Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorthesis

Thorben Schomacker

Application of Transformer-based Methods to Latin Text Analysis

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Thorben Schomacker

Application of Transformer-based Methods to Latin Text Analysis

Bachelorarbeit eingereicht im Rahmen Bachelorprüfung

im Studiengang Bachelor of Science Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Marina Tropmann-Frick
Zweitgutachter: Prof. Dr. Olaf Zukunft

Abgegeben am 16.11.2020

Thorben Schomacker

Thema der Arbeit
Anwendung Transformer-basierter Methoden für die Analyse lateinischer Texte

Stichworte
NLP, Extractive Summarization, Transformer, BERT, BertSum, Transfer Learning

Kurzzusammenfassung
Textzusammenfassung ist ein etabliertes Problem im NLP-Bereich. Der rasch anwachsende Erfolg von deep learning Algorithmen führte zur Entwicklung des attention Mechanismus, welcher wiederum die Grundlage für die Transformer Architektur bildet. Die Transformer Architektur ist ein transfer learning Ansatz NLP Probleme zu lösen. BERT, ein pre-trained Transformer Modell, hat herausragende Ergebnis beim Lösen verschiedener NLP-Probleme erzielt. In dieser Abschlussarbeit wird BertSum, eine Erweiterung BERTs spezialisiert auf extrahierende Textzusammenfassung, auf neuronale Textzusammenfassung von lateinischen und deutschen Texten angewandt. Dies stellt eine besondere Herausforderung dar, denn die Texte wurden zu einer Zeit verfasst, in der noch keine festgeschriebene Orthographie, Morphologie und Semantik existierte.

Thorben Schomacker

Title of the paper
Application of Transformer-based Methods to Latin Text Analysis

Keywords
NLP, Extractive Summarization, Transformer, BERT, BertSum, Transfer Learning

Abstract
Text summarization is an established problem in the field of NLP. The rapidly growing success of deep learning algorithms in solving NLP problems has led to the attention mechanism, which is the foundation for the Transformer architecture, a transfer learning approach for NLP tasks. BERT, a pre-trained Transformer model, has performed exceptionally well on various NLP tasks. This thesis applies BertSum, an enhancement of BERT specialized for extractive text summarization, to the neural text summarization of Latin and German texts. The distinctiveness of the chosen corpus is that it consists of medieval documents. This poses a challenge because the documents were written in a time where orthography, morphology, and semantics were not well defined.

Contents

1 Introduction

For current historical research on the middle ages, formulae are important sources. Researchers have very few documents from the early middle ages and the formulae collections include texts which normally have not been preserved in archives. Formulae often give invaluable information on everyday life of the early middle ages. Since they are written in Latin, in a time where orthography, morphology, and semantics were not well defined, they are not easy to interpret or to be used for research. Regests were introduced to orient the reader by providing a summarization of the document. They are published together with their corresponding text in a collection of letters or in a collection of records. They are not written in the language of their corresponding text but of the collection or edition they are published in. In the case of this thesis the regests are written in German and their corresponding text is written in Latin. Writing a regest requires an expenditure of time as well as financial resources. Additionally, these regests are sometimes copyright protected so they cannot be freely used in open science. The goal of this thesis is to apply recent developments from the field of Natural Language Processing (NLP) to this problem to save resources and contribute to open science in humanities by generating open regests.

1.1 An estimation of the topic

The subject of this thesis is the result of a very fruitful collaboration of the two fields: computer science and history. To give a more complete introduction to the issue, a historian was asked to add his perspective:

Writing regests of medieval documents is a time consuming and difficult enterprise. Ideally a regest summarizes the content of a document (charter or diploma) in a brief form, thus helping the reader to get a grasp of the underlying legal act (usually a donation to an institution, an act of sale or an exchange of property, often accompanied by individual clauses like the establishment of prayer services or restrictions in the use of the property). The importance of regests becomes even more evident when it comes to documents of the high and late middle ages when their numbers rise sharply into the 10thousands per year. According to *current practice, these documents are not edited anymore, but merely represented by a regest.*

There are different approaches when it comes to composing a regest: One tradition emphasizes the legal character of the act, using standardized terminology when summarizing it, while another approach aims at staying as close to the terminology used by the individual document. Furthermore, according to German tradition, regests have to be composed in one single sentence. Composing a regest hence requires first to translate the document to make sure its legal content (and language) are fully understood. Being able to automatically create regests from the original latin text hence would mean a considerable and time-saving aid.

- Dr. Horst Lößlein, research associate[1]

1.2 Structure of this thesis

This thesis is divided into five chapters. The ***first chapter*** describes the current developments in neural language models. The ***second chapter*** talks about BERT, a transfer learning approach built on the ideas described in the first chapter. The ***third chapter*** presents BertSum, a summarization-specialized enhancement of the previously introduced BERT. BertSum is then applied to the task of writing these regests in a few experiments in the ***fourth chapter***. The ***fifth chapter*** discusses the quality of the results of those experiments and shows the strengths and weaknesses of the chosen approach. It also focuses on how future work can learn from and overcome these weaknesses.

2 Neural Language Models

The problem described in chapter 1 can also be defined as making a prediction (in this case a summary) based on a set of data. One promising way to solve such problems are Neural Language Models. These models make use of neural networks and a key feature of them is that they are capable of recognizing that two words are similar without losing the ability to

[1]https://www.geschichte.uni-hamburg.de/arbeitsbereiche/mittelalter/personen/horst-loesslein.html (10/20/2020)

encode each word as distinct from the other. This is accomplished by using a distributed representation of words (Bengio et al. 2000) (Goodfellow et al. 2016, p. 464)

2.1 Word embeddings

The basis for a neural language model is distributed representations. The model learns a distributed representation for each word. These distributed representations are feature vectors. Each feature gets a number. Figure 1 visualizes the two-dimensional representations of words. The x-axis is one semantical feature and y-axis is another. For example, ***French*** and ***English*** are very close. This means that they represent similar semantics or share information. This information could, for example, be that they are both a language. Distributed representations allow the model to share statistical strength between one word (and its context) and other similar words (and their contexts). (Goodfellow et al. 2016, p. 464)

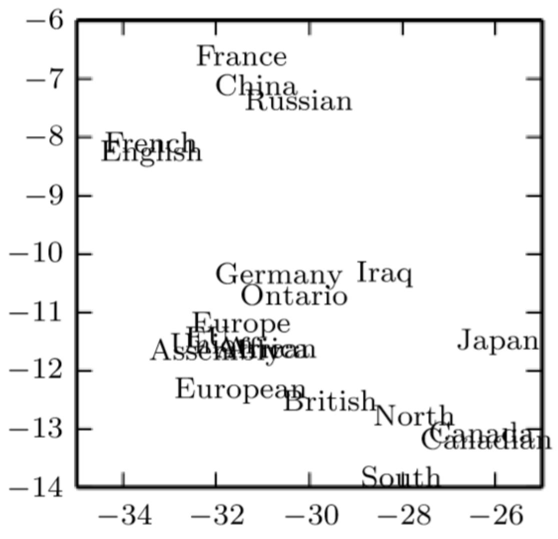


Figure 1: Example Word Embedding with data from (Bahdanau et al. 2016) and visualized by (Goodfellow et al. 2016, p. 465)

2.2 Recurrent Neural Network

A family of neural networks or neural language models, which is specialized on processing sequential data, such as continuous texts, are Recurrent Neural Networks RNNs (Rumelhart et al. 1986). Most RNNs can process sequences of values of variable length. One main idea of RNNs is sharing parameters across different parts of a model. This permits models to be extended and applied to sequences of different sizes and to generalize across them. Parameter sharing is especially important when a piece of information can be found at different positions in the sequence. For example, "I went shopping yesterday" and "Yesterday I went shopping". If the task were to extract time information, the model should recognize "yesterday" whether it appears at the beginning or at the end of the sequence. (Goodfellow et al. 2016, p. 373) RNNs could be thought of as a chain of multiple copies of the same network, each one passing or sharing the parameters to its successor.

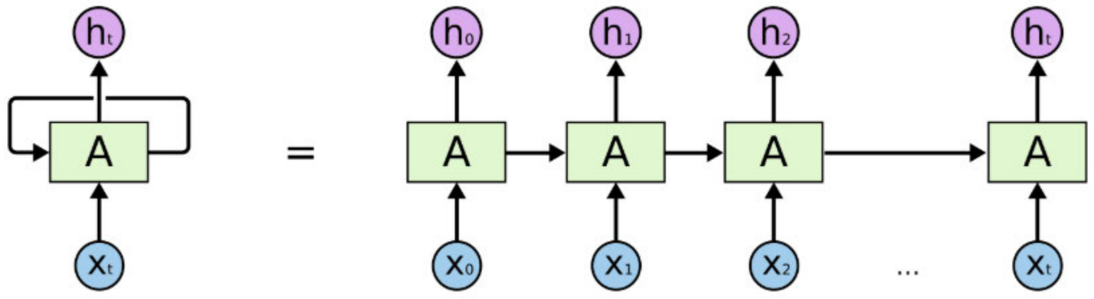


Figure 2: An unrolled RNN by (Olah 2015)

RNNs make it possible to map an input sequence to an output value or to an output sequence of the same length as the input sequence. (Goodfellow et al. 2016, p. 396). RNNs could be applied to problems like stock price prediction. The input is a sequence of the historical stock prices (each is one number) and the output is the future stock price for a specific point in time. Another use case could be predicting the next word. The input is a sequence of the previously written words and the output is the predicted next word. The most used representative of this family is the Long Short-term Memory (LSTM) introduced by (Hochreiter and Schmidhuber 1997). LSTMs use so-called gates as paths through time that have derivatives that neither vanish nor explode (Goodfellow et al. 2016, 408f.). This makes LSTMs applicable to problems involving long-term dependencies. The fact that the input length determines the output length is the major limitation of this model family.

2.3 Encoder-Decoder Framework

Many tasks in Natural Language Processing, for example machine translation, deal with producing a variable length output from variable length input. The previously introduced RNNs are limited to producing output of the same length as the input. (Cho et al. 2014) and (Sutskever et al. 2014) were the first to overcome this limitation by introducing the so-called encoder-decoder framework. They applied this approach to the field of machine translation. Most applications of this framework to the task of summarization are built on the findings of (Li et al. 2015), (Rush et al. 2015) and (Nallapati et al. 2016). All three describe the application of the framework to the task as very promising.
The basic idea behind the frameworks could be explained by imagining it as three different sub processes (Goodfellow et al. 2016, p. 476):

1. ***Reading*** raw data (such as source words) and converting them into distributed representations, with one feature vector associated with each word position. This process can also be considered as feature extraction.
2. ***Memorizing*** / Storing the reader's output as a list of feature vectors. These elements can be retrieved later, without the necessity of visiting all of them or retrieving them in the order of the input.
3. ***Exploiting*** the content of the memory to sequentially perform a task. At each time step this process has the ability to look at the content of one memory element (or a few with different weights). This process generates the output for the task. For instance, it translates a text.

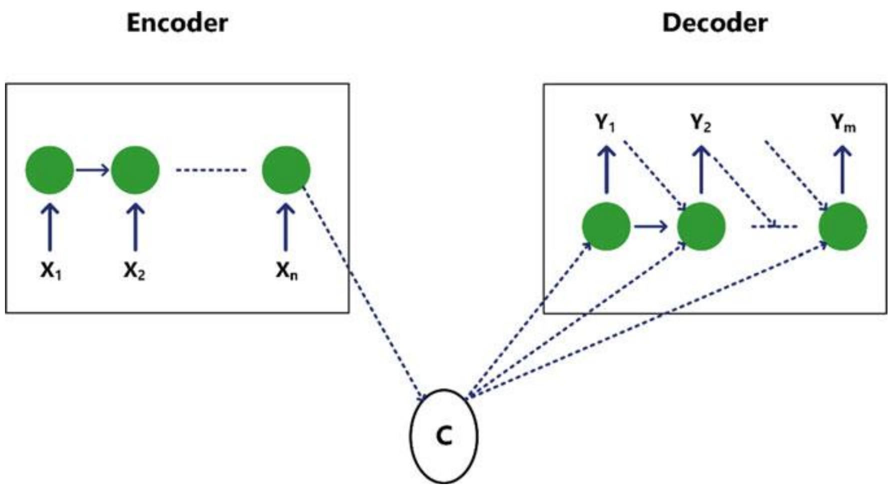


Figure 3: An illustration of the first encoder-decoder based model proposed for machine translation by (Asadi and Safabakhsh 2020)

Figure 3 shows the three sub processes very well. The first process, ***reading,*** could be an RNN (Cho et al. 2014)(Sutskever et al. 2014)(Jean et al. 2014) or a convolutional network (Nal Kalchbrenner and Phil Blunsom 2013). It is called an ***encoder***. The encoder produces a summary of the input data by means of the ***memorizing*** sub process. This summary is the context c of the input data. c gives global level information on all the inputs and helps to discern the most important information (Thiruvengadam 2019) (Vaswani et al. 2017). c could be a list of vectors, a tensor or a vector. (Goodfellow et al. 2016, p. 474). The ***decoder exploits*** or processes the input represented by c to generate an output. (Asadi and Safabakhsh 2020)

2.4 Attention

The encoder-decoder framework, when dealing with larger inputs, leads to complex models and sometimes reaches its understanding limitations. This is caused by the fact that the model must compress all the necessary information of the input into the fixed-length context vector. The same thing applies to humans when they are reading and understanding a text (taking the text as an input). One approach to reduce the complexity of the models and improve their results is the attention mechanism. It has the human eye as its biological role model. Except for a tiny patch, the human eye is mostly very low resolution. This patch is called the fovea and only sees a thumbnail-sized area. By making several eye movements, called saccades, the human brain is able to capture the most visually salient or task-relevant information (Goodfellow et al. 2016, p. 366). Humans apply this for instance on texts or images. Instead of focusing on single words or pixels, they pay attention to the salient parts of the text or image. The "highlighting" of this more relevant or salient information is shown in Figure 4. The lighter areas in the right pictures are more relevant to determine a description of the picture, so more attention was paid to them.

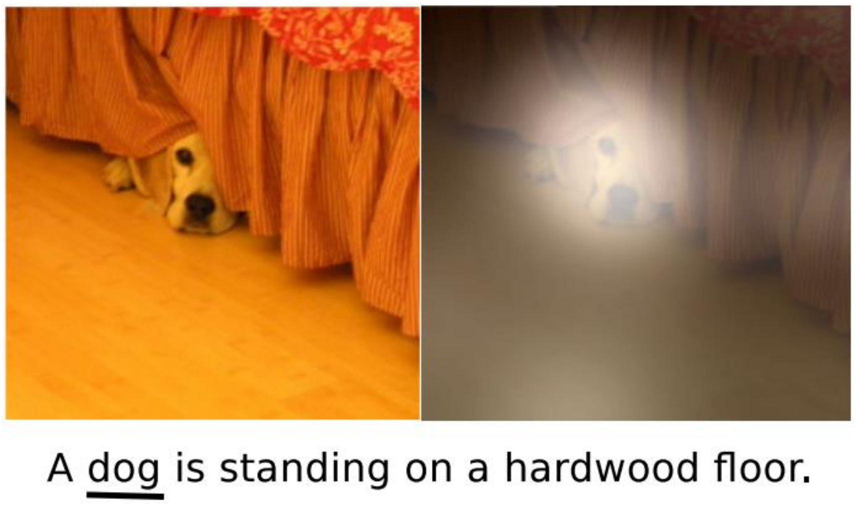


Figure 4: Example of Visual Attention by (Xu et al. 2016)

Attention in deep learning works similarly to its biological role model. It is needed when the text is longer than a few words, since the encoder state at the end would forget the information processed at the beginning. This “forgetting” problem is called the vanishing gradient problem. Attention helps the decoder attend to important tokens throughout longer text sequences. Attention was firstly applied in neural machine translation by (Bahdanau et al. 2016) and (Li et al. 2015). Neural machine translation attempts to create a neural network that reads and translates whole sentences instead of phrases, as it was done traditionally. In short the model of (Bahdanau et al. 2016) encodes the input sentence into a sequence of vectors and chooses a subset of these vectors adaptively while decoding the translation so that this subset receives more attention from the translation . This is shown in Figure 5.

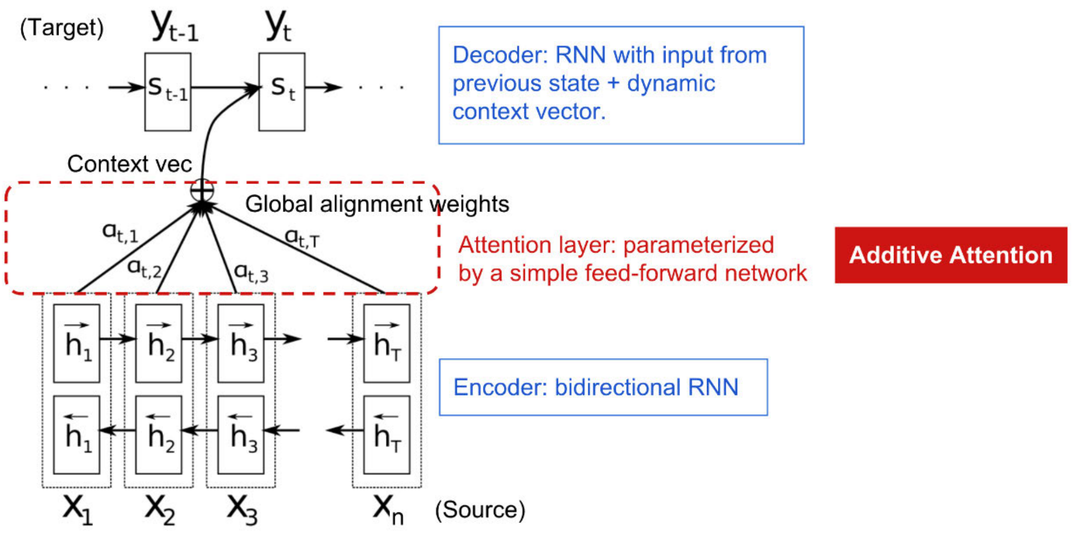

(Target)
y_{t-1}
y_t
s_{t-1}
s_t
Decoder: RNN with input from previous state + dynamic context vector.
Context vec
Global alignment weights
$a_{t,1}$
$a_{t,2}$
$a_{t,3}$
$a_{t,T}$
Attention layer: parameterized by a simple feed-forward network
Additive Attention
$\overrightarrow{h}_1$ $\overrightarrow{h}_2$ $\overrightarrow{h}_3$ $\overrightarrow{h}_T$
$\overleftarrow{h}_1$ $\overleftarrow{h}_2$ $\overleftarrow{h}_3$ $\overleftarrow{h}_T$
Encoder: bidirectional RNN
X_1 X_2 X_3 X_n
(Source)

Figure 5: Example of attention in translation by (Bahdanau et al. 2016) and enriched by (Weng 2018)

This attention model differs from the basic encoder-decoder model in two main ways: First, instead of passing only the last hidden state of the encoding stage, the encoder passes all the hidden states to the decoder. Second, an attention decoder performs an extra step before producing its output. To focus on the parts of the input that are relevant to this decoding time step, the decoder does the following at each time step (Bahdanau et al. 2016) (Alammar 2018a):

1. The attention decoder RNN takes in the embedding of the <END> token, and an initial decoder hidden state h_j
2. This RNN produces an output and the hidden state vector h_i. (The output is irrelevant)
3. ***Attention Step***: The encoder hidden states (***annotations***) h_i are used to calculate the context vector c_i:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

Equation 1: context vector (Bahdanau et al. 2016)

The weight α_{ij} of each annotation h_j is computed by

$$\alpha_{ij} = \frac{exp\left(e_{ij}\right)}{\sum_{k=1}^{T_x} exp\left(e_{ik}\right)}$$

Equation 2: annotation weight (Bahdanau et al. 2016)

where

$$e_{ij} = a\left(s_{i-1}, h_j\right)$$

Equation 3: The associated energy of α_{ij} (Bahdanau, Cho, and Bengio 2016)

4. h_i and c_i are concatenated into the vector s_i
5. s_i is passed through a feedforward neural network
6. The output is the target word y_i

The problems and previous limitations described in this chapter are the same for generating summaries. For instance, the words in Figure 6 that receive the most attention during translation would also be important for summarizing the texts. So, applying the attention-mechanism to text summarization can lead to state-of-the-art results on performing text summarization as shown in (Liu 2019) and (Liu and Lapata 2019).

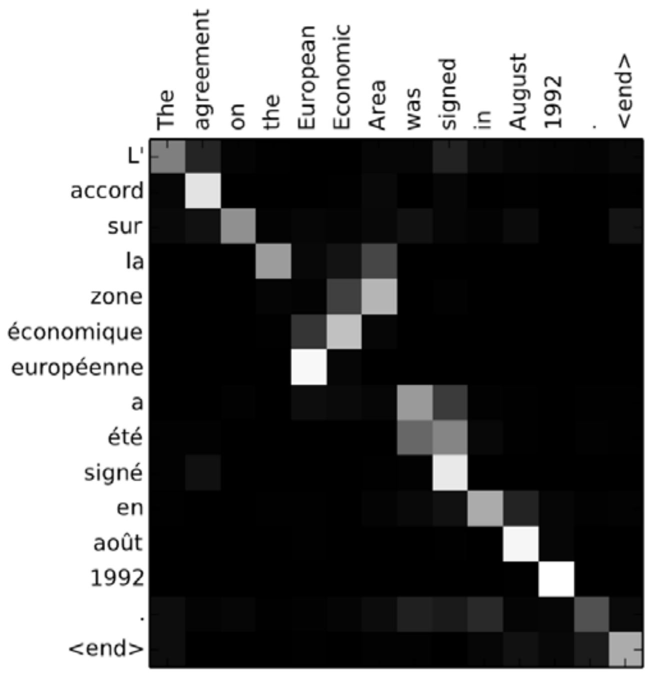


Figure 6: The x-axis and y-axis of each plot correspond to the words in the source sentence (English) and the generated translation (French), respectively. Each pixel shows the weight αij of the annotation of the j-th source word for the i-th target word (Bahdanau et al. 2016)

2.5 Transformer

(Vaswani et al. 2017) further developed attention into ***self-attention*** in their newly introduced transformer architecture. A complete overview of the transformer architecture is illustrated in Figure 10. The concept of self-attention could be explained by the following example:
We want to translate the following input sentence:

"The animal didn't cross the street because it was too tired"

The model for translation tries to understand the sentence word by word. One possible pitfall is the correct understanding of pronouns. In this particular case the word "it". Self-attention makes it possible to associate "it" correctly with "the animal" instead of wrongly with "the street". This is possible because self-attention permits the model to look back to a previously processed word when encoding the current one. So, the main addition to the previously introduced attention is that self-attention allows the inputs to interact with each other ("self") and determine which other input they should pay more attention to ("attention"). Figure 7 depicts an example of self-attention. The input "it" interacts with many previous inputs. Mainly with "The" and "animal" which indicates a link between them. The transformer architecture is based on the encoder-decoder framework, so it uses the information incorporated in RNNs hidden state.

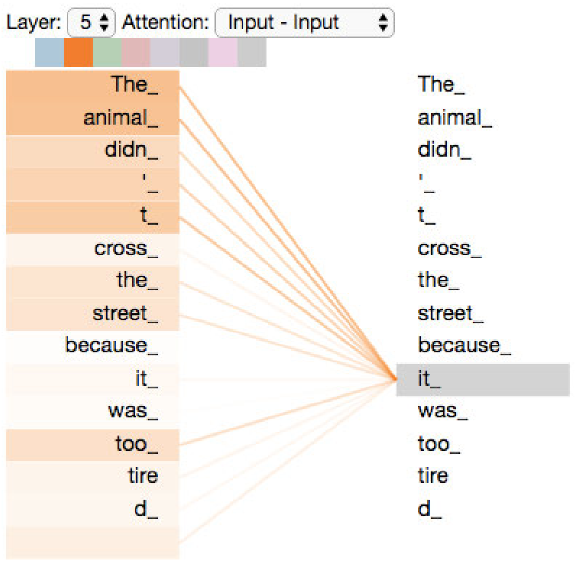


Figure 7: Example of Self-Attention by (Alammar 2018b)

2.5.1 Scaled Dot-Product Attention

One part of self-attention is a new attention function: ***Scaled Dot-Product Attention***. By introducing this new function (Vaswani et al. 2017) attempted to outperform the most commonly used functions: additive attention (Bahdanau et al. 2016) and dot-product attention. ***Scaled Dot-Product Attention*** is computed as follows:

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

Equation 4: Scaled Dot-Product Attention Function (Vaswani et al. 2017)

where Q is the matrix of the queries simultaneously; d_k are the keys the dimension of and d_v are the values of the dimension. Figure 8 visualizes this attention function.

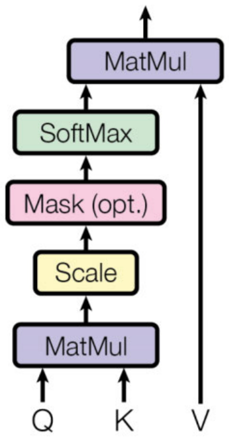


Figure 8: Scaled Dot-Product Attention Function (Vaswani et al. 2017)

2.5.2 Multi-head Attention

(Vaswani et al. 2017) linearly project the queries, keys, and values h times, instead of performing a single attention function. Then the attention function is performed on each of these projected versions thereby yielding d_v-dimensional output values. These values are concatenated and projected to produce the final values as illustrated in Figure 9

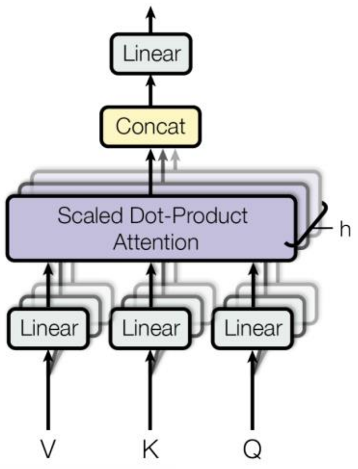


Figure 9: multi-head attention by (Vaswani et al. 2017)

$$MultiHead(Q, K, V) = Concat(head_1, \ldots, head_h)W^O$$
$$wherehead_i = Attention)$$

Equation 5: multi-head attention by (Vaswani et al. 2017)

Multi-head attention is applied in three different ways each opening up a new possibility (Vaswani et al. 2017):

1. It allows the decoder at every position to attend to all positions in the input sequence
2. It allows the encoder to attend to all positions in the previous layer of the encoder
3. It allows the decoder to attend to all positions in the previous layer of the decoder. A possible risk in allowing this is to lose the ability to predict based on previous values (auto-regressive property). To preserve this ability leftward information is prevented.

2.5.3 Why Self-Attention

There are a few widely used alternatives to self-attention-based models. Table 1 shows the findings of (Vaswani et al. 2017) comparing Self-Attention in three areas to similar approaches:

1. Total complexity per layer
2. Amount of parallelizable computations (minimum number of sequential operations)
3. Maximum path length between long-range dependencies

Layer Type	**Complexity per Layer**	**Sequential Operations**	**Maximum Path Length**
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$

Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

Table 1: Self-Attention in Comparison (Vaswani et al. 2017)

Although the first and the second area are very important for building scalable machine-learning models, the decision to choose an attention-based approach for this thesis is mainly based on the fact that Self-Attention has the shortest Maximum Path Length. The shorter the paths the easier it is for the model to learn long-term dependencies. And learning long-term dependencies is a key task when summarizing texts. Learning long-term dependencies is important because the relevant information is spread across the text. A long-term dependency could for instance be that a piece of information in the first paragraph is relevant for understanding the penultimate paragraph. And both paragraphs could be important for understanding and summarizing the whole text.

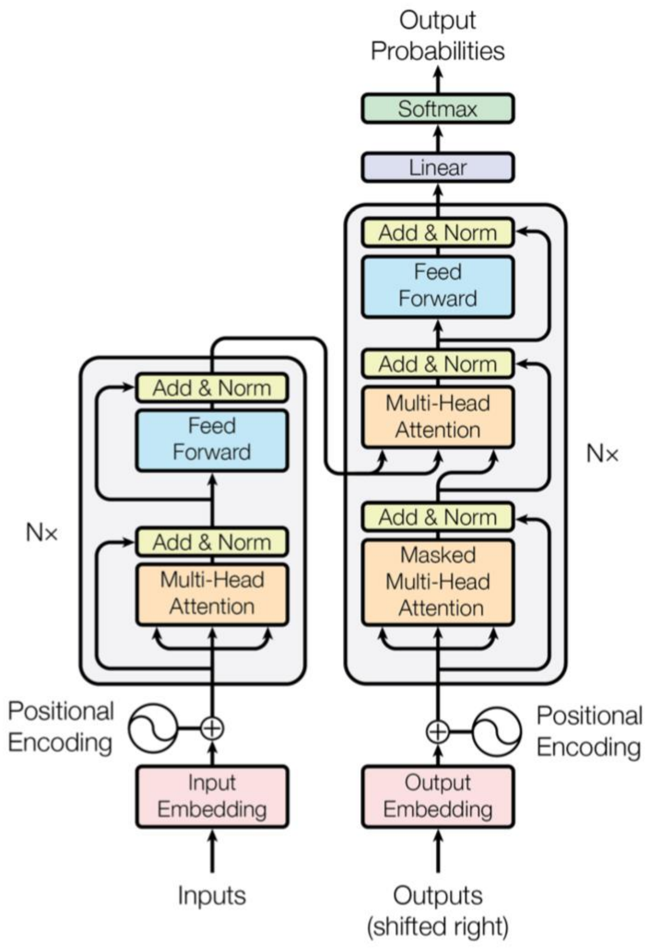


Figure 10: The Transformer Architecture by (Vaswani et al. 2017)

3 Bert

Contextualized representations and attention-based language models are very for solving difficult NLP tasks such as text summarization. (Devlin et al. 2019b) introduced Bidirectional Encoder Representations from Transformers (***BERT***) which includes both. BERT additionally offers the possibility of using pre-trained models which makes it very successful solving NLP tasks where very little training data is available. Unlike similar models BERT leverages bidirectionality to create more precise models.

3.1 Pre-trained Models

The recent progress of hardware and methodology for collecting and analysis has resulted in more detailed and precise NLP models. These models are very time and energy consuming. (Strubell et al. 2019) compared the CO_2 emission of training NLP models to some other familiar values.

Consumption	**CO_2e (lbs)**
Air travel, 1 passenger, NY↔SF	1984
Human life, avg, 1 year	11,023
American life, avg, 1 year	36,156
Car, avg incl. fuel, 1 lifetime	126,000
Training one model (GPU)	
NLP pipeline (parsing, SRL)	39
w/ tuning & experimentation	78,468
Transformer (big)	192
w/ neural architecture search	626,155

Table 2: Estimated CO2 emissions from training common NLP models, compared to familiar consumption by (Strubell et al. 2019)

Table 2 indicates that the re-using parts of models or even whole models could save a large amount CO_2 emission. Additionally, the lack of large, task-specific data sets is one of the biggest challenges in NLP. Both issues are tackled by ***transfer learning***. The basic idea could be boiled down to separating the model's creation by training a generalized model (***pre-trained model***) on general text data sets like Wikipedia articles to gain a general understanding of the target language. Afterwards the model gets specialized to the task. The pre-trained model could be re-used for different tasks and even smaller task-specific data sets can lead to precise models. So, pre-trained models could help save time, energy and solve tasks without having a large, task-specific data set. Because the data set of this thesis is rather small, using a pre-trained model could be promising.

3.2 Strategies for using pre-trained Models

The previous chapter explained the motivation behind pre-trained models. These pre-trained models have a rather broad or general language understanding. They are trained on a large dataset. The second step is to fit the model to solve the task with a smaller but more specific data set. There are two strategies for applying them to specific down-stream tasks:

1. ***feature based***: task specific architectures that include the pre-trained approach as additional features. Such as ELMo (Peters et al. 2018) illustrated in Figure 11
2. ***fine-tuning***: uses minimal task-specific parameters and is trained on the downstream tasks by simply fine-tuning all pre-trained parameters. Such as BERT (Devlin et al. 2019b) illustrated in Figure 12.

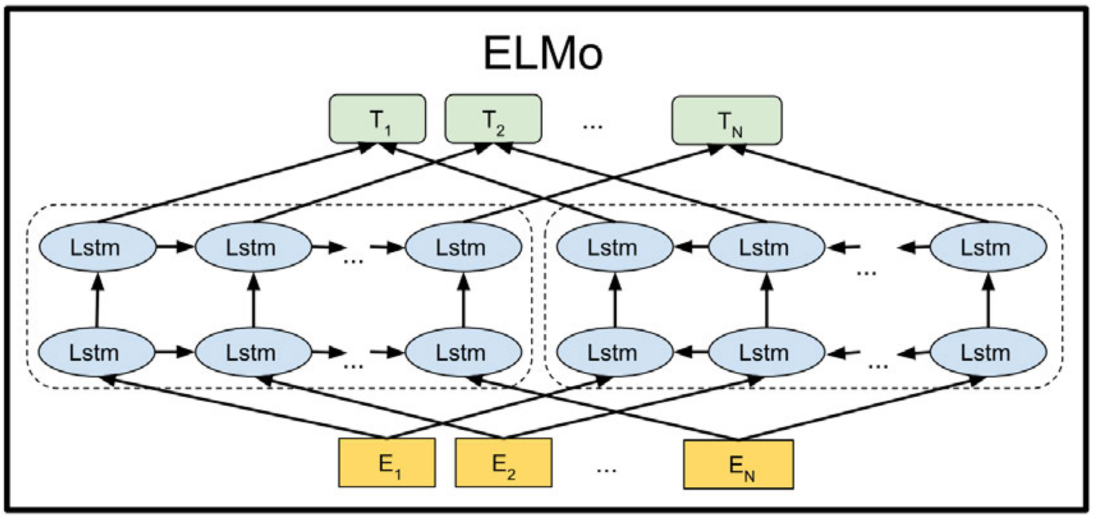


Figure 11: ELMo illustrated by (Devlin et al. 2019b)

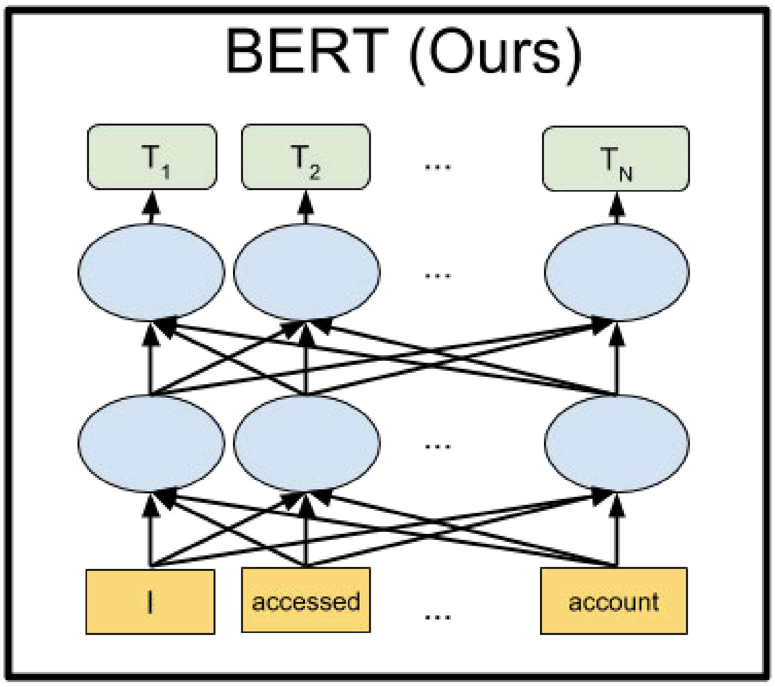


Figure 12: Tokenization in BERT by (Devlin et al. 2019b)

Fine-tuning can also be described as taking the weights of a (pre-)trained model and using them as initialization for a new model. In this thesis, a model is trained on a large data set from the same domain, in this case in the same language as the fine-tuning data set, to gain a general language understanding and is later fine-tuned with a more specific data set. It uses the approach described in (Devlin et al. 2019b). (Devlin et al. 2019b) argue that one major advantage of this approach is that it overcomes previous pre-trained-representation-based models' limitation by introducing a bidirectional approach: ***BERT*** (Bidirectional Encoder Representations from Transformers).

3.3 Bidirectionality

Bidirectionality means that the contextual representation of a word is based on the previous and the next input, thereby gaining a better representation of the word. Take for example the sentence: "I accessed the bank account". A unidirectional model would base its representation of "bank" only on "I accessed the" and leaving out "account" because its representation is only based on previous information. In contrast BERT bases its representation on "I accessed the … account" because its representation is based on previous and subsequent information. This leads to a more detailed and thereby possibly more precise understanding of the word "bank". This bidirectionality is produced by using a "masked language model" (MLM) pretraining objective, which is inspired by the Cloze task (Taylor 1953). The MLM randomly masks some of the tokens from the input and aims to predict the original vocabulary id of the masked word based only on its context, thereby enabling the representation to combine the left and the right context. This makes it possible to pretrain a deep bidirectional Transformer. The masking procedure can be illustrated like this:

		input	output
80% of the time	Replace word with [MASK] token	my dog is hairy	my dog is [MASK]
10% of the time	Replace word with random word	my dog is hairy	my dog is apple
10% of the time	Keep word unchanged	my dog is hairy	my dog is hairy

Table 3: masking example by (Devlin et al. 2019b)

Table 3 shows that the Transformer encoder can and does not know which words it will be asked to predict or which have been replaced by random words. Thereby it is forced to have a distributional contextual representation of every token. (Devlin et al. 2019b) even conclude, after testing different masking strategies, that random replacement does not harm the model's language understanding capability.

3.4 Next Sentence Prediction

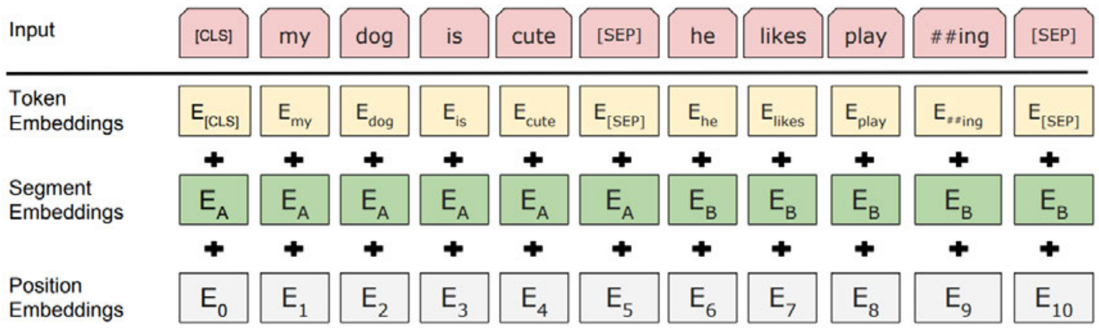
While bidirectionality enables a better word-level understanding, "next sentence prediction" (NSP) helps to better understand the relationship between two sentences. NSP jointly pretrains text-pair representations. It is an important feature of BERT because many important downstream tasks such as Question Answering (QA) and Natural Language Inference (NLI) are based on understanding these relationships. In order to train a model which understands these relationships (Devlin et al. 2019b) used a binarized NSP task generated from a monolingual corpus. When choosing a sentence A and a sentence B, in 50% of the cases B is the actual sentence that followed A (labeled: IsNext) and in 50% it is a random sentence from the corpus (labeled: NotNext). An example is shown in Table 4.

		A	B	Label
50% of the time	B is the actual sentence	[CLS] the man went to [MASK] store [SEP]	he bought a gallon [MASK] milk [SEP]	IsNext
50% of the time	B is a random sentence from the corpus	[CLS] the man went to [MASK] store [SEP]	penguin [MASK] are flight ##less birds [SEP]	NotNext

Table 4: next sentence prediction example by (Devlin et al. 2019b)

3.5 Input/Output Representations

To enable BERT to handle a variety of down-stream tasks, (Devlin et al. 2019b) assigned a sentence input representation which can be a sentence or a pair of sentences in one token. So a "sentence" in (Devlin et al. 2019b) is more like a text span than a linguistic sentence. BERT consists of three different embedding layers: Token Embeddings, Segment Embeddings, and the Position Embeddings. As Figure 13 illustrates the input representation is the sum of these three embeddings.


Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}
Figure 13: BERT input representation by (Devlin et al. 2019b)

Token Embeddings: This layer uses WordPiece tokenization to transform words into vector representations of a fixed dimension (Wu et al. 2016). WordPiece tokenization is a data-driven method which is designed to achieve a balance between vocabulary size and out-of-vocab words. Therefore “playing” in Figure 13 was split into “play” and “ing”. Additionally, each tokenized sequence starts with a [CLS] token and between Sentence A and B in a sequence is a [SEP] token.

Segment Embeddings: The [SEP] token allows segment embeddings to differentiate between the two sentences packed into the same token.

Position Embeddings: BERT is based on the transformer architecture, which does not encode the sequential nature of its inputs. For understanding whole sentences such as, “I stood up from the bank to withdraw money from the bank.” it is crucial to differentiate between “bank” at the sixth and the “bank” at the twelfth position. Although they are both the same sequence of letters their representation and thereby their “meaning” in a transformer-based architecture is different.

3.6 Fine-tuning BERT

The self-attention mechanism in BERT allows it to model many down-stream tasks. For each task, the task-specific inputs and outputs are plugged into BERT and all the parameters get fine-tuned end-to-end. Compared to pre-training, fine-tuning is relatively inexpensive (Devlin et al. 2019b).

3.7 Overview of Bert Models

(Devlin 2019a)’s goal is “*to enable research in institutions with fewer computational resources and encourage the community to seek directions of innovation alternative to increasing model capacity.”* Therefore, the team is eager to publish new and more specialized models. The three most relevant for this thesis are:

- ***BERT**$_{base}$* Comparable in size to the OpenAI Transformer
- ***BERT**$_{large}$* Huge Model which achieves state-of-the-art results
- ***BERT**$_{multilinugal}$* is trained on the Wikipedia dumps of the 100 languages with the largest Wikipediaes, which includes Latin.

4 BertSum

The previously introduced BERT model is tested and has excelled on multiple NLP tasks such as question-answering (QA) and general language understanding. However (Devlin et al. 2019b) left the field of text summarization uncovered in their experiments. (Liu 2019)'s experiments with BERT on text summarization led to a new variant or extension of BERT: BertSum. BertSum is specialized on text summarization and comes in two versions:

1. BertSum fine-tuned for extractive summarization (Liu 2019)
2. BertSum fine-tuned for summarization (extractive and abstractive) (Liu and Lapata 2019)

The second version can be understood as an extension of the first. It offers the additional possibility of generating abstractive summaries. But at the same time, it offers fewer possibilities to change parameters on the extractive part. Because the use case of this thesis is generating extractive summaries, BertSum fine-tuned for extractive summarization (Liu 2019) was chosen as the foundation of the experiments.

4.1 Extractive Summarization

Extractive Summaries are a concatenation of the most import spans of the text. Abstractive Summaries, on the other hand, also contain phrases and words not used in the original text (Liu 2019). More technically: Extractive Summarization can be defined as the task of assigning a label $Y_i \epsilon \{0,1\}$ to each sentence $sent_i$ of a document d. Yi indicates whether the sentence should be included in the summarization (Liu and Lapata 2019).

4.2 Extractive Summarization with BERT

Because BERT is trained as a masked-language model (please see chapter 3.3) the output vectors are token-based. In order to apply it to summarization tasks it needs to be sentence-based, because the content of a text is not only based on words but on their interaction in the form of sentences. Another issue is that BERT differentiates sentences based only on two labels (sentence A or sentence B) for segmentation embeddings (please see chapter 3.4). Therefore (Liu 2019) modifies the input sequence and embeddings of BERT to make its application to text summarization problems possible.

Encoding Multiple Sentences**:** As illustrated in Figure 14 and Figure 15 (Liu 2019) inserts a [CLS] token before each sentence and a [SEP] token after each sentence . As explained in chapter 0 (Devlin et al. 2019b) uses the [CLS] at the start of each tokenized sequence (a sentence or a pair sentences). By inserting a [CLS] token at the start of each sentence instead of each tokenized sequence the [CLS] represents individual sentences, allowing it to collect features from its preceding sentence.

Interval Segment Embeddings: For distinguishing different sentences within a document (Liu 2019) uses interval segment embeddings. A segment embedding E_A or E_B is assigned to each sentence $sent_i$ depending if on i is odd or even. See Table 5 for an example.

$sent_1$	$sent_2$	$sent_3$	$sent_4$	$sent_5$
E_A	E_B	E_A	E_B	E_A

Table 5: Example of interval segment embeddings

This allows document representations to be learned hierarchically where lower transformer layers represent adjacent sentences or in other words neighboring sentences. While higher transformer layers, in combination with self-attention, represent multi-sentence discourse (Liu and Lapata 2019). So, higher transformer layers have a more complex or higher understanding of the text

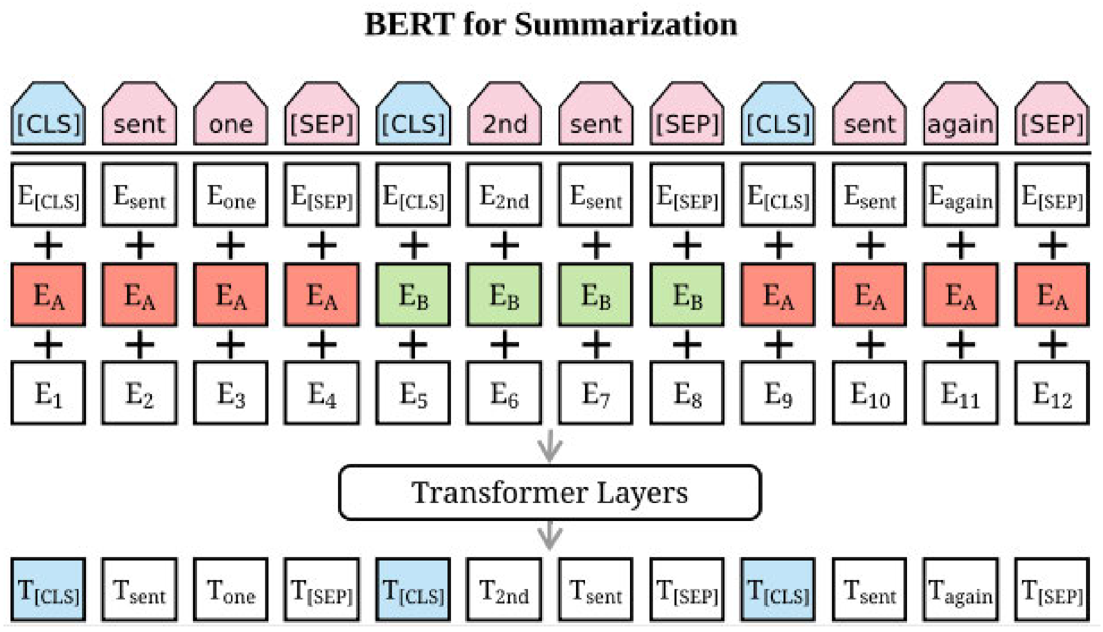

BERT for Summarization
[CLS]
sent
one
[SEP]
[CLS]
2nd
sent
[SEP]
[CLS]
sent
again
[SEP]
E[CLS] Esent Eone E[SEP] E[CLS] E2nd Esent E[SEP] E[CLS] Esent Eagain E[SEP]
EA EA EA EA EB EB EB EB EA EA EA EA
E1 E2 E3 E4 E5 E6 E7 E8 E9 E10 E11 E12
Transformer Layers
T[CLS] Tsent Tone T[SEP] T[CLS] T2nd Tsent T[SEP] T[CLS] Tsent Tagain T[SEP]

Figure 14: Bert for Summarization by (Liu and Lapata 2019)

4.3 Fine-tuning with Summarization Layers

The BERT layer produces a sentence vector T_i of the i-th [CLS] symbol from the top BERT. T_i will be used as the representation for $sent_i$. (Liu 2019) built several summarization-specific layers stacked on top of these outputs. These are jointly fine-tuned with BERT and used to capture document-level features for extracting summaries. The final predicted score $\hat{Y}_i$ will be calculated for each sentence $sent_i$. The whole model's loss is the Binary Classification Entropy of $\hat{Y}_i$ against gold label Y_i. Entropy can be described as the informativeness of a feature.

4.3.1 Simple Classifier

The Simple Classifier of (Liu 2019) only adds one linear layer on the BERT outputs and, as with the original BERT (Devlin et al. 2019b), uses a sigmoid function σ to get the predicted score:

$$\hat{Y}_i = \sigma(W_o T_i + b_o)$$

Equation 6

4.3.2 Inter-sentence Transformer

Instead of a simple sigmoid classifier, the inter-sentence transformer applies more transformer layers only on sentence representations to make it more efficient. These layers help to better understand the important points of the text by extracting document-level features:

$$\tilde{h}^l = ln\left(h^{l-1} + MHAtt\left(h^{l-1}\right)\right)$$

Equation 7: (Liu 2019)

$$h^l = ln\left(\tilde{h}^l + FFN\left(\tilde{h}^l\right)\right)$$

Equation 8: (Liu 2019)

where $h^0 = PosEmb(T)$ and T are the sentence vectors output by BERT and $PosEmb$ is the function of adding positional embeddings (indicating the position of each sentence) to T. (Liu 2019) ln is the layer normalization operation, which is a modification of batch normalization. To overcome batch normalization's dependency on the mini-batch, (Ba et al. 2016) introduced layer normalization. Layer normalization is directly applicable to RNNs. It substantially reduces the training time and is very effective at stabilizing the hidden state dynamics. $MHAtt$ is the multi-head attention operation. It is part of the Transformer architecture by (Vaswani et al. 2017). For more information see chapter 2.5.2.

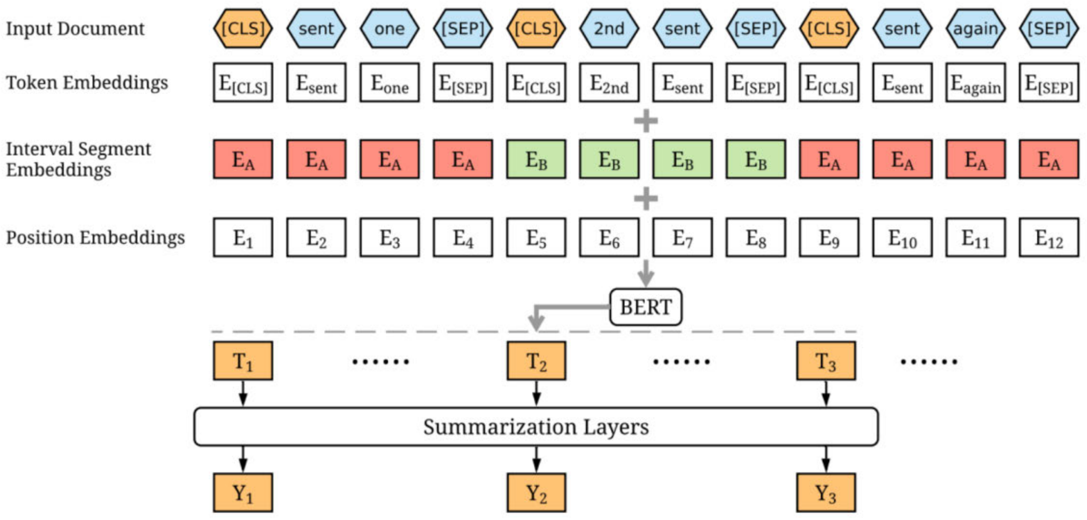


Figure 15: BERTSum Architecture

4.3.3 Recurrent Neural Network

(Chen et al. 2018) showed that recurrent neural networks (RNN) still have their benefits, especially in combination with a transformer, and LSTMs are the most common kind of RNN. So, an LSTM layer applied to the Bert outputs could improve the capabilities of learning summarization-specific features. Pergate layer normalization (Ba et al. 2016) is used within each LSTM cell to stabilize the training. At time step i, the input to the LSTM layer is the BERT output Ti, and the output is calculated as:

$$\begin{pmatrix} F_i \\ I_i \\ O_i \\ G_i \end{pmatrix} = LN_h(W_h h_{i-1}) + LN_x(W_x T_i)$$

Equation 9

$$C_i = \sigma(F_i) \odot C_{i-1} + \sigma(I_i) \odot tanh(G_{i-1})$$

Equation 10

$$h_i = \sigma(O_t) \odot tanh\big(LN_c(C_t)\big)$$

Equation 11

where F_i, I_i, O_i are forget gates, input gates and output gates respectively; G_i is the hidden vector and C_i is the memory vector; hi is the output vector; LN_h, LN_x, LN_c are the different layer-normalization operations; Bias terms are not shown. The final output layer is also a sigmoid classifier:

$$\hat{Y}_i = \sigma(W_o h_i + b_o)$$

Equation 12

5 Experiments

The previous chapters presented a neural machine summarization approach: BertSum. In this section four experiments are conducted to find out how well BertSum can be applied to summarizing medieval Latin texts. The ***first experiment*** shows the possible difference between using a language specific BERT model or the mulitilingual variant of BERT as the pre-trained model. The ***second experiment*** applies BertSum on summarizing German texts, which have been machine-translated from Latin texts. The ***third experiment*** summarizes the original Latin texts to avoid possible errors regarding the content which could be caused by the translation. The ***fourth experiment*** is like the second one but enhanced by applying regularization. In the experiments a few (hyper-)parameters in the architecture and for training the model will be changed to find out which changes could have a positive effect on the output. To make the experiments comparable, every one of them will have 3 tables. The first describes the parameter set up. The second lists the Rouge Score results for the three different flavors: classifier, transformer, and LSTM (RNN). Followed by an example text and another table. This table consists of the gold standard (the original man-made summary) and a candidate (a machine created summary) for each of the three flavors. There will also be a brief written overview about the changes made to the specific parameter. For further information please see the corresponding passages:

- ***Translation***: Which approaches were chosen for the experiment. Please see Chapter 5.2.2
- ***NLP Library***: Which NLP library and language model was used for processing the text. Please see Chapter 5.2.3
- ***Pretrained Model***: Which pretrained model was used. Please see Chapter 3.7

5.1 The dataset: Formulae corpus

The dataset is part of the Formulae - Litterae - Chartae project[2] of the Academy of Sciences and Humanities in Hamburg and the University of Hamburg. The goals of the project are to research and to critically edit early medieval Formulae. Researchers have very few documents from the early middle ages and the formulae collections include texts which normally have not been handed down in the archives. It is a dynamically developing dataset. The experiments were run on the version from 8/17/2020[3] . At that time, the dataset contained 10462 Latin texts, of which 6474 have a German regest (a summary). A selection

[2]https://www.formulae.uni-hamburg.de/das-projekt.html (10/20/2020)
[3]https://github.com/Formulae-Litterae-Chartae/formulae-corpora/tree/5795f416dcacde9d3a0c99b2921283fe62472cf3 (10/20/2020)

of these texts is openly published[4] but most of the texts are not openly available due to copyright restrictions. The experiments will be conducted on all texts that have a German regest. It will be referred as ***FORMULAEGR***.

5.2 Implementation details

The foundational parameters of this setup are unchanged to (Liu 2019)'s original setup: PyTorch, OpenNMT (Klein et al. 2017) and a Pretrained Model are used to implement the model. BERT and summarization layers are jointly fine-tuned. Adam with β1 = 0.9, β2 = 0.999 is used for fine-tuning.

5.2.1 Creating .story Files from the raw Data

(Liu 2019)'s BertSum used .story files as input for the training. The corpus is stored in .xml files and needed to be converted to .story files. Each text has its own .xml file containing various kinds of information. For the experiments only two parts of the .xml file are important: the textpart and the regest. The textpart contains a tokenized form of the original text. Each token of the textpart will be joined, with a blank space in between, to one string. The regest is already one text. The textpart string will now be written to a .story file with the same name as the original .xml file, followed by two line breaks, the "@highlight" marker, then two more line breaks and finally the regest. For further implementation details see xml_story_converter.py.
An example of a .story file with an automatically translated Germen text might look like this:
urn.cts.formulae.echternach.wampach0065.story:

```
„
Der Herr, dem ehrwürdigen Vater in Christus, Adebertus, dem Abt des Klosters Epternaco von Baba, der Tochter des Herina-Gebers, aus Angst vor Gott und zur Errettung ihrer Seele, gibt jeder hier an dem oben genannten Ort seiner Güter an einem Ort im Dorf Serinse an dem Ort, der Doffeningen genannt wird, das heißt die beiden Länder, die Felder, die Wiesen, die Wiesen , Lavendel, Bauernhöfe dort, aquarumve fließendes Wasser und was auch immer ich dort gesehen habe. Wenn einer der Erben - wie oben. 2 Jahre Regierungszeit hat der König freundschaftlich.

@highlight

baba schenkt kloster echternacht ihre güter zu doffeningen im serinsergau ( saargau ? )
„
```

5.2.2 Adding a Translation layer

Regests are usually written in the language of the edition in which they are published. In our corpus the corresponding texts are written in Latin and their regests are in German. To apply BertSum to this scenario a translation layer is needed.
One could research the following approaches:

[4] https://werkstatt.formulae.uni-hamburg.de/(10/20/2020)

- ***Input-To-Target-Translation***: Translating the text to the language of the targeted summarization language before it gets tokenized.
- ***Target-To-Input-Translation:*** Translating the targets (i.e., the regests) to the language of the source text. Train the model on this data.

For all experiments I used the Google Translator Python package[5]. The chosen translator was occasionally unable to produce any translation and ran into a "translation error" (similar problem described in: [6]). These errors indicate that the translator had problems translating the text as one semantic unit. The texts were written in a time without any fixed orthographic rules. To help the translator translate as much of the text as possible a translation using a batches mechanism was introduced. It first tries to separate the text into batches with "." as separator. These batches are like sentences. Many of the translation errors could already be solved with this separator. Secondly the same procedure is repeated with "," as separator. There are a few texts with enumerations or sentences which are not separated by "." which could be translated by this approach. Thirdly the approach is repeated with " " (blank space) as separator. There were only very few texts left untranslated after the first two attempts and these were then translated word-by-word.

5.2.3 Changing the NLP Library

The original BertSum used the Stanford Core NLP Library[7] [8] which only supports English, Arabic, Chinese, French, German, and Spanish. Additionally, it is not trivial to use it in Python applications or to change the language. So, the author decided to use another NLP Library from Stanford: Stanza[9]. This library has an installable Python package and can create results in the same format as Stanford Core NLP does. It also has the additional benefit that it is very easy to switch between languages and their specific models. It supports over 80 different languages including Latin. For Latin tokenization, the Latin model with the ittb[10] package is used. It is the largest Latin package, and it is trained on medieval Latin texts. For German tokenization, the German model with the gsd[11] package is used. It is the largest German package.

5.2.4 Hyperparameters

The goal of the experiments is to apply BertSum to a new dataset. Therefore, it was attempted to have the same relation of the overall number of documents to the other hyperparameters to better compare the results. The only major deviation from the rule is the batch size. It is set to 960 because otherwise the resulting number would seem to be too

[5] https://pypi.org/project/googletrans (10/20/2020)
[6] https://support.google.com/translate/thread/24431170?hl=en (10/20/2020)
[7] https://stanfordnlp.github.io/CoreNLP/ (10/20/2020)
[8] https://github.com/nlpyang/BertSum/issues/13#issuecomment-484087833 (10/20/2020)
[9] https://stanfordnlp.github.io/stanza/ (10/20/2020)
[10] https://universaldependencies.org/treebanks/la_ittb/index.html (10/20/2020)
[11] https://universaldependencies.org/treebanks/de_gsd/index.html (10/20/2020)

small [12] [13]. Table 6 and Table 8 show the setup of (Liu 2019). Table 7 and Table 8 show the setup for this thesis.

	Total	Training	Validation	Testing
CNN		90,266	1,220	1,093
DM		196,961	12,148	10,397
CNNDM	312,085	287,227 (~92%)	13,368 (~4%)	11,490 (~4%)

Table 6: Data splitting of (Liu 2019)

	Total	Training	Validation	Testing
FORMULAEGER	4777	4395 (~92%)	191 (~4%)	191 (~4%)

Table 7: Data splitting of the FORMULAEGER corpus

	CNNDM		**FORMULAEGER**	
	absolute number	**factor to number of training documents**	**absolute number**	**factor to number of training documents**
Training Documents	287,277		4395	
batch size	3000	0,01	960	0,22
train steps	5000	0,16	700	0,16
warm up	10000	0,03	135	0,03

Table 8: Hyperparamters in comparison

5.2.5 Discussion

(Popel and Bojar 2018) investigated the effect of hyperparameters like learning rate and batch size on neural machine translation with transformers. Since neural machine translation is a sequence-to-sequence problem quite like summarization, and since BertSum is also based on the transformer architecture their findings could be useful to determine the hyperparameters for the following experiments. (Popel and Bojar 2018, p. 59) recommend setting the batch size as high as possible and establishing the largest possible batch size before starting the main training. Their work indicates that larger batch sizes result in better BLEU (bilingual evaluation understudy) scores of the results. So, it might be interesting to further improve the batch size.

5.2.6 Learning rate

(Goodfellow et al. 2016, pp. 82–86) write that most deep learning algorithms use optimization to improve their results. Optimization is the task of trying to maximize or

[12] https://github.com/nlpyang/BertSum/issues/44#issuecomment-501161474 (10/20/2020)
[13] https://github.com/nlpyang/BertSum/issues/33#issuecomment-494850040 (10/20/2020)

minimize a function f(x). In most cases the application tries to minimize it. The corresponding function is then called loss, error, or cost function. To come closer to minimum, f(x) gets iteratively decreased. This technique is called ***gradient descent***. The value by which f(x) is decreased is determined by the ***learning rate***. There are different ways to determine the learning rate. (Liu 2019) uses the learning rate proposed by (Vaswani et al. 2017) with warming-up on first 10,000 steps:

$$lr = 2e^{-3} \cdot min\,(step^{-0.5}, step \cdot warmup^{-1.5})$$

Equation 13: Learning Rate

The learning rate is usually a small number. If the learning rate is too big, minima could be overstepped and if it is too small the training time will increase drastically. This scheme or decay method is firstly applied in (Vaswani et al. 2017) and is often called **noam decay** after its author. It is basically a linear warmup followed by an exponential decay. [14]

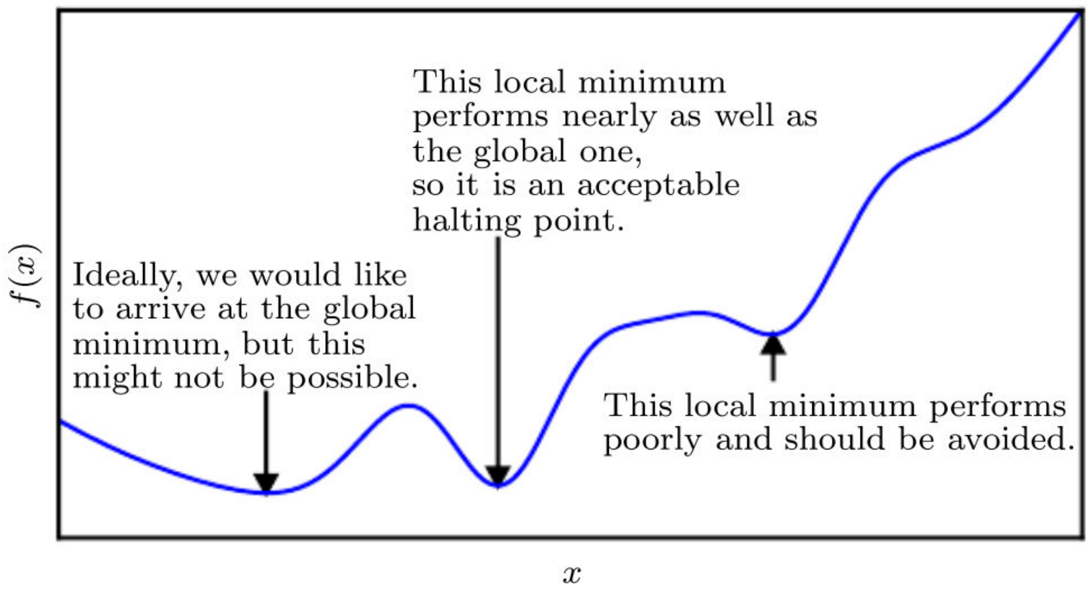


Figure 16: Learning rate picture by (Goodfellow et al. 2016, p. 85)

5.2.7 Regularization

The experiments are based on a rather small corpus and the results of the classifier, transformer and RNN look in many cases the same. This indicates that overfitting could be a problem. Overfitting occurs when the gap between the training error and test error is too large. In comparison: underfitting occurs when the model is not able to obtain a sufficiently low error value on the training set. Whether a model is more likely to over- or to underfit can be regulated by adjusting its capacity. Models with low capacity may have problems to fit the training set. Models with a high capacity can overfit by memorizing parts of the training data,

[14] https://github.com/tensorflow/tensor2tensor/issues/280#issuecomment-359477755 (10/20/2020)

which can negatively influence the result on the test set (Goodfellow et al. 2016, p. 111). Figure 17 depicts examples of different capacity characteristics. Any modification applied to a learning algorithm that is intended to reduce its generalization error but not its training error is called regularization (Goodfellow et al. 2016, p. 120). (Peng et al. 2015) showed that ***L2 regularization*** (Goodfellow et al. 2016, pp. 231–234) could help to generalize the model especially when smaller data sets are used. So, L2 regularization could help improve the model's results. It modifies the training criterion for linear regression to include weight decay. It is implemented as described in the documentation[15].

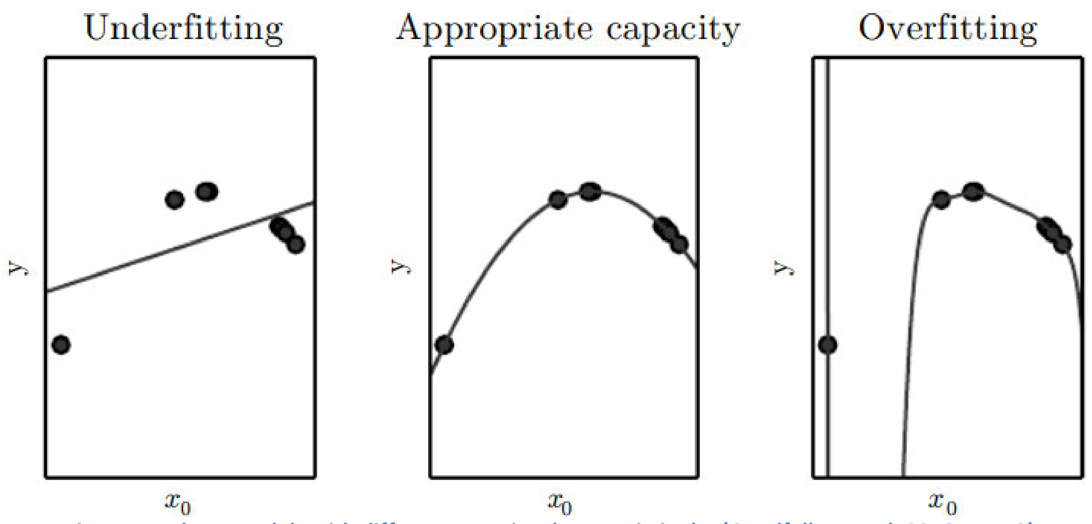


Figure 17: Three models with different capacity characteristics by (Goodfellow et al. 2016, p. 113)

5.2.8 Trigram Blocking

(Liu 2019) used Trigram blocking to reduce redundancy. If a summary and a candidate have an overlapping trigram, the candidate will be skipped.

5.3 Evaluation Method: ROUGE

For evaluating the experimental results, the ROUGE metric will be used. ROUGE or Recall-Oriented Understudy for Gisting Evaluation was introduced by (Lin 2004). It measures the overlap of n-grams between a candidate and a set of reference summaries. ***ROUGE-1*** refers to the overlap of unigrams (each word) and ***ROUGE-2*** to the overlap of bigrams. ***ROUGE-N***, the generalization of ROUGE-1 and ROUGE-2, is calculated as follows:

$$\text{ROUGE-N} = \sum_{S\in\{ReferenceSummaries\}} \frac{\sum_{gram_n\in S} Count_{match}(gram_n)}{\sum_{S\in\{ReferenceSummaries\}} \sum_{gram_n\in S} Count(gram_n)}$$

Equation 14

[15] https://pytorch.org/docs/stable/optim.html#torch.optim.Adam (10/20/2020)

Where n stands for the length of the n-gram, $gram_n$, and $Count_{match}(gram_n)$ is the maximum number of n-grams co-occurring in a candidate summary and a set of reference summaries. ***ROUGE-L*** refers to the Longest Common Subsequence (LCS). It includes sentence level structure similarity and detects the longest co-occurring sequence of n-grams. To see how ROUGE-L can work reliably at the sentence level consider the following example:

reference sentence (S1)	*police killed the gunman*
candidate sentence (S2)	police kill the gunman
candidate sentence (S3)	the gunman kill police

Table 9: ROUGE-2 and ROUGE-L Example from (Lin 2004)

The ROUGE-2 score for S2 and S3 would be the same because they have one bigram "the gunman", although S2 and S3 have very contrasting meanings. But in the case of ROUGE-L, S2 has a score of 0.75 (3/4) because the complete sequence is four words long and three words of the four word are equal to S1 ("police the gunman"). In contrast only "the gunman" (2/4) in S3 is equal to S1 which results in 0.5 as score for S3. Therefore ROUGE-L would classify S2 to be better than S3, just as a human reader would.

5.4 Presenting the Evaluation

Each experiment has a table similar to (Liu 2019)s Table 1. The information of this table is included in Table 10. This form of representation allows the results to be compared to the original paper. Unlike (Liu 2019), the scores are not the average of the best three steps but one specific selected step. This makes it possible to compare the result at the specific step. The difference between the average and one specific step is marginal. For instance the difference of BERTSum + Transformer is 8 in Table 17 and 8.03 Table 18 so the difference is only 0.03 or 0.4%. Table 17 and Table 18 in chapter 5.7 show the scores for both ways of calculating it.

5.5 Training BERTSum with pretrained bert-multilingual on the original BERTSum Data

This experiment should illustrate the differences between a model with a fitting monolingual pretrained model and a model with a multilingual model and to show the advantage of monolingual models over multilingual ones. The original BertSum used a monolingual model and My BertSum used a multilingual model.

Model	**Rouge-1**	**Rouge-2**	**Rouge-L**
Original BERTSum + Classifier (Liu 2019)	43.23	20.22	39.60
My BERTSum + Classifier (Step 4900)	42.5	19.61	38.94
Original BERTSum + Transformer (Liu 2019)	43.25	20.24	39.63
My BERTSum + Transformer (Step 4900)	40.80	18.32	37.22
Original BERTSum + LSTM (Liu 2019)	43.22	20.17	39.59

My BERTSum + LSTM (Step 4900)	42.47	19.59	38.91

Table 10: Test set results on the CNN/DailyMail dataset using ROUGE F_1

5.6 Training BERTSum with pretrained bert-multilingual on FORMULAEGR with Input-To-Target-Translation

The goal of this experiment is to produce German summaries from Latin texts. It is done by first translating the Latin text to German. Afterwards the text is tokenized and trained with multilingual pre-trained model.

Translation	**NLP Library**	**Pretrained Model**
Input-To-Target-Translation	Stanza: German, GSD	bert-base-multilingual-cased

Table 11: Experiment parameters

training	**testing**	**validation**	**batch**	**Steps**	**Warmup**	**checkpoint**	**RNN size**
4395	191	191	960	700	135	100	768

Table 12: Experiment hyperparameters

Model	**Rouge-1**	**Rouge-2**	**Rouge-L**
BERTSum + Classifier	9.11	0.71	7.07
BERTSum + Transformer	9.41	0.77	7.32
BERTSum + LSTM	9.39	0.76	7.29

Table 13: Test set results of the first experiment at step 700 using ROUGE F_1

The source Text (urn.cts.formulae.echternach.wampach0065[16]) (translated using the translationlayer):
„Der Herr, dem ehrwürdigen Vater in Christus, Adebertus, dem Abt des Klosters Epternaco von Baba, der Tochter des Herina-Gebers, aus Angst vor Gott und zur Errettung ihrer Seele, gibt jeder hier an dem oben genannten Ort seiner Güter an einem Ort im Dorf Serinse an dem Ort, der Doffeningen genannt wird, das heißt die beiden Länder, die Felder, die Wiesen, die Wiesen , Lavendel, Bauernhöfe dort, aquarumve fließendes Wasser und was auch immer ich dort gesehen habe. Wenn einer der Erben - wie oben. 2 Jahre Regierungszeit hat der König freundschaftlich.“

Its corresponding Latin Original:
Domino venerabili in Christo patri Adeberto abbati de monasterio Epternaco Baba filia Herini donatrix, pro Dei timore et pro remedio anime sue donat ad prefatum locum res suas in pago Serinse in loco qui dicitur Doffeningen, id est tam terris, campis, pratis, pascuis, silvis, casis,

[16] Geschichte der Grundherrschaft Echternach im Frühmittelalter Nr. 65, in: Camille Wampach, Geschichte der Grundherrschaft Echternach im Frühmittelalter, Luxemburg 1929, [URI: http://d–nb.info/368617769], S. 128–129.)

mansis, aquis aquarumve decursibus et quicquid ibi visa sum habere. Si quis vero de heredibus meis – ut supra. Actum anno II regnante Karlomanno rege.

gold	baba schenkt kloster echternacht ihre güter zu doffeningen im serinsergau (saargau ?)
classifier	der herr , dem ehrwürdigen vater in christus , adebertus , dem abt des klosters epternaco von baba , der tochter des herina - gebers , aus angst vor gott und zur errettung ihrer seele , gibt jeder hier an dem oben genannten ort seiner güter an einem ort im dorf serinse an dem ort , der doffeningen genannt wird , das heißt die beiden länder , die felder , die wiesen , die wiesen , lavendel , bauernhöfe dort , aquarumve fließendes wasser und was auch immer ich dort gesehen habe .<q>2 jahre regierungszeit hat der könig freundschaftlich .<q>wenn einer der erben - wie oben .
transformer	der herr , dem ehrwürdigen vater in christus , adebertus , dem abt des klosters epternaco von baba , der tochter des herina - gebers , aus angst vor gott und zur errettung ihrer seele , gibt jeder hier an dem oben genannten ort seiner güter an einem ort im dorf serinse an dem ort , der doffeningen genannt wird , das heißt die beiden länder , die felder , die wiesen , die wiesen , lavendel , bauernhöfe dort , aquarumve fließendes wasser und was auch immer ich dort gesehen habe .<q>wenn einer der erben - wie oben .<q>2 jahre regierungszeit hat der könig freundschaftlich .
RNN	der herr , dem ehrwürdigen vater in christus , adebertus , dem abt des klosters epternaco von baba , der tochter des herina - gebers , aus angst vor gott und zur errettung ihrer seele , gibt jeder hier an dem oben genannten ort seiner güter an einem ort im dorf serinse an dem ort , der doffeningen genannt wird , das heißt die beiden länder , die felder , die wiesen , die wiesen , lavendel , bauernhöfe dort , aquarumve fließendes wasser und was auch immer ich dort gesehen habe .<q>2 jahre regierungszeit hat der könig freundschaftlich .<q>wenn einer der erben - wie oben .

Table 14: Gold vs Candidate Step 700 no 33

5.7 Training BERTSum with pretrained bert-multilingual on FORMULAEGR with Input-To-Target-Translation

The goal of this experiment is to see if the setup performs better when the summaries, on which the model should learn, are translated to Latin. Thereby the results of the model are also in Latin.

Translation	NLP Library	Pretrained Model
Input-To-Target-Translation	Stanza: German, GSD	bert-base-multilingual-cased

Table 15: Experiment parameters

training	testing	validation	batch	Steps	Warmup	checkpoint	rnn size

4395	191	191	960	700	135	100	768

Table 16: Experiment hyperparameters

Model	**Rouge-1**	**Rouge-2**	**Rouge-L**
BERTSum + Classifier	7.44	0.31	6.03
BERTSum + Transformer	8.03	0.34	6.52
BERTSum + LSTM	7.86	0.33	6.42

Table 17: Test set results of the second experiment using the average of the three best results ROUGE F_1

Model	**Rouge-1**	**Rouge-2**	**Rouge-L**
BERTSum + Classifier	7.61	0.33	6.14
BERTSum + Transformer	8	0.34	6.51
BERTSum + LSTM	7.89	0.34	6.45

Table 18: Test set results of the second experiment at step 400 using ROUGE F_1

The source Text (urn.cts.formulae.fulda_stengel.stengel0042[17]):
„Venerabili in Christo patri Sturmione abbati. Ego Eggiolt cogitans pro remedium anime fratris mei Hiltwini vel pro aeterna retributione, propterea dono ac trado traditumque in perpetuum esse volo a die presente ad monasterium sancti salvatoris, quod dicitur Fulda, ubi tu presenti tempore abbas preesse videris, hoc est quod dono in villa Buchsolare a in pago Wormacinse duas curtiles cum casis, cum terris ibidem adiacentibus et vineis seu pratis vel silvis in loco, qui dicitur Ascae, atque in Wibileschiricha ad ipsas curtiles aspicientibus, id est pratis, pascuis, aquis aquarumque decursibus, sicut superius dixi, pro anime Liutwini donatum esse volo. Propterea donationem rogavi scribere, ut tu, abba Sturmi, supra scriptam rem et successores tui habeant, teneant, possideant vel quicquid exinde facere voluerint, liberam ac firmissimam in omnibus habeant potestatem. Si quis vero, quod futurum esse non credo, si ego ipse aut aliquis de heredibus meis vel proheredibus meis aut ulla opposita persona extranea, qui contra hanc kartolam donationis meae, quam propter nomen domini fieri rogavi vel pro reverentia ipsius loci, venire aut eam infrangere conatus fuerit, iram trine maiestatis incurrat et cum supra scripto sancto illo ante tribunal Christi deducat rationes, insuper autem inferat partibus ipsius monasterii dupla pecunia, quantum ista res in se conteneat, et quod repetit evindicare non valeat, sed presens epistola hec omni tempore firma et inviolata permaneat, stipulatione subnexa. Actum Wangiona civitate publice sub die V. idus mai. Anno XIIII. regnante domno nostro Pippino gloriosissimo rege. † Signum Aggioldi, qui hanc donationem fieri rogavit. † Altoni. † Odalfridi. † Hruotfridi. † Gummundi. † Reginfridi. † Adalmanni. † Nordmanni †. Ego Hiaebo presbiter et amanuensis hanc kartolam donationis rogante Aggioldo scripsi.“

[17] Urkundenbuch des Klosters Fulda; Teil: Bd. 1., Nr. 42, in: Edmund E. Stengel, Urkundenbuch des Klosters Fulda; Teil: Bd. 1., (Die Zeit der Äbte Sturmi und Baugulf), Marburg 1958, [URI: http://d-nb.info/454869509], S. 72-73

gold	eggiolt (aggioldus) ponit duas villas monasterii bossweiler (bockenheim) wiebelskirchen saluti et sociis ejus proprietates ascae hiltwin (liutwin) sturmi penes abbatem .
classifier	venerabili in christo patri sturmione abbati .<q>ego eggiolt cogitans pro remedium anime fratris mei hiltwini vel pro aeterna retributione , propterea dono ac trado traditumque in perpetuum esse volo a die presente ad monasterium sancti salvatoris , quod dicitur fulda , ubi tu presenti tempore abbas preesse videris , hoc est quod dono in villa buchsolare a in pago wormacinse duas curtiles cum casis , cum terris ibidem adiacentibus et vineis seu pratis vel silvis in loco , qui dicitur ascae , atque in wibileschiricha ad ipsas curtiles aspicientibus , id est pratis , pascuis , aquis aquarumque decursibus , sicut superius dixi , pro anime liutwini donatum esse volo .<q>propterea donationem rogavi scribere , ut tu , abba sturmi , supra scriptam rem et successores tui habeant , teneant , possideant vel quicquid exinde facere voluerint , liberam ac firmissimam in omnibus habeant potestatem .
transformer	venerabili in christo patri sturmione abbati .<q>ego eggiolt cogitans pro remedium anime fratris mei hiltwini vel pro aeterna retributione , propterea dono ac trado traditumque in perpetuum esse volo a die presente ad monasterium sancti salvatoris , quod dicitur fulda , ubi tu presenti tempore abbas preesse videris , hoc est quod dono in villa buchsolare a in pago wormacinse duas curtiles cum casis , cum terris ibidem adiacentibus et vineis seu pratis vel silvis in loco , qui dicitur ascae , atque in wibileschiricha ad ipsas curtiles aspicientibus , id est pratis , pascuis , aquis aquarumque decursibus , sicut superius dixi , pro anime liutwini donatum esse volo .<q>propterea donationem rogavi scribere , ut tu , abba sturmi , supra scriptam rem et successores tui habeant , teneant , possideant vel quicquid exinde facere voluerint , liberam ac firmissimam in omnibus habeant potestatem .
RNN	ego eggiolt cogitans pro remedium anime fratris mei hiltwini vel pro aeterna retributione , propterea dono ac trado traditumque in perpetuum esse volo a die presente ad monasterium sancti salvatoris , quod dicitur fulda , ubi tu presenti tempore abbas preesse videris , hoc est quod dono in villa buchsolare a in pago wormacinse duas curtiles cum casis , cum terris ibidem adiacentibus et vineis seu pratis vel silvis in loco , qui dicitur ascae , atque in wibileschiricha ad ipsas curtiles aspicientibus , id est pratis , pascuis , aquis aquarumque decursibus , sicut superius dixi , pro anime liutwini donatum esse volo .<q>venerabili in christo patri sturmione abbati .<q>si quis vero , quod futurum esse non credo , si ego ipse aut aliquis de heredibus meis vel proheredibus meis aut ulla opposita persona extranea , qui contra hanc kartolam donationis meae , quam propter nomen domini fieri rogavi vel pro reverentia ipsius loci , venire aut eam infrangere conatus fuerit , iram trine maiestatis incurrat et cum supra scripto sancto illo ante tribunal christi deducat rationes , insuper autem inferat partibus ipsius

	monasterii dupla pecunia , quantum ista res in se conteneat , et quod repetit evindicare non valeat , sed presens epistola hec omni tempore firma et inviolata permaneat , stipulatione subnexa .

Table 19: Gold vs Candidate Step 400 no 33

5.8 Training BERTSum with pretrained bert-multilingual on FORMULAEGR with Input-To-Target-Translation with L2 regularization

This experiment follows the same approach as the one in chapter 5.6. The results in Table 14 and the fact the model is trained on a rather small data set indicate that overfitting could be a problem. To solve the overfitting-issue L^2 regularization with a weight decay of 0.00001 is applied in this experiment. The Rouge score improvements of Table 22 in comparison to Table 13 showed that applying regularization has indeed a positive effect on the results.

Translation	**NLP Library**	**Pretrained Model**
Input-To-Target-Translation	Stanza: German, GSD	bert-base-multilingual-cased

Table 20: Experiment Parameters

training	**testing**	**validation**	**batch**	**Steps**	**Warmup**	**checkpoint**	**rnn size**	**weight decay**
4395	191	191	960	700	135	100	768	

Table 21: Experiment Hyperparameters

Model	**Rouge-1**	**Rouge-2**	**Rouge-L**
BERTSum + Classifier	9.42	0.80	7.53
BERTSum + Transformer	9.76	0.92	7.62
BERTSum + LSTM	9.70	0.92	7.56

Table 22: Test set results of the first experiment at step 700 using ROUGE F_1

The source Text (urn.cts.formulae.freising.bitterauf0427[18]):
„Sei es allen bekannt und bleibe in den Baiouuaria tamingenuis als denen, die gedient haben, aber kann dies daher einer seiner eigenen Äbte sein, der Name des schweren Alodeoratoriums Salomonconstruxit ihres Vaters an dem Ort, den er für oder am Fluss des Edikts des Sindpaldeshusir Filusa vorbereitet hat. Wegen eines frommen und mansuetumHittonem quamperficere kam ein Bischof zum Gebetshaus, Salomo erwähnte demütig inplorando bereits, soweit er an derselben Stelle sepollicere aignaretur kommen sollte, dass er auf diese Weise der Bischof den gleichen Reeve machte. Deindeveniens Bischof und viele andere Adlige und Adsisterunt.Tunc Ort namens Bischof widmen diese Versammlung sowie die Verteidigung Altareminsuper hinzugefügt. Als also all diese Dinge erfüllt sind, ist es geschehen, in die Gegenwart des rechtmäßigen, ob es sich um einen

[18] Die Traditionen des Hochstifts Freising, Nr. 427, in: Theodor Bitterauf, Die Traditionen des Hochstifts Freising, Band 1: 744 – 926, München, ND Aalen 1905 1967, S. 366-367

vernünftigen Potestatemissus handelt, den Salomo direkten Kontakt mit dem Namen des Einhart, und dort hatte er ihre Catervaqui dazu beleidigt, die Widmung des siebten und die Kirche aufzugeben, sowie den Etsoror desselben Salomos, mit ihren Teilen oder der gleichen Art und Weise Das Schloss der Website Frigisinense ubipreciosum, aus dem der Leichnam des heiligen korbinischen Beichtvaters Christi besteht, regiert dort den Hitto, den Bischof dieser Familie, der seine Kontrolle mit dem dazugehörigen Omnibusad ausübt, oder an irgendetwas an dem vorgenannten Ort Sindpaldeshusirhereditatis, der für sie gemacht wurde ohne Veränderung, durch den Profit aus dem gleichen Grund, er und ihre Schwester: und der Name der Engilsuind utille, bis zu welchem Punkt in ihrem Leben darauf gerichtet ist, dies zu nutzen, und viele blieben in der Macht, die Fähigkeit zu genießen, zu fallen, ettransitoria; illorumvero nach der Abreise außer nulla Opposition gegen den genannten domoobtinetur. Dies geschah unter dem Konsul, der 7 ist. dh. menne nov.anno Hloduuuico zugunsten des Reiches Christi ersetzt 6. die Abgabe XIII. In den Augen dieser anderen, Johan Archipresbiter. Marcho Priester. Emihho, Nidperht, Diakone. Ring. Erchanpert. Uuolfperht. Podalunc.Strodo. Die Ohren sind von Zeugen angezogen, Ilprant. Sigipald. Eparheri. Rihpald.Cozolt. Petro. Ampricho. Engilhart. Reginhoh. Adalperht. Juto. Arnolt.Drudmunt. Reidker. Frese. Reginperht. Helier. Arpeo. Uuelacrim. Hartrat.Hrodolt und viele andere. Also bestellte ich Pirtilo Hitton episcopiscripsi und unterschrieb.

gold	abt salomon und seine schwester engilsuind übergeben einekirche zu „ sindpaldeshusun“ .
classifier	sei es allen bekannt und bleibe in den baiouuaria tamingenuis als denen , die gedient haben , aber kann dies daher einer seiner eigenen äbte sein , der name des schweren alodeoratoriums salomonconstruxit ihres vaters an dem ort , den er für oder am fluss des edikts des sindpaldeshusir filusa vorbereitet hat .<q>deindeveniens bischof und viele andere adlige und adsisterunt.tunc ort namens bischof widmen diese versammlung sowie die verteidigung altareminsuper hinzugefügt .<q>wegen eines frommen und mansuetumhittonem quamperficere kam ein bischof zum gebetshaus , salomo erwähnte demütig inplorando bereits , soweit er an derselben stelle sepollicere aignaretur kommen sollte , dass er auf diese weise der bischof den gleichen reeve machte .
transformer	die regierungszeit unseres herrn jesus christus , ein jahr xxviiii.regni tassilone führer , begann ich über ultimumdeductus uuaninc krankheit in meinem leben oder dem leben der zukunft nachzudenken , damit ich mich mit dem liebenden meister erkundigen kann .<q>dies ist genau das richtige , wenn man bedenkt , dass ein apudmemet erfunden wurde , als sie das alodem - erbe mir oder meinem bruder überließ , dem mein vater ein utto an dem ort gegeben hatte , der rechpach genannt wird , anstelle von oder an irgendetwas dem montenuncupante an den feldern der weiden der pras angesichts der tatsache , dass alle instrumente der angelegenheiten , die die nosiura betreffen , das laufen , die ich dem beichtvater christi in der kirche st. corbinian und dem grab der burg und der makellosen jungfrau mariae frigisingasite übergeben würde .<q>auf die gleiche weise , wie wir all diese dinge getan haben , die der stil der dinge sind , werden sie

	inpraesentia für ihren vater oder bruder verstanden , und auch , um einen körper von velconiugis und der priester , das heißt den uuicpót , erlapald , hunolt , zu propinquieren , so dass , wenn quiscontra innerhalb dieser tradition ist und versuchen sollte , sich zu bewegen , in dem , in dem sich die namen befinden und eine übertragungsfunktion des rechts von diesen , dennoch könnte ein unternehmen weiterhin bestimmungen abschließen .
RNN	sei es allen bekannt und bleibe in den baiouuaria tamingenuis als denen , die gedient haben , aber kann dies daher einer seiner eigenen äbte sein , der name des schweren alodeoratoriums salomonconstruxit ihres vaters an dem ort , den er für oder am fluss des edikts des sindpaldeshusir filusa vorbereitet hat .<q>wegen eines frommen und mansuetumhittonem quamperficere kam ein bischof zum gebetshaus , salomo erwähnte demütig inplorando bereits , soweit er an derselben stelle sepollicere aignaretur kommen sollte , dass er auf diese weise der bischof den gleichen reeve machte .<q>deindeveniens bischof und viele andere adlige und adsisterunt.tunc ort namens bischof widmen diese versammlung sowie die verteidigung altareminsuper hinzugefügt .

Table 23: Gold vs Candidate Step 200 no 33

6 Conclusion

The best achieved Rouge L_1 score in this thesis is 9.76 which in comparison to (Liu 2019)'s best L_1 score of 43.25 a 430% decrease. This relation shows the general quality decline of this thesis in comparison to (Liu 2019). The two factors, which are probably the most influencing on the results are the similarity of the pre-trained model to the task-specific training data set and the small size of the task-specific training data set. The pre-trained (Multilingual BERT) model was trained on 104 languages among others Latin and German. Although (Pires et al. 2019) showed that Multilingual BERT is robust and very well capable of generalizing cross lingually, it would still have a great impact if the pre-trained model was explicitly trained on German or Latin. The experiment in chapter 5.5 showed what possible difference on the results a language-specific per-trained model could have. Additionally the size of the training data set contained 4777 texts in this thesis and 312,085 in (Liu 2019). This 6500% difference indicates that increasing the amount of training data is probably the greatest leverage to reach the result quality of (Liu 2019). When manually comparing the result of the experiments to its corresponding text the reader is prompted by the fact, that a great part of the summary consists of the first spans of the text. And that important information such as the name of the main character of the document is left out. This shows that the generated summaries are of a rather bad quality. The fact that the generated summaries are almost as long as the original text indicates that much noise or irrelevant information is still included in the summary. Another issue that can be seen in chapter 5.6 and chapter 5.7 is that some candidates of the three different flavors where exact equals. An indicator that the model is likely to be overfitted. The experiment in chapter 5.8 used L_2 regularization as countermeasure for this issue. The results in this chapter had a higher result compared to chapter 5.6 which indicates that further investigating regularization methods, especially with transformer-specialized regularization methods such as the DropHead method (Zhou et al. 2020), seem to be very promising. Furthermore the discussion in chapter 5.2.5 suggests that experimenting with the batch size could also increase the quality of the results. To put it all in a nutshell, the results of this thesis could not be used as a replacement for man-made regests nor as support for human summarizers. But the findings of this thesis can serve as the groundwork for further investigations on the neural machine summarization of medieval latin texts and the neural machine writing of regests.

Acknowledgements

I would like to express my gratitude to my primary supervisor, Marina Tropmann-Frick, who guided me throughout my entire bachelor thesis. Additionally, I would like to thank Matthew Munson, Henry Martens, Angelika Nyenti, Marcel Will and Horst Loesslein for providing me very fruitful feedback. Each one of them shed an individual light on parts of my thesis because of their individual academic background. A special thanks to the Formulae-Litterae-Chartae Team who led me to my research on their laboriously collected data. And to Jan Dalski who helped me run my code on the HAW Hamburgs ICC.

List of Figures

List of Tables

List of Abbreviations

BERT	Bidirectional Encoder Representations from Transformers
BLEU	bilingual evaluation understudy
LSTM	Long Short-term Memory
MLM	Masked Language Model
NLI	Natural Language Inference

NLP	Natural Language Processing
NSP	Next Sentence Prediction
QA	Question Answering
RNN	Recurrent Neural Networks
ROUGE	Recall-Oriented Understudy for Gisting Evaluation

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, den ________________ __ [redacted]

Bibliography

Alammar, Jay: Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models With Attention). Available online at https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/, checked on 5/29/2020.

Alammar, Jay (2018): The Illustrated Transformer, 6/17/2018. Available online at https://jalammar.github.io/illustrated-transformer/, checked on 5/14/2020.

Asadi, Ahmad; Safabakhsh, Reza (2020): The Encoder-Decoder Framework and Its Applications. In : Deep Learning: Concepts and Architectures, pp. 133–167.

Ba, Jimmy Lei; Kiros, Jamie Ryan; Hinton, Geoffrey E. (2016): Layer Normalization. In *arXiv:1607.06450 [cs, stat],* checked on 5/13/2020.

Bahdanau, Dzmitry; Cho, Kyunghyun; Bengio, Yoshua (2016): Neural Machine Translation by Jointly Learning to Align and Translate. In *arXiv:1409.0473 [cs, stat],* checked on 5/16/2020.

Bengio, Yoshua; Ducharme, Réjean; Vincent, Pascal (2000): A Neural Probabilistic Language Model. In *CrossRef Listing of Deleted DOIs* 1 (6), pp. 932–938. DOI: 10.1162/153244303322533223.

Chen, Mia Xu; Firat, Orhan; Bapna, Ankur; Johnson, Melvin; Macherey, Wolfgang; Foster, George et al. (2018): The Best of Both Worlds: Combining Recent Advances in Neural Machine Translation. Available online at https://arxiv.org/pdf/1804.09849.

Cho, Kyunghyun; van Merrienboer, Bart; Gulcehre, Caglar; Bahdanau, Dzmitry; Bougares, Fethi; Schwenk, Holger; Bengio, Yoshua (2014): Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. Available online at https://arxiv.org/pdf/1406.1078.pdf, checked on 5/30/2020.

Devlin, Jacob (2019a): BERT Multilingual. In *GitHub*, 10/17/2019. Available online at https://github.com/google-research/bert/blob/master/multilingual.md, checked on 4/30/2020.

Devlin, Jacob; Chang, Ming-Wei; Lee, Kenton; Toutanova, Kristina (2019b): BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *arXiv:1810.04805 [cs],* checked on 3/29/2020.

Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron (2016): Deep learning. Cambridge, Massachusetts London, England: The MIT Press (Adaptive computation and machine learning).

Jean, Sébastien; Cho, Kyunghyun; Memisevic, Roland; Bengio, Yoshua (2014): On Using Very Large Target Vocabulary for Neural Machine Translation. Available online at https://arxiv.org/pdf/1412.2007.

Klein, Guillaume; Kim, Yoon; Deng, Yuntian; Senellart, Jean; Rush, Alexander M. (2017): OpenNMT: Open-Source Toolkit for Neural Machine Translation. Available online at https://arxiv.org/pdf/1701.02810.

Li, Jiwei; Luong, Minh-Thang; Jurafsky, Dan (2015): A Hierarchical Neural Autoencoder for Paragraphs and Documents. In *arXiv:1506.01057 [cs],* checked on 5/29/2020.

Lin, Chin-Yew (2004): ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, pp. 74–81.

Liu, Yang (2019): Fine-tune BERT for Extractive Summarization. In *arXiv:1903.10318 [cs],* checked on 4/30/2020.

Liu, Yang; Lapata, Mirella (2019): Text Summarization with Pretrained Encoders. In *arXiv:1908.08345 [cs],* checked on 4/30/2020.

Nal Kalchbrenner; Phil Blunsom (2013): Recurrent Continuous Translation Models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1700–1709.

Nallapati, Ramesh; Zhou, Bowen; santos, Cicero Nogueira dos; Gulcehre, Caglar; Xiang, Bing (2016): Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond. In *arXiv:1602.06023 [cs],* checked on 5/29/2020.

Olah, Chris (2015): Understanding LSTM Networks -- colah's blog. Available online at https://colah.github.io/posts/2015-08-Understanding-LSTMs/, updated on 5/29/2020, checked on 5/30/2020.

Peters, Matthew E.; Neumann, Mark; Iyyer, Mohit; Gardner, Matt; Clark, Christopher; Lee, Kenton; Zettlemoyer, Luke (2018): Deep contextualized word representations. Available online at https://arxiv.org/pdf/1802.05365.

Popel, Martin; Bojar, Ondřej (2018): Training Tips for the Transformer Model. In *The Prague Bulletin of Mathematical Linguistics 110, April*. Available online at https://arxiv.org/pdf/1804.00247.

Rumelhart, David E.; Hinton, Geoffrey E.; Williams, Ronald J. (1986): Learning representations by back-propagating errors. In *Nature* 323 (6088), pp. 533–536. DOI: 10.1038/323533a0.

Rush, Alexander M.; Chopra, Sumit; Weston, Jason (2015): A Neural Attention Model for Abstractive Sentence Summarization. In *arXiv:1509.00685 [cs],* checked on 5/29/2020.

Strubell, Emma; Ganesh, Ananya; McCallum, Andrew (2019): Energy and Policy Considerations for Deep Learning in NLP. Available online at https://arxiv.org/pdf/1906.02243.

Sutskever, Ilya; Vinyals, Oriol; Le V, Quoc (2014): Sequence to Sequence Learning with Neural Networks. Available online at https://arxiv.org/pdf/1409.3215.pdf, checked on 5/30/2020.

Taylor, Wilson L. (1953): "Cloze Procedure": A New Tool for Measuring Readability. In *Journalism Quarterly* 30 (4), pp. 415–433. DOI: 10.1177/107769905303000401.

Thiruvengadam, Aditya (2019): Transformer Architecture: Attention Is All You Need. In *Medium,* checked on 5/16/2020.

Vaswani, Ashish; Shazeer, Noam; Parmar, Niki; Uszkoreit, Jakob; Jones, Llion; Gomez, Aidan N. et al. (2017): Attention is All you Need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.): Advances in Neural Information Processing Systems 30: Curran Associates, Inc, pp. 5998–6008. Available online at http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf, checked on 5/13/2020.

Weng, Lilian (2018): Attention? Attention! In *Lil'Log*. Available online at https://lilianweng.github.io/2018/06/24/attention-attention.html, checked on 5/16/2020.

Wu, Yonghui; Schuster, Mike; Chen, Zhifeng; Le, Quoc V.; Norouzi, Mohammad; Macherey, Wolfgang et al. (2016): Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. Available online at https://arxiv.org/pdf/1609.08144.

Xu, Kelvin; Ba, Jimmy; Kiros, Ryan; Cho, Kyunghyun; Courville, Aaron; Salakhutdinov, Ruslan et al. (2016): Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *arXiv:1502.03044 [cs],* checked on 5/16/2020.

Alammar, Jay (2018a): Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models With Attention). Available online at https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/, checked on 5/29/2020.

Alammar, Jay (2018b): The Illustrated Transformer, 6/17/2018. Available online at https://jalammar.github.io/illustrated-transformer/, checked on 5/14/2020.

Asadi, Ahmad; Safabakhsh, Reza (2020): The Encoder-Decoder Framework and Its Applications. In : Deep Learning: Concepts and Architectures, pp. 133–167.

Ba, Jimmy Lei; Kiros, Jamie Ryan; Hinton, Geoffrey E. (2016): Layer Normalization. In *arXiv:1607.06450 [cs, stat],* checked on 5/13/2020.

Bahdanau, Dzmitry; Cho, Kyunghyun; Bengio, Yoshua (2016): Neural Machine Translation by Jointly Learning to Align and Translate. In *arXiv:1409.0473 [cs, stat],* checked on 5/16/2020.

Bengio, Yoshua; Ducharme, Réjean; Vincent, Pascal (2000): A Neural Probabilistic Language Model. In *CrossRef Listing of Deleted DOIs* 1 (6), pp. 932–938. DOI: 10.1162/153244303322533223.

Chen, Mia Xu; Firat, Orhan; Bapna, Ankur; Johnson, Melvin; Macherey, Wolfgang; Foster, George et al. (2018): The Best of Both Worlds: Combining Recent Advances in Neural Machine Translation. Available online at https://arxiv.org/pdf/1804.09849.

Cho, Kyunghyun; van Merrienboer, Bart; Gulcehre, Caglar; Bahdanau, Dzmitry; Bougares, Fethi; Schwenk, Holger; Bengio, Yoshua (2014): Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. Available online at https://arxiv.org/pdf/1406.1078.pdf, checked on 5/30/2020.

Devlin, Jacob (2019a): BERT Multilingual. In *GitHub*, 10/17/2019. Available online at https://github.com/google-research/bert/blob/master/multilingual.md, checked on 4/30/2020.

Devlin, Jacob; Chang, Ming-Wei; Lee, Kenton; Toutanova, Kristina (2019b): BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *arXiv:1810.04805 [cs],* checked on 3/29/2020.

Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron (2016): Deep learning. Cambridge, Massachusetts London, England: The MIT Press (Adaptive computation and machine learning).

Hochreiter, S.; Schmidhuber, J. (1997): Long short-term memory. In *Neural Computation* 9 (8), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

Jean, Sébastien; Cho, Kyunghyun; Memisevic, Roland; Bengio, Yoshua (2014): On Using Very Large Target Vocabulary for Neural Machine Translation. Available online at https://arxiv.org/pdf/1412.2007.

Klein, Guillaume; Kim, Yoon; Deng, Yuntian; Senellart, Jean; Rush, Alexander M. (2017): OpenNMT: Open-Source Toolkit for Neural Machine Translation. Available online at https://arxiv.org/pdf/1701.02810.

Li, Jiwei; Luong, Minh-Thang; Jurafsky, Dan (2015): A Hierarchical Neural Autoencoder for Paragraphs and Documents. In *arXiv:1506.01057 [cs],* checked on 5/29/2020.

Lin, Chin-Yew (2004): ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, pp. 74–81.

Liu, Yang (2019): Fine-tune BERT for Extractive Summarization. In *arXiv:1903.10318 [cs],* checked on 4/30/2020.

Liu, Yang; Lapata, Mirella (2019): Text Summarization with Pretrained Encoders. In *arXiv:1908.08345 [cs],* checked on 4/30/2020.

Nal Kalchbrenner; Phil Blunsom (2013): Recurrent Continuous Translation Models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1700–1709.

Nallapati, Ramesh; Zhou, Bowen; santos, Cicero Nogueira dos; Gulcehre, Caglar; Xiang, Bing (2016): Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond. In *arXiv:1602.06023 [cs],* checked on 5/29/2020.

Olah, Chris (2015): Understanding LSTM Networks -- colah's blog. Available online at https://colah.github.io/posts/2015-08-Understanding-LSTMs/, updated on 5/29/2020, checked on 5/30/2020.

Peng, Hao; Mou, Lili; Li, Ge; Chen, Yunchuan; Lu, Yangyang; Jin, Zhi (2015): A Comparative Study on Regularization Strategies for Embedding-based Neural Networks. Available online at https://arxiv.org/pdf/1508.03721.pdf, checked on 10/13/2020.

Peters, Matthew E.; Neumann, Mark; Iyyer, Mohit; Gardner, Matt; Clark, Christopher; Lee, Kenton; Zettlemoyer, Luke (2018): Deep contextualized word representations. Available online at https://arxiv.org/pdf/1802.05365.

Pires, Telmo; Schlinger, Eva; Garrette, Dan (2019): How multilingual is Multilingual BERT? Available online at https://arxiv.org/pdf/1906.01502.pdf, checked on 10/15/2020.

Popel, Martin; Bojar, Ondřej (2018): Training Tips for the Transformer Model. In *The Prague Bulletin of Mathematical Linguistics 110, April*. Available online at https://arxiv.org/pdf/1804.00247.

Rumelhart, David E.; Hinton, Geoffrey E.; Williams, Ronald J. (1986): Learning representations by back-propagating errors. In *Nature* 323 (6088), pp. 533–536. DOI: 10.1038/323533a0.

Rush, Alexander M.; Chopra, Sumit; Weston, Jason (2015): A Neural Attention Model for Abstractive Sentence Summarization. In *arXiv:1509.00685 [cs],* checked on 5/29/2020.

Strubell, Emma; Ganesh, Ananya; McCallum, Andrew (2019): Energy and Policy Considerations for Deep Learning in NLP. Available online at https://arxiv.org/pdf/1906.02243.

Sutskever, Ilya; Vinyals, Oriol; Le V, Quoc (2014): Sequence to Sequence Learning with Neural Networks. Available online at https://arxiv.org/pdf/1409.3215.pdf, checked on 5/30/2020.

Taylor, Wilson L. (1953): “Cloze Procedure”: A New Tool for Measuring Readability. In *Journalism Quarterly* 30 (4), pp. 415–433. DOI: 10.1177/107769905303000401.

Thiruvengadam, Aditya (2019): Transformer Architecture: Attention Is All You Need. In *Medium,* checked on 5/16/2020.

Vaswani, Ashish; Shazeer, Noam; Parmar, Niki; Uszkoreit, Jakob; Jones, Llion; Gomez, Aidan N. et al. (2017): Attention is All you Need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.): Advances in Neural Information Processing Systems 30: Curran Associates, Inc, pp. 5998–6008. Available online at http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf, checked on 5/13/2020.

Weng, Lilian (2018): Attention? Attention! In *Lil'Log*. Available online at https://lilianweng.github.io/2018/06/24/attention-attention.html, checked on 5/16/2020.

Wu, Yonghui; Schuster, Mike; Chen, Zhifeng; Le, Quoc V.; Norouzi, Mohammad; Macherey, Wolfgang et al. (2016): Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. Available online at https://arxiv.org/pdf/1609.08144.

Xu, Kelvin; Ba, Jimmy; Kiros, Ryan; Cho, Kyunghyun; Courville, Aaron; Salakhutdinov, Ruslan et al. (2016): Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *arXiv:1502.03044 [cs],* checked on 5/16/2020.

Zhou, Wangchunshu; Ge, Tao; Xu, Ke; Wei, Furu; Zhou, Ming (2020): Scheduled DropHead: A Regularization Method for Transformer Models. Available online at https://arxiv.org/pdf/2004.13342v1.pdf, checked on 10/13/2020.