	P	R	F1	F1 Pr	F1 Cl	F1 MC
[SG17] baseline majority	0.21	0.33	0.26	0.77	0	0
[SG17] baseline heuristic	0.72	0.72	0.72	0.87	0.56	0.74
[SG17] human upper bound	-	-	0.87	0.92	0.75	0.93
[SG17] SVM all features	0.77	0.77	0.77	0.86	0.59	0.87
SL all features	0.58	0.53	0.55	0.78	0.45	0.42
SL all, unified Corpus	0.53	0.53	0.53	0.49	0.58	0.54

Table 7.7: Premise, Claim and Conclusion (Major Claim) classification models

7.3 Proposition type classifier

Table 7.6 contains informations about the premise-claim distribution of the Student Essay dataset. This dataset is heavily unbalanced. Because of the Unified Corpus Format, it is possible to merge corpora to build a more balanced dataset. The second half of the table shows a balanced unified corpus using propositions from the Student Essays Corpus and the Araucaria corpus as well.

Table 7.7 summarises the results of the premise-claim-conclusion (major claim) classification experiment. The heuristic baseline for this experiment labels the first argument proposition of each paragraph as a claim and everything else as a premise. It defines the last argument part of the introduction and the first argument part of the conclusion as a major claim.

The sentence-level feedforward network's performance was below the heuristic baseline. It only achieved a 0.55 average F1 score, with above 0.75 for only the premises.

Using the Unified Corpus, the next model was trained on a balanced dataset. It made a better balance between F1 scores of the three classes (0.49 for premises, 0.58 for claims and 0.54 for major claims), however, the average F1 score of 0.53 did not improve compared to the previous model.

	Student Essays	Araucaria Filtered
Sum	5696	3044
Non-related	2848	1512
Related	2848	1532

Table 7.8: Comparison of the Sudent Essays Corpus and the Araucaria Corpus Data samples with missing features are filtered. Datasets are balanced.

Name	P	R	F1	F1 Non-related	F1 Related
SL merged	0.72	0.72	0.72	0.73	0.72
TL merged	0.70	0.65	0.68	0.68	0.67
SL general A	0.60	0.60	0.60	0.61	0.60
SL general B	0.63	0.62	0.61	0.56	0.66

Table 7.9: Relation detection models using unified corpus

Regarding these results, it is important to note that the Araucaria corpus does not always contain the full text of the argumentation. Sometimes, only relevant sentences or only propositions are included.

7.4 Unified corpus

In the previous section, one use of the Unified Corpus Format was described. Among providing additional data samples to achieve a balanced dataset, in this section, two more possibilities are described. Here, the Student Essays Corpus and the Araucaria Corpus are used together for the relation detection problem.

In the first experiment, the data from both corpora are merged to build a larger corpus. Due to missing features, the majority of the Araucaria corpus cannot be used in full potential. As a compromise, the RST features are not used in this experiment and all the Araucaria proposition pairs with missing features are removed. Originally, the Student Essays Corpus contains 27828 proposition pairs, after balancing, it becomes 5696 pairs. In the meantime, the Araucaria corpus contains 20864 proposition pairs, after the filtering and balancing, only 3044 remains.

The model named *SL merged* in Table 7.9 is trained on a merged unified corpus of the datasets described above. With the 0.72 average F1 score, it has a performance comparable to the best methods trained on the Students Essay Corpus.

The model *TL merged* is a transfer learning model of BERT using a different unified corpus. It performed slightly worse than the SL model with an F1 score of 0.68.

The second experiment aimed to investigate the generality of the models, corpora. The model named *SL general A* is trained on the Student Essay Corpus but tested on the Araucaria corpus. The model named *SL general B* is trained[1] on the Araucaria corpus and tested on the Student Essays Corpus.

Generalisation in both ways performed around 0.61 F1 score. Compared to the heuristic baseline of the relation detection models in Table 7.2, this is 0.05 worse, however, it performs better on related pairs ($0.66, 0.60$ compared to 0.44).

7.5 Tool performance example

This section compares the original annotation of the Student Essay Corpus *essay01* (see full annotation: Appendix A) with the predicted annotation using the previously described tool (see full annotation: Appendix B).

It is important to note that the proposition type classification and relation detection samples required to build the annotation file might be in the training dataset of the corresponding models.

Figure 7.1 represents a relation graph of the annotation files. The boxes (nodes) are the propositions of the argument and the edges are the connections between the propositions. The text of every box contains the proposition's ID in the two argumentation annotation file, first the original than the predicted. The original annotation uses directed connections, therefore the corresponding edges in the figure are arrows pointing to the supported (attacked) proposition's direction. The edges of the prediction are non-directed, thus the edges are just lines. The nodes are organised in a hierarchical view: the conclusion is on the top (T2/T4) and every other proposition is under it's supported the (attacked) proposition.

In this example, the 'sentence as a proposition' concept generates fits relatively well. There are no sentences with multiple propositions and only three additional sentences. One of them is the title of the essay (-/T0).

The proposition type classification correctly identifies ten out of fourteen propositions. It classifies the conclusion (major claim) as a claim. There is three other mislabelled proposition. The non-proposition sentence is classified as claims. In Figure 7.1, the correctly labelled propositions are shown in green colour and the mislabelled propositions in red.

The relation detection method's performance is less promising. In Figure 7.1, the

[1] This training used rmsprop optimizer instead of adam

correctly detected relations are green. There are four of them: T1/T1-T2/T4, T4/T5-T2/T4, T3/T2-T1/T1 and T10/T11-T13/T15. If we do not count the non-proposition sentences' edges, there are nine missing connection labelled with blue and nine additional connection coloured with red. The number of correctly non-connected pairs are $\binom{14}{2} - edges = 91 - (4+9+9) = 69$. The F1 score of this related / non-related classification performance is 0.31.

All the relations in the prediction annotation file are marked as Default Inference (support). There are two attacks in the original file: T1-T2 and T8-T2.

Interestingly, the title sentence is connected to every other sentence except 3.

Observing the figure, we can notice that there are relatively more edges on the left side of the image. Because the propositions are organised to be sorted from left to right, it means that there is a correlation between the propositions position in the text and the likeliness of the predicted edges. The non-proposition sentence (-/T12) seems to be an exception to this rule.

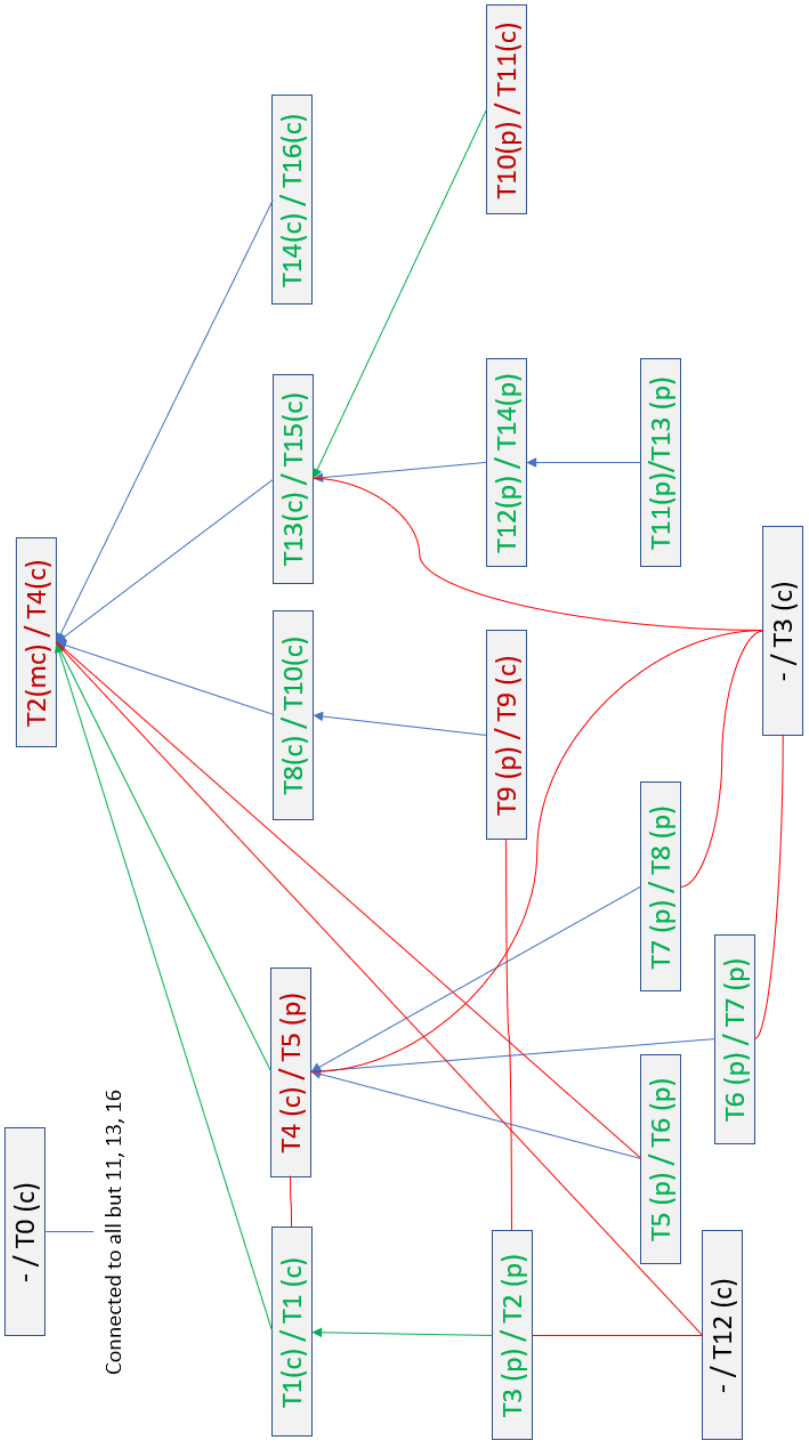


Figure 7.1: Relation graph of the original and the predicted annotation
Proposition types: correct (green): 10, incorrect (red): 4
Relations: TP (green): 4, FN (blue): 9, FP (red): 9, TN (white): 69

Chapter 8

Analysis

This experiment provided a new algorithm for argumentation mining tasks. The results of the relation detection classification problem demonstrated that this is an improvement compared to previous studies [SG17, Mil17]. However, the performance observed in the argument structure type classification is far below those achieved by Stab and Gurevych [SG17].

8.1 Data balance

Regarding the SVM model of Stab and Gurevych [SG17] and the ArguE model of Milz [Mil17], we can see that the F1 score of the different classes may vary (see Table 7.2, 7.3 and 7.7). Investigating the corpus data distribution, we can conclude that there is a correlation between the data balance and the performance of the model. It is a reasonable explanation to assume that the model performs worse on classes that have a fewer number of samples in the dataset because it is trained on the relatively less than the ones supported with more samples.

The performance of this experiment's models confirms that using a balanced dataset affects the scores in a way that the more the dataset is balanced, the more the consistency of the model's performance on different classes. Both the relation detection experiment using the balanced dataset of the Student Essays Corpus [SG17], and the proposition type classification using the balanced Unified Corpus supports this conclusion.

However, both experiments showed that regarding the overall performance of the model, using a balanced dataset may not be a better solution. The balanced and the unbalanced dataset in the relation detection experiment (see 7.2) scored the same average

F1 score of 0.76.

Therefore, it is an open question to decide the balance of the dataset to train the networks on. If one wants a model that is equally reliable on every class of the data, one must choose a balanced dataset. If the user expects that the real-life application may have the same data distribution as the dataset the model is trained on, it is possible to train on an unbalanced dataset and expect equally good results.

8.2 Supporting features

This combination of findings provides some support for the conceptual premise that the location of an argument proposition has a relatively large impact when it comes to determining the proposition's role in a persuasive essay. The models without the supporting features of the full textual information, such as the position of the proposition, the distance between the proposition pairs, has just 0.02 better F1 score than the heuristic baseline depending only on the propositions' position in the text.

It can, therefore, be assumed that the importance of the full textual information is necessary to store for structured argumentation such as persuasive essays to create a well-performing argumentation mining model.

The results of the relation detection experiment summarised in Table 7.2 shows that embedding vectors such as BERT are important tools to build argumentation mining models. The model using only the BERT sentence-level embedding vectors of the proposition pairs achieved a 0.64 F1 score, 0.18 above the majority baseline and only 0.02 below the heuristic baseline. The Transfer Learning model using BERT shows promising results as well. Therefore, we can conclude that transformers are the next step to understand argumentation mining.

The experiments showed that the sentence-level embedding vectors are enough for a cross-sentence task (relation detection), however, they are not as satisfying for a one-sentence task (proposition type classification) as the token-level features of previous works.

8.3 General Argumentation Mining models

In the previous section, the importance of the positions of the argument parts in a persuasive essay is concluded. However, the Araucaria database consists rather of unstructured argumentation from web forums [RPR$^+$08].

To achieve a general argumentation mining discipline, we have to build models that perform as well on a structured argumentation as on an unstructured argumentation. Therefore, the experiment summarised in Table 7.9 was developed to investigate how well this current argumentation mining model can perform in a general task.

To observe these results, the performance of the *SL merged* model can be viewed as an upper bound. It is the performance of the feed-forward model containing Sentence-Level embedding vectors that has information about both databases in the training phase of the experiment. In the meantime, *SL general A* and *SL general B* models contain information of a single data corpus each. The results of the experiment show that there is no significant difference between the performance of *A* and *B* (both achieved near an average F1 score of 0.60). However, the generalisation of the model is far from the ideal as the ideal performance of the merged model achieved a 0.72 average F1 score, 0.12 above the separately trained models.

Chapter 9

Future work

There are still many unanswered questions about the features of a good argumentation mining model, but these experiments showed that further argumentation mining corpus building work is required to include the whole text of the argumentation.

Despite these promising results of this experiment, general argumentation mining remains only a goal. However, this work showed that using a unified corpus format, the different argumentation mining corpora can be used together, but the hope for a model that can perform well in future corpora is still unanswered.

The experiment showed that the recent development of pre-trained language models can impact the field of argumentation mining as well as any other field of natural language processing. Despite the great results, the relation detection problem is only one of the many subtasks of argumentation mining, therefore there is abundant room for further progress in determining whether the transfer learning is a viable method to solve argumentation mining problems.

There are multiple possible next steps regarding the tool. First of all, an intelligent argument proposition detector is needed for the pipeline. Secondly, the Argument Theories used to built the annotations have restrictions. Some of these restrictions in the theory used in the building of the Student Essay Corpus: 1) there are only one conclusion proposition, 2) the chain of propositions is: conclusion-claim-premise-premise..., 3) the relation graph is a tree. Further research should be undertaken to investigate the possible implementation of these restrictions.

To develop a full picture of argumentation mining, additional studies will be needed that focus on the high-level argumentation structures. The promising results of these classification problems make us hopeful that general argumentation mining is achievable, but several more complex questions remain unanswered at present.

Chapter 10

Conclusion

This study set out to investigate the possibilities of argumentation mining and to analyse the effects of the current development of pre-trained language models regarding argumentation mining problems. The essay has discussed the reasons for a unified corpus format and demonstrated the use of an existing unified corpus. This project was undertaken to design argument mining classifiers and evaluate their performance compared to the existing state-of-the-art.

This study has shown that the full textual context of the arguments can improve the performance of the argumentation mining model (0.08 F1 score improvement). This research has also confirmed that transfer learning using pre-trained language models can be effective in argumentation mining tasks (0.77 F1 score in the argument relation classification problem). The relevance of a unified corpus format regarding a general argumentation mining model is supported by the current findings. The model trained on a merged unified corpus performed almost as good as the one trained on the specific task (0.72 F1 score compared to 0.76), suggesting that the model described in the experiment is effective to solve argumentation mining problems using merged unified corpora.

Being limited to argument proposition type classification and argument relation classification, this study lacks the necessary steps of a full argumentation mining model. The major limitation to achieving a full pipeline is that this experiment's models are relying on a third party argument component identification. Also, the experiment is limited to low-level argumentation tasks.

Despite the statement of the importance of the full text, the tool included in this experiment only rely on the argument proposition features.

The study was limited by the absence of a proper machine built to test large language model fine-tuning, therefore the transfer learning investigations are limited to only a few tests and future research is required for a better understanding of the possibilities and limitations of the technique regarding argumentation mining.

The information provided by this study can be used to develop more general, more accurate argumentation mining models.

Bibliography

[ABV18] Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, 2018.

[ACK+17] Yamen Ajjour, Wei-Fan Chen, Johannes Kiesel, Henning Wachsmuth, and Benno Stein. Unit segmentation of argumentative texts. In *Proceedings of the 4th Workshop on Argument Mining*, pages 118–128, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[AEAW16] Rob Abbott, Brian Ecker, Pranav Anand, and Marilyn A Walker. Internet argument corpus 2.0: An sql schema for dialogic social media and the corpora to go with it. In *LREC*, 2016.

[AWM09] Cem Akkaya, Janyce Wiebe, and Rada Mihalcea. Subjectivity word sense disambiguation. In *Proceedings of the 2009 conference on empirical methods in natural language processing*, pages 190–199, 2009.

[BCDG09] Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. The fifth pascal recognizing textual entailment challenge. In *TAC*, 2009.

[BM03] Jill Burstein and Daniel Marcu. A machine learning approach for identification thesis and conclusion statements in student essays. *Computers and the Humanities*, 37(4):455–467, 2003.

[BSF94] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[C+15] François Chollet et al. Keras. `https://keras.io`, 2015.

[CDA+17] Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*, 2017.

[Coh87] Robin Cohen. Analyzing the structure of argumentative discourse. *Computational linguistics*, 13(1-2):11–24, 1987.

[CT15] Lucas Carstens and Francesca Toni. Towards relation based argumentation mining. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 29–34, 2015.

[CVMBB14] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[CZZZ18] Zihan Chen, Hongbo Zhang, Xiaoji Zhang, and Leqi Zhao. Quora question pairs, 2018.

[DB05] William B Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

[DCLT18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[DDS+09] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[EDG17] Steffen Eger, Johannes Daxenberger, and Iryna Gurevych. Neural end-to-end learning for computational argumentation mining. *arXiv preprint arXiv:1704.06104*, 2017.

[Edw08] Damer T Edward. Attacking faulty reasoning: A practical guide to fallacy-fre e arguments. *Cengage Learning*, 209, 2008.

[FBCC+10] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. Building watson: An overview of the deepqa project. *AI magazine*, 31(3):59–79, 2010.

[FH11] Vanessa Wei Feng and Graeme Hirst. Classifying arguments by scheme. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 987–996, 2011.

[FKKK13] Eirini Florou, Stasinos Konstantopoulos, Antonis Koukourikos, and Pythagoras Karampiperis. Argument extraction for supporting public policy formulation. In *Proceedings of the 7th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, pages 49–54, 2013.

[Fre11] James B Freeman. *Dialectics and the macrostructure of arguments: A theory of argument structure*, volume 10. Walter de Gruyter, 2011.

[GLPK14] Theodosis Goudas, Christos Louizos, Georgios Petasis, and Vangelis Karkaletsis. Argument extraction from news, blogs, and social media. In *Hellenic Conference on Artificial Intelligence*, pages 287–299. Springer, 2014.

[Gov13] Trudy Govier. *A practical study of argument*. Cengage Learning, 2013.

[Had08] Jacques Hadamard. *Mémoire sur le problème d'analyse relatif à l'équilibre des plaques élastiques encastrées*, volume 33. Imprimerie nationale, 1908.

[HG14] Clayton J Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth international AAAI conference on weblogs and social media*, 2014.

[HR18] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

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

[HZC+17] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[JE14] Yangfeng Ji and Jacob Eisenstein. Representation learning for text-level discourse parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 13–24, 2014.

[KB14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[KBH16] Jeremy Kawahara, Aicha BenTaieb, and Ghassan Hamarneh. Deep features to classify skin lesions. In *2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI)*, pages 1397–1400. IEEE, 2016.

[Kel60] Henry J Kelley. Gradient theory of optimal flight paths. *Ars Journal*, 30(10):947–954, 1960.

[KP17] Jiman Kim and Chanjong Park. End-to-end ego lane estimation based on sequential transfer learning for self-driving cars. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 30–38, 2017.

[KZHS07] Namhee Kwon, Liang Zhou, Eduard Hovy, and Stuart W Shulman. Identifying and classifying subjective claims. In *Proceedings of the 8th annual international conference on Digital government research: bridging disciplines & domains*, pages 76–81. Digital Government Society of North America, 2007.

[LB02] Edward Loper and Steven Bird. Nltk: the natural language toolkit. *arXiv preprint cs/0205028*, 2002.

[LBH+14] Ran Levy, Yonatan Bilu, Daniel Hershcovich, Ehud Aharoni, and Noam Slonim. Context dependent claim detection. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1489–1500, 2014.

[Liu10] Bing Liu. Sentiment analysis and subjectivity. *Handbook of natural language processing*, 2(2010):627–666, 2010.

[LJN10] Annie Louis, Aravind Joshi, and Ani Nenkova. Discourse indicators for content selection in summarization. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 147–156. Association for Computational Linguistics, 2010.

[LKN09] Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. Recognizing implicit discourse relations in the penn discourse treebank. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 343–351, 2009.

[LNK14] Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. A pdtb-styled end-to-end discourse parser. *Natural Language Engineering*, 20(2):151–184, 2014.

[LR14] John Lawrence and Chris Reed. Aifdb corpora. In *COMMA*, pages 465–466, 2014.

[LR15] John Lawrence and Chris Reed. Combining argument mining techniques. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 127–136, 2015.

[LRM+14] John Lawrence, Chris Reed, Simon McAlister, Andrew Ravenscroft, Colin Allen, and David Bourget. Mining arguments from 19th century philosophical texts using topic based modelling. In *Proceedings of the First Workshop on Argumentation Mining*, pages 79–87, 2014.

[LT15] Marco Lippi and Paolo Torroni. Context-independent claim detection for argument mining. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[LT16] Marco Lippi and Paolo Torroni. Argumentation mining: State of the art and emerging trends. *ACM Transactions on Internet Technology (TOIT)*, 16(2):10, 2016.

[MBPR07] Marie-Francine Moens, Erik Boiy, Raquel Mochales Palau, and Chris Reed. Automatic detection of arguments in legal texts. In *Proceedings of the 11th international conference on Artificial intelligence and law*, pages 225–230. ACM, 2007.

[MBXS17] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6294–6305, 2017.

[MDP+11] Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 142–150. Association for Computational Linguistics, 2011.

[MFP+17] Afonso Menegola, Michel Fornaciali, Ramon Pires, Flávia Vasques Bittencourt, Sandra Avila, and Eduardo Valle. Knowledge transfer for melanoma screening with deep learning. In *2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)*, pages 297–300. IEEE, 2017.

[Mil17] Tobias Milz. Argue - an argumentation mining and modelling approach for classication of argumentative discourse units and structures. Master's thesis, University of Passau, 2017.

[Mis95] Connie A Missimer. *Good arguments: An introduction to critical thinking*. Prentice Hall, 1995.

[MLG+18] Ana I Maqueda, Antonio Loquercio, Guillermo Gallego, Narciso García, and Davide Scaramuzza. Event-based vision meets deep learning on steering prediction for self-driving cars. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5419–5427, 2018.

[Mon54] Beardsley C Monroe. *Practical Logic*. Prentice-Hall, 1954.

[MSC+13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[Nem18] Gergely Daniel Nemeth. Hyphenation using deep neural networks. Bachelor thesis, Budapest University of Technology and Economics, 2018.

[PABP17] Matthew E Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. Semi-supervised sequence tagging with bidirectional language models. *arXiv preprint arXiv:1705.00108*, 2017.

[Pel14] Andreas Peldszus. Towards segment-based recognition of argumentation structure in short texts. In *Proceedings of the First Workshop on Argumentation Mining*, pages 88–97, 2014.

[Per10] Vivien Perutz. A helpful guide to essay writing. *Student Services, Anglia Ruskin University*, 2010.

[Pet19] Georgios Petasis. Segmentation of argumentative texts with contextualised word representations. In *Proceedings of the 6th Workshop on Argument Mining*, pages 1–10, Florence, Italy, August 2019. Association for Computational Linguistics.

[PM09] Raquel Mochales Palau and Marie-Francine Moens. Argumentation mining: the detection, classification and structure of arguments in text. In *Proceedings of the 12th international conference on artificial intelligence and law*, pages 98–107. ACM, 2009.

[PN16] Isaac Persing and Vincent Ng. 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, 2016.

[PNI+18] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[PS15] Andreas Peldszus and Manfred Stede. Joint prediction in mst-style discourse parsing for argumentation mining. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 938–948, 2015.

[RDP+15] Ruty Rinott, Lena Dankin, Carlos Alzate Perez, Mitesh M Khapra, Ehud Aharoni, and Noam Slonim. Show me your evidence-an automatic method for context dependent evidence detection. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 440–450, 2015.

[RDS+15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[RPR+08] Chris Reed, Raquel Mochales Palau, Glenn Rowe, Marie-Francine Moens, et al. Language resources for studying argument. In *LREC*, 2008.

[RWB12] Niall Rooney, Hui Wang, and Fiona Browne. Applying kernel methods to argumentation mining. In *Twenty-Fifth International FLAIRS Conference*, 2012.

[RZLL16] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[Sch15] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[SEW15] Reid Swanson, Brian Ecker, and Marilyn Walker. Argument mining: Extracting arguments from online dialogue. In *Proceedings of the 16th annual meeting of the special interest group on discourse and dialogue*, pages 217–226, 2015.

[SG14a] Christian Stab and Iryna Gurevych. 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, 2014.

[SG14b] Christian Stab and Iryna Gurevych. Identifying argumentative discourse structures in persuasive essays. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 46–56, 2014.

[SG17] Christian Stab and Iryna Gurevych. Parsing argumentation structures in persuasive essays. *Computational Linguistics*, 43(3):619–659, 2017.

[SHK+14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[SKH19] Maximilian Spliethöver, Jonas Klaff, and Hendrik Heuer. Is it worth the attention? a comparative evaluation of attention layers for argument unit segmentation. In *Proceedings of the 6th Workshop on Argument Mining*, pages 74–82, Florence, Italy, August 2019. Association for Computational Linguistics.

[SKPK15] Christos Sardianos, Ioannis Manousos Katakis, Georgios Petasis, and Vangelis Karkaletsis. Argument extraction from news. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 56–66, 2015.

[Smi00] Robin Smith. Aristotles logic. *The Stanford Encyclopedia of Philosophy*, 2000.

[SP97] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[SPW+13] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[SVL14] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[SZ14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[Tay53] Wilson L Taylor. cloze procedure: A new tool for measuring readability. *Journalism Bulletin*, 30(4):415–433, 1953.

[TM06] Maite Taboada and William C Mann. Rhetorical structure theory: Looking back and moving ahead. *Discourse studies*, 8(3):423–459, 2006.

[VKD+19] Jacky Visser, Barbara Konat, Rory Duthie, Marcin Koszowy, Katarzyna Budzynska, and Chris Reed. Argumentation in the 2016 us presidential elections: annotated corpora of television debates and social media reaction. *Language Resources and Evaluation*, pages 1–32, 2019.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[VT99] Ellen M Voorhees and Dawn M Tice. The trec-8 question answering track evaluation. In *TREC*, volume 1999, page 82. Citeseer, 1999.

[VT07] Kimberly Voll and Maite Taboada. Not all words are created equal: Extracting semantic orientation as a function of adjective relevance. In *Australasian Joint Conference on Artificial Intelligence*, pages 337–346. Springer, 2007.

[Wal12] Douglas Walton. Argument mining by applying argumentation schemes. *Studies in Logic*, 4(1):2011, 2012.

[WEK12] Bonnie Webber, Markus Egg, and Valia Kordoni. Discourse structure and language technology. *Natural Language Engineering*, 18(4):437–490, 2012.

[Whi09] Anne Whitaker. Academic writing guide 2010: A step-by-step guide to writing academic papers. *City University of Seattle*, 2009.

[WNB17] Adina Williams, Nikita Nangia, and Samuel R Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*, 2017.

[WRM08] Douglas Walton, Christopher Reed, and Fabrizio Macagno. *Argumentation schemes*. Cambridge University Press, 2008.

[WSB18] Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*, 2018.

[WSM+18] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[WWH04] Theresa Wilson, Janyce Wiebe, and Rebecca Hwa. Just how mad are you? finding strong and weak opinion clauses. In *aaai*, volume 4, pages 761–769, 2004.

[YHG+16] Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. Stacked attention networks for image question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 21–29, 2016.

[ZBSC18] Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. Swag: A large-scale adversarial dataset for grounded commonsense inference. *arXiv preprint arXiv:1808.05326*, 2018.

[ZKZ+15] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27, 2015.

[ZZL15] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.

Appendix A

Student Essay Annotation: Original

```
<Annotation corpus="brat-project">
  <Proposition id="T2">
    <ADU type="conclusion"/>
    <text>we should attach more importance to cooperation</text>
    <TextPosition start="503" end="550"/>
  </Proposition>
  <Proposition id="T1">
    <ADU type="claim"/>
    <text>competition can effectively promote the development of
       economy</text>
    <TextPosition start="78" end="140"/>
    <Relation relationID="RT1T2" type="Default Conflict" typeBinary
       ="1" partnerID="T2"/>
    <Relation relationID="T2Default Conflict" type="Default
       Conflict" typeBinary="1" partnerID="T2"/>
  </Proposition>
  <Proposition id="T3">
    <ADU type="premise"/>
    <text>In order to survive in the competition, companies
       continue to improve their products and service, and as a
       result, the whole society prospers</text>
    <TextPosition start="142" end="283"/>
    <Relation relationID="T1Default Inference" type="Default
       Inference" typeBinary="0" partnerID="T1"/>
```

```
</Proposition>
<Proposition id="T4">
  <ADU type="claim"/>
  <text>through cooperation, children can learn about
     interpersonal skills which are significant in the future
     life of all students</text>
  <TextPosition start="591" end="714"/>
  <Relation relationID="RT4-T2" type="Default Inference"
     typeBinary="0" partnerID="T2"/>
  <Relation relationID="T2Default Inference" type="Default
     Inference" typeBinary="0" partnerID="T2"/>
</Proposition>
<Proposition id="T5">
  <ADU type="premise"/>
  <text>What we acquired from team work is not only how to
     achieve the same goal with others but more importantly, how
     to get along with others</text>
  <TextPosition start="716" end="851"/>
  <Relation relationID="T4Default Inference" type="Default
     Inference" typeBinary="0" partnerID="T4"/>
</Proposition>
<Proposition id="T6">
  <ADU type="premise"/>
  <text>During the process of cooperation, children can learn
     about how to listen to opinions of others, how to
     communicate with others, how to think comprehensively, and
     even how to compromise with other team members when
     conflicts occurred</text>
  <TextPosition start="853" end="1086"/>
  <Relation relationID="T4Default Inference" type="Default
     Inference" typeBinary="0" partnerID="T4"/>
</Proposition>
<Proposition id="T7">
  <ADU type="premise"/>
```

```
  <text>All of these skills help them to get on well with other
     people and will benefit them for the whole life</text>
  <TextPosition start="1088" end="1191"/>
  <Relation relationID="T4Default Inference" type="Default
     Inference" typeBinary="0" partnerID="T4"/>
</Proposition>
<Proposition id="T8">
  <ADU type="claim"/>
  <text>competition makes the society more effective</text>
  <TextPosition start="1332" end="1376"/>
  <Relation relationID="RT8T2" type="Default Conflict" typeBinary
     ="1" partnerID="T2"/>
  <Relation relationID="T2Default Conflict" type="Default
     Conflict" typeBinary="1" partnerID="T2"/>
</Proposition>
<Proposition id="T9">
  <ADU type="premise"/>
  <text>the significance of competition is that how to become
     more excellence to gain the victory</text>
  <TextPosition start="1212" end="1301"/>
  <Relation relationID="T8Default Inference" type="Default
     Inference" typeBinary="0" partnerID="T8"/>
</Proposition>
<Proposition id="T10">
  <ADU type="premise"/>
  <text>when we consider about the question that how to win the
     game, we always find that we need the cooperation</text>
  <TextPosition start="1387" end="1492"/>
  <Relation relationID="T13Default Inference" type="Default
     Inference" typeBinary="0" partnerID="T13"/>
</Proposition>
<Proposition id="T11">
  <ADU type="premise"/>
```

```
    <text>Take Olympic games which is a form of competition for
       instance, it is hard to imagine how an athlete could win the
        game without the training of his or her coach, and the help
        of other professional staffs such as the people who take
       care of his diet, and those who are in charge of the medical
        care</text>
    <TextPosition start="1549" end="1846"/>
    <Relation relationID="T12Default Inference" type="Default
       Inference" typeBinary="0" partnerID="T12"/>
  </Proposition>
  <Proposition id="T12">
    <ADU type="premise"/>
    <text>The winner is the athlete but the success belongs to the
       whole team</text>
    <TextPosition start="1848" end="1915"/>
    <Relation relationID="T13Default Inference" type="Default
       Inference" typeBinary="0" partnerID="T13"/>
  </Proposition>
  <Proposition id="T13">
    <ADU type="claim"/>
    <text>without the cooperation, there would be no victory of
       competition</text>
    <TextPosition start="1927" end="1992"/>
    <Relation relationID="RT13-T2" type="Default Inference"
       typeBinary="0" partnerID="T2"/>
    <Relation relationID="T2Default Inference" type="Default
       Inference" typeBinary="0" partnerID="T2"/>
  </Proposition>
  <Proposition id="T14">
    <ADU type="claim"/>
    <text>a more cooperative attitudes towards life is more
       profitable in one's success</text>
    <TextPosition start="2154" end="2231"/>
    <Relation relationID="RT14-T2" type="Default Inference"
       typeBinary="0" partnerID="T2"/>
```

<Relation relationID="T2Default Inference" type="Default Inference" typeBinary="0" partnerID="T2"/>
</Proposition>
<OriginalText>Should students be taught to compete or to cooperate?

It is always said that competition can effectively promote the development of economy. In order to survive in the competition, companies continue to improve their products and service, and as a result, the whole society prospers. However, when we discuss the issue of competition or cooperation, what we are concerned about is not the whole society, but the development of an individual's whole life. From this point of view, I firmly believe that we should attach more importance to cooperation during primary education.

First of all, through cooperation, children can learn about interpersonal skills which are significant in the future life of all students. What we acquired from team work is not only how to achieve the same goal with others but more importantly, how to get along with others. During the process of cooperation, children can learn about how to listen to opinions of others, how to communicate with others, how to think comprehensively, and even how to compromise with other team members when conflicts occurred. All of these skills help them to get on well with other people and will benefit them for the whole life.

On the other hand, the significance of competition is that how to become more excellence to gain the victory. Hence it is always said that competition makes the society more effective. However, when we consider about the question that how to win the game, we always find that we need the cooperation. The greater our goal is, the more competition we need. Take Olympic games which is a form of competition for instance, it is hard to imagine how an athlete could win the game without the training of his or her coach, and the help of other professional staffs such as the people who take care of his diet, and those who are in charge of the medical care. The winner is the athlete but the success belongs to the whole team. Therefore without the cooperation, there would be no victory of competition.

Consequently, no matter from the view of individual development or the relationship between competition and cooperation we can receive the same conclusion that a more cooperative attitudes towards life is more profitable in one's success.</OriginalText>
</Annotation>

Appendix B

Student Essay Annotation: Prediction

```
<Annotation corpus="brat-project">
  <Proposition id="T0">
    <ADU type="claim" confidence="0.70752794"/>
    <text>Should students be taught to compete or to cooperate?</
       text>
    <TextPosition start="0" end="52"/>
    <Relation relationID="RT0-T1" type="Default Inference"
       confidence="0.6742419" typeBinary="0" partnerID="T1"/>
    <Relation relationID="RT0-T2" type="Default Inference"
       confidence="0.5457543" typeBinary="0" partnerID="T2"/>
    <Relation relationID="RT0-T3" type="Default Inference"
       confidence="0.82408696" typeBinary="0" partnerID="T3"/>
    <Relation relationID="RT0-T4" type="Default Inference"
       confidence="0.64171195" typeBinary="0" partnerID="T4"/>
    <Relation relationID="RT0-T5" type="Default Inference"
       confidence="0.82201535" typeBinary="0" partnerID="T5"/>
    <Relation relationID="RT0-T6" type="Default Inference"
       confidence="0.7917588" typeBinary="0" partnerID="T6"/>
    <Relation relationID="RT0-T7" type="Default Inference"
       confidence="0.7821401" typeBinary="0" partnerID="T7"/>
    <Relation relationID="RT0-T8" type="Default Inference"
       confidence="0.75868577" typeBinary="0" partnerID="T8"/>
    <Relation relationID="RT0-T9" type="Default Inference"
       confidence="0.6491749" typeBinary="0" partnerID="T9"/>
```

```
  <Relation relationID="RT0-T10" type="Default Inference"
     confidence="0.7542227" typeBinary="0" partnerID="T10"/>
  <Relation relationID="RT0-T12" type="Default Inference"
     confidence="0.6366765" typeBinary="0" partnerID="T12"/>
  <Relation relationID="RT0-T14" type="Default Inference"
     confidence="0.74888575" typeBinary="0" partnerID="T14"/>
  <Relation relationID="RT0-T15" type="Default Inference"
     confidence="0.6513359" typeBinary="0" partnerID="T15"/>
</Proposition>
<Proposition id="T1">
  <ADU type="claim" confidence="0.6714652"/>
  <text>It is always said that competition can effectively
     promote the development of economy.</text>
  <TextPosition start="55" end="140"/>
  <Relation relationID="RT1-T2" type="Default Inference"
     confidence="0.49492022" typeBinary="0" partnerID="T2"/>
  <Relation relationID="RT1-T4" type="Default Inference"
     confidence="0.6072445" typeBinary="0" partnerID="T4"/>
  <Relation relationID="RT1-T5" type="Default Inference"
     confidence="0.58558154" typeBinary="0" partnerID="T5"/>
</Proposition>
<Proposition id="T2">
  <ADU type="premise" confidence="0.57934546"/>
  <text>In order to survive in the competition, companies
     continue to improve their products and service, and as a
     result, the whole society prospers.</text>
  <TextPosition start="142" end="283"/>
  <Relation relationID="RT2-T9" type="Default Inference"
     confidence="0.56399" typeBinary="0" partnerID="T9"/>
  <Relation relationID="RT2-T12" type="Default Inference"
     confidence="0.51818097" typeBinary="0" partnerID="T12"/>
</Proposition>
<Proposition id="T3">
  <ADU type="claim" confidence="0.5139293"/>
```

```
    <text>However, when we discuss the issue of competition or
       cooperation, what we are concerned about is not the whole
       society, but the development of an individual's whole life.<
       /text>
    <TextPosition start="285" end="454"/>
    <Relation relationID="RT3-T5" type="Default Inference"
       confidence="0.72832954" typeBinary="0" partnerID="T5"/>
    <Relation relationID="RT3-T7" type="Default Inference"
       confidence="0.5502592" typeBinary="0" partnerID="T7"/>
    <Relation relationID="RT3-T8" type="Default Inference"
       confidence="0.51797146" typeBinary="0" partnerID="T8"/>
    <Relation relationID="RT3-T15" type="Default Inference"
       confidence="0.50491124" typeBinary="0" partnerID="T15"/>
  </Proposition>
  <Proposition id="T4">
    <ADU type="claim" confidence="0.6329969"/>
    <text>From this point of view, I firmly believe that we should
       attach more importance to cooperation during primary
       education.</text>
    <TextPosition start="456" end="575"/>
    <Relation relationID="RT4-T5" type="Default Inference"
       confidence="0.5986914" typeBinary="0" partnerID="T5"/>
    <Relation relationID="RT4-T6" type="Default Inference"
       confidence="0.5261127" typeBinary="0" partnerID="T6"/>
    <Relation relationID="RT4-T12" type="Default Inference"
       confidence="0.4926278" typeBinary="0" partnerID="T12"/>
  </Proposition>
  <Proposition id="T5">
    <ADU type="premise" confidence="0.59917516"/>
    <text>First of all, through cooperation, children can learn
       about interpersonal skills which are significant in the
       future life of all students.</text>
    <TextPosition start="577" end="714"/>
  </Proposition>
  <Proposition id="T6">
```

```
    <ADU type="premise" confidence="0.43773472"/>
    <text>What we acquired from team work is not only how to
       achieve the same goal with others but more importantly, how
       to get along with others.</text>
    <TextPosition start="716" end="851"/>
  </Proposition>
  <Proposition id="T7">
    <ADU type="premise" confidence="0.73656386"/>
    <text>During the process of cooperation, children can learn
       about how to listen to opinions of others, how to
       communicate with others, how to think comprehensively, and
       even how to compromise with other team members when
       conflicts occurred.</text>
    <TextPosition start="853" end="1086"/>
  </Proposition>
  <Proposition id="T8">
    <ADU type="premise" confidence="0.64947534"/>
    <text>All of these skills help them to get on well with other
       people and will benefit them for the whole life.</text>
    <TextPosition start="1088" end="1191"/>
  </Proposition>
  <Proposition id="T9">
    <ADU type="claim" confidence="0.59651214"/>
    <text>On the other hand, the significance of competition is
       that how to become more excellence to gain the victory.</
       text>
    <TextPosition start="1193" end="1301"/>
  </Proposition>
  <Proposition id="T10">
    <ADU type="claim" confidence="0.75686955"/>
    <text>Hence it is always said that competition makes the
       society more effective.</text>
    <TextPosition start="1303" end="1376"/>
  </Proposition>
  <Proposition id="T11">
```

```
  <ADU type="claim" confidence="0.56730825"/>
  <text>However, when we consider about the question that how to
     win the game, we always find that we need the cooperation.</
     text>
  <TextPosition start="1378" end="1492"/>
  <Relation relationID="RT11-T15" type="Default Inference"
     confidence="0.45236707" typeBinary="0" partnerID="T15"/>
</Proposition>
<Proposition id="T12">
  <ADU type="claim" confidence="0.51830024"/>
  <text>The greater our goal is, the more competition we need.</
     text>
  <TextPosition start="1494" end="1547"/>
</Proposition>
<Proposition id="T13">
  <ADU type="premise" confidence="0.50758696"/>
  <text>Take Olympic games which is a form of competition for
     instance, it is hard to imagine how an athlete could win the
      game without the training of his or her coach, and the help
      of other professional staffs such as the people who take
     care of his diet, and those who are in charge of the medical
      care.</text>
  <TextPosition start="1549" end="1846"/>
</Proposition>
<Proposition id="T14">
  <ADU type="premise" confidence="0.5804807"/>
  <text>The winner is the athlete but the success belongs to the
     whole team.</text>
  <TextPosition start="1848" end="1915"/>
</Proposition>
<Proposition id="T15">
  <ADU type="claim" confidence="0.5888416"/>
  <text>Therefore without the cooperation, there would be no
     victory of competition.</text>
  <TextPosition start="1917" end="1992"/>
```

```
  </Proposition>
  <Proposition id="T16">
    <ADU type="claim" confidence="0.52110475"/>
    <text>Consequently, no matter from the view of individual
       development or the relationship between competition and
       cooperation we can receive the same conclusion that a more
       cooperative attitudes towards life is more profitable in one
       's success.</text>
    <TextPosition start="1994" end="2231"/>
  </Proposition>
  <OriginalText>Should students be taught to compete or to
     cooperate?

It is always said that competition can effectively promote the
    development of economy. In order to survive in the competition,
    companies continue to improve their products and service, and as
     a result, the whole society prospers. However, when we discuss
    the issue of competition or cooperation, what we are concerned
    about is not the whole society, but the development of an
    individual's whole life. From this point of view, I firmly
    believe that we should attach more importance to cooperation
    during primary education.
First of all, through cooperation, children can learn about
    interpersonal skills which are significant in the future life of
     all students. What we acquired from team work is not only how
    to achieve the same goal with others but more importantly, how
    to get along with others. During the process of cooperation,
    children can learn about how to listen to opinions of others,
    how to communicate with others, how to think comprehensively,
    and even how to compromise with other team members when
    conflicts occurred. All of these skills help them to get on well
     with other people and will benefit them for the whole life.
```

On the other hand, the significance of competition is that how to become more excellence to gain the victory. Hence it is always said that competition makes the society more effective. However, when we consider about the question that how to win the game, we always find that we need the cooperation. The greater our goal is, the more competition we need. Take Olympic games which is a form of competition for instance, it is hard to imagine how an athlete could win the game without the training of his or her coach, and the help of other professional staffs such as the people who take care of his diet, and those who are in charge of the medical care. The winner is the athlete but the success belongs to the whole team. Therefore without the cooperation, there would be no victory of competition.

Consequently, no matter from the view of individual development or the relationship between competition and cooperation we can receive the same conclusion that a more cooperative attitudes towards life is more profitable in one's success.</OriginalText>
</Annotation>