POLITECNICO DI MILANO

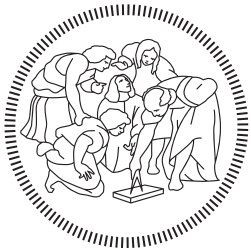
Facoltà di Ingegneria

Scuola di Ingegneria Industriale e dell'Informazione

Dipartimento di Elettronica, Informazione e Bioingegneria

Master of Science in

Computer Science and Engineering



Abstractive Text Summarization with Neural Sequence-to-Sequence Models

Advisor: PROF. MATTEO MATTEUCCI

Co-advisor: DR. GEORG REHM

Master Graduation Thesis by:

DMITRII AKSENOV

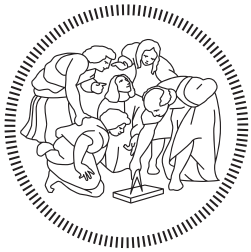
Student Id n. 10627370

Academic Year 2018-2019

POLITECNICO DI MILANO

Facoltà di Ingegneria
Scuola di Ingegneria Industriale e dell'Informazione
Dipartimento di Elettronica, Informazione e Bioingegneria

Corso di Laurea Magistrale in
Computer Science and Engineering



Abstractive Text Summarization with Neural Sequence-to-Sequence Models

Relatore: PROF. MATTEO MATTEUCCI

Correlatore: DR. GEORG REHM

Tesi di Laurea Magistrale di:

DMITRII AKSENOV
Matricola n. 10627370

Anno Accademico 2018-20219

ACKNOWLEDGMENTS

First of all I would like to thank Dr. Georg Rehm at the DFKI for giving me the opportunity to carry out state-of-the-art research in this field.

Furthermore, I would like to thank Prof. Matteo Matteucci for supervision of my thesis.

Special thanks to Prof. Dr.-Ing. Sebastian Möller, Dr. Julián Moreno-Schneider, Dr. -Ing. Leonhard Hennig, Peter Bourgonje and Robert Schwarzenberg and for their guidance.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

NLP	Natural language Processing
DL	Deep Learning
RL	Reinforcement Learning
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Unit
CNN	Convolutional Neural Network
API	Application Programming Interface
DFKI	Deutsches Forschungszentrum für Künstliche Intelligenz
LREC	The International Conference on Language Resources and Evaluation
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
BLEU	Bilingual Evaluation Understudy
BPE	Byte Pair Encoding
BERT	Bidirectional Encoder Representations from Transformers
ELMo	Embeddings from Language Models
ULMFiT	Universal Language Model Fine-tuning
GloVe	Global Vectors for Word Representation
LM	Language Model
TF	Term Frequency
IDF	Inverse Document Frequency
BPE	Byte Pair Encoding
HPC	High Performance Computing
REST	Representational State Transfer
Turtle	Terse RDF Triple Language

ABSTRACT

Nowadays, we face a permanent increase in the amount of unstructured information in text form. That calls for methods of automatic text summarization. In this thesis, we concentrate on the task of a single-document neural networks-based abstractive text summarization. We did several major scientific contributions. First of all, we explored to what extent knowledge from a pre-trained language model can be beneficial for the task of abstractive summarization. To this end, we experimented with conditioning the encoder, the decoder and the generator of a Transformer-based neural model on the BERT language model. The BERT conditioning showed huge improvement when used in encoder and decoder but was not useful for generator conditioning. Then, to alleviate the BERT's input size limitation we proposed a method of BERT-windowing. It allows chunk-wise processing of texts longer than the 512 tokens and respectively extends the BERT applicability. We also explored how locality modeling, i. e. the explicit restriction of calculations to the local context, can affect the summarization ability of Transformer. This was done by introducing 2-dimensional convolutional self-attention into the first layers of the encoder. Our abstractive models were evaluated and compared with state-of-the-art models on the CNN/Daily Mail dataset using ROUGE scores. We additionally trained our models on the German SwissText dataset to demonstrate the suitability of our model to languages other than English. All our final models outperformed the Transformer-based baseline and showed their superiority in manual qualitative analysis. Based on the results achieved we developed a summarization system. As the BERT-based model showed better results than convolutional self-attention-based we decided to use it in the release version of our summarization system. Last but not the least, we developed the extractive sentence-level summarization module to be able to handle significantly long documents that can not be efficiently processed by neural networks. This module is based on the TF-IDF sentence-level summarization but uses BERT's next sentence prediction capability to increase the consistency of the result summaries. In our summarization system, it is used as the first step of the summarization process before applying the abstractive model.

Abbiamo fornito numerosi importanti contributi scientifici. Prima di tutto, abbiamo esplorato fino a che punto la conoscenza di un modello linguistico pre-addestrato può essere utile per il compito di sintesi astratta. A tal fine, abbiamo sperimentato il condizionamento dell'encoder, del decoder e del generatore di un modello neurale basato su Transformer sul modello del linguaggio BERT. Il condizionamento BERT ha mostrato enormi miglioramenti se usato in encoder e decoder ma non è stato utile per il condizionamento del generatore. Quindi, per alleviare la limitazione delle dimensioni di input del BERT, abbiamo proposto un metodo di finestre BERT. Consente l'elaborazione in blocco di testi più lunghi dei 512 token e estende rispettivamente l'applicabilità BERT. Abbiamo anche esplorato il modo in cui la modellazione della località, cioè la limitazione esplicita dei calcoli al contesto locale, può influenzare la capacità di riepilogo di Transformer. Ciò è stato fatto introducendo auto-attenzione convoluzionale bidimensionale nei primi strati dell'encoder. I nostri modelli astrattivi sono stati valutati e confrontati con modelli all'avanguardia nel set di dati CNN / Daily Mail utilizzando i punteggi ROUGE. Abbiamo inoltre addestrato i nostri modelli sul set di dati SwissText tedesco per dimostrare l'idoneità del nostro modello a lingue diverse dall'inglese. Tutti i nostri modelli finali hanno sovraperformato la linea di base basata su Transformer e hanno mostrato la loro superiorità nell'analisi qualitativa manuale. Sulla base dei risultati raggiunti abbiamo sviluppato un sistema di riepilogo. Poiché il modello basato su BERT ha mostrato risultati migliori rispetto all'auto-attenzione convoluzionale, abbiamo deciso di utilizzarlo nella versione di rilascio del nostro sistema di riepilogo. Ultimo ma non meno importante, abbiamo sviluppato il modulo di riepilogo a livello di frase estrattivo per essere in grado di gestire documenti significativamente lunghi che non possono essere elaborati in modo efficiente dalle reti neurali. Questo modulo si basa sul riepilogo a livello di frase TF-IDF ma utilizza la capacità di previsione della frase successiva di BERT per aumentare la coerenza dei riepiloghi dei risultati. Nel nostro sistema di riepilogo, viene utilizzato come primo passo del processo di riepilogo prima di applicare il modello astrattivo.

1 INTRODUCTION

In the current digital age, we face a permanent increase in the amount of unstructured information in text form. Millions of individuals, organizations, and universities are creating new textual information every day. Such volume is very difficult to be processed manually. For a big set of documents, it may require an inefficiently long time. Moreover, the generated summaries are very subjective and prone to reflect the bias of the person performing summarization. That calls for methods to automatically extract the relevant information from unstructured documents and present it in some condensed form. Such methods are denoted automatic text summarization methods and have many real-wolds applications in research, finance, commerce, web, and public media.

Within the field of text summarization, different paradigms are recognized in two dimensions: extractive vs. abstractive, and single-document vs. multi-document. In extractive summarization, the task is to extract sentences or words from the text which carry the most important information, directly presenting the result of this as the summary. Abstractive summarization methods paraphrase the text and, by changing the text, aim to generate more flexible and consistent summaries. The extractive approach is easier and is ensured to generate grammatically and syntactically correct summaries. On the other hand, only within the abstractive approaches are possible paraphrasing, incorporation of the real-world knowledge and consistency between sentences. Furthermore, single-document summarization works on a single document at a time, while multi-document summarization deals with multiple documents at once and produces a single summary for them.

1.1 MOTIVATION

In this thesis, we concentrate on the task of a single-document abstractive summarization. Most recent abstractive models utilize the neural network-based sequence-to-sequence approach. Traditionally, these networks are trained in a supervised manner, meaning that they learn to generate summaries as much close to the human-generated gold summaries as possible. During training, such models calculate the conditional probability of a gold summary given the input sequence by maximizing the loss function (typically cross-entropy). Most approaches are based on the encoder-decoder framework where the encoder encodes the input sequence into a vector representation and

the decoder produces a new summary given the draft summary generated in previous iterations. The recent most efficient state-of-the-art sequence-to-sequence model is Transformer which is built primarily on the attention mechanism [3].

Such an abstractive summarization approach has several serious drawbacks. First, the neural models are restricted only to small texts not able to successfully remember long texts. Second, as all the summarization datasets are relatively small the resulted model does not represent the language at whole its richness. Hence, we find it important to research new summarization approaches that avoid text length limitations and empower some pre-training to improve the language encoding capabilities of the model.

Besides, many works showed that Transformer can achieve higher scores when self-attention is replaced by some other functions [31], [32]. One of the proposed functions is convolutional self-attention which adds hierarchy to the model pushing it to explicitly model the local dependencies between tokens [26]. That idea was never used in the text summarization research which makes it a valid target for this work.

1.2 OBJECTIVE

This work aims to build a state-of-the-art neural network-based abstractive multi-sentence text summarization system capable to summarize both short and long documents. The system is developing under the umbrella of the QURATOR [59] and LYNX [57] projects where it is to be used as a summarization microservice application. Lynx is a European Research Project which integrates and interlink various legal documents (legislation, case law, standards, industry norms and best practices) into the Legal Knowledge Graph. This graph will be accessible via the ecosystem of smart cloud services including one dedicated to text summarization. QURATOR ("Curation Technologies") is the initiative and framework that aims at developing artificial intelligence methods and research projects to be used in the industrial solutions for curating digital content. Both projects are multi-lingual that means that the summarization method is required to be language independent in its nature that means to be theoretically able to summarize documents in different languages.

First, we aim at building an extractive text summarization system to be able to shorten the texts of any length to the consistent texts of size processable by the state-of-the-art neural networks. The model must be improved by the conditioning on the pre-trained language model and explicit modeling of the local dependencies. Finally, he extractive summarization module must be integrated with the best abstractive summarization model via the REST API.

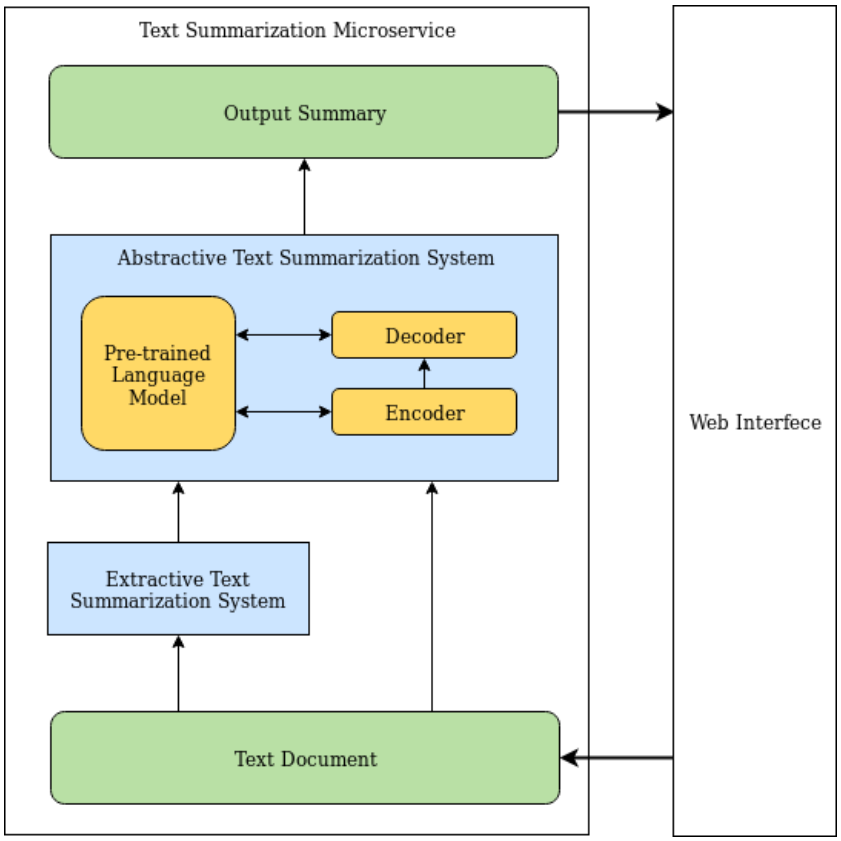


Figure 1.1: System Architecture

1.3 THESIS OUTCOME

In this thesis, we developed the text summarization system comprising the extractive and abstractive summarisation modules. The system was written in the Python language and uses the Deep Learning framework PyTorch, Docker containerization and web application framework Flask. The system was deployed in the Application Server (AS) to be accessed via Postman API and GUI. Figure 1.1 shows the general architecture of the developed system.

For the experiments, we are using the English CNN / Daily Mail dataset, the de facto text summarization standard, and nearly presented German SwissText dataset containing the set of Wikipedia articles with their abstracts. Our system was evaluated via ROUGE scores, the classical summarization evaluation method. The datasets were tokenized using the same tool to assume the comparability of the results of the experiments. In this way, we show the applicability of our method for texts in different languages. Moreover, to our knowledge, the results we obtained on the SwissText data were the first publicly released results which must contribute to the consecutive research on German texts' summarization.

Our extractive module is based on the well-known TF-IDF sentence-level extractive text summarization. Our key contribution to this

method is the usage of the pre-trained language model to keep the semantic and syntax consistency of the generated summaries.

The abstractive summarization model is based on the Transformer network with the pointer generator mechanism and the encoder-decoder framework. We did two major contributions to this baseline. First, we test several strategies of conditioning the encoder, the decoder and the generator of the model on a language model BERT obtaining the configuration of the model achieving close to the state-of-the-art results. In addition, we also developed a method of windowing the text and chunk-wise processing by a pre-trained language model. This method allowed us to encode the sequences which length exceeds the language model's input limitations. Second, we developed a model that exploits the idea of convolutional self-attention applied to the task of text summarization. We showed how it improves the evaluation score when used on the bottom layers of the encoder and discussed the possibility of its integration with the aforementioned language model conditioning.

In the final evaluation chapter, we compared our models with the state-of-the-art models and conducted a qualitative analysis on the text examples from the CNN / Daily Mail dataset.

We results of this work constituted a paper that was accepted for the LREC 2020 conference [60]. We put the pre-print of the paper in the appendix A.

1.4 OUTLINE

This thesis is separated into 8 chapters, including Introduction. The next chapters are:

Chapter 2 "Fundamentals and Related Work" presents the related works in the field of abstractive text summarization. It first introduces the relevant ideas behind state-of-the-art sequence-to-sequence models and then shows how they were applied to the specified problem by other scientists in the previous research.

Chapter 3 "Requirements" analyzes the functional and technical requirements for every components of the summarization system.

Chapter 4 "Models Description" describes our approach to the automatic summarization problem. It gives a high-level description to the system architecture and introduces various hypothesis which are later evaluated in the experiments.

Chapter 6 "Implementation" describes the implementation part of our work. In this chapter, we give a description of the used technologies and the program environment of the system. Then, we present

the setups for all the experiments, list the most important hyperparameters and their values and introduce the metrics used to assess the system's efficiency.

Chapter 5 "Datasets Description" introduces datasets used in training, validation and testing steps. Their differences and similarities are highlighted to find data-dependent patterns in the evaluation step.

Chapter 7 "Experimental Results" shows the results of the experiments stated in the chapter 6. In this chapter, we discuss which hypotheses were confirmed, how different ideas influenced the total performance of the system and what dependencies and patterns were found.

Chapter 8 "Conclusions" summarizes the thesis giving the discussion about the achieved results and drawing the final conclusion. It also proposes directions for future work.

2 FUNDAMENTALS AND RELATED WORK

In the first section of this chapter, we are going to provide a short description of the key ideas adopted by the modern Deep Learning-based NLP. The relevant ideas to be considered include, but are not limited to, neural networks, recurrence, attention, language modeling, different tokenization methods, distributed representations, and pre-training.

Then, we give a deep overviews of the cutting-edge methods in Deep Learning-based abstractive text summarization. We show how in the related scientific papers these ideas together with some new task-specific tricks were used to tackle the abstractive text summarization problem. This task required to solve some unique sub-problems such as limited networks' memory, non-uniformity of the vocabulary distribution, the limited number of contexts in a corpus, the degree of the internal structure in the texts and non-equality of loss functions with target measures.

Taking approaches described above we form several promising hypotheses to increase the model's accuracy which is to be described in more detail in the next chapter.

2.1 FUNDAMENTALS OF DEEP LEARNING IN NATURAL LANGUAGE PROCESSING

2.1.1 *Natural Language Processing*

Natural language processing (NLP) is a broad field of knowledge that deals with computational algorithms to automatically analyze, model and generate natural human language. In the age of the ubiquitous World Wide Web and the unstoppable rise of unstructured text information, NLP became one of the most important technologies. It covers many different tasks such as automatic parsing, text summarization, machine translation, named entity. NLP is highly connected with Artificial Intelligence science and often is considered as its part.

Traditionally, the NLP pipeline included many task-specific steps from pre-possessing to the final application. In recent years, this traditional approach was massively shifted by the new Deep Leaning-based approaches, which have achieved very high performance in various NLP tasks. This approach implies a training of end-to-end models without manual feature engineering.

Even within the Deep Learning framework, pre-processing is one of the most important steps in any NLP application. This process

determines the input to the models which significantly affects the final accuracy. The standard pipeline includes the following steps:

1. Cleaning (noisy data removal, stopword removal, capitalization, etc.)
2. Annotation(tokenization, part-of-speech tagging, etc.).
3. Normalization (Stemming, Lemmatization)
4. Analysis (TF-IDF, counting, etc.)

For neural networks, the most important pre-processing step is tokenization which splits the text into discrete tokens allowing it to be processed in sequential order by an NLP system. Tokenization allows us using a fixed vocabulary and One Hot Encoding to encode a sequence into the set of orthogonal vectors which can be consequently processed by a neural-network-based engine.

The most common approach is the word tokenization which splits the text into pieces separated by spaces and punctuation. The method is simple and fast but operates only with fixed vocabulary, whereas summarization is an open-vocabulary problem. Besides, this method performs badly on the morphologically rich languages

Another approach that mitigates the fixed vocabulary problem is the character tokenization. The character tokenizer splits texts into individual characters ensuring the vocabulary equal to the language alphabet. Despite the method's ability to adapt to new words, it is not widely adopted by the NLP community. The probability distribution of the next character given the previous is close to the discrete uniform that requires larger models to be able to catch the language structure. Larger models, in turn, need more data that drop the accuracy of the built models compared to the usage of word tokenization.

In the recent past, the NLP community, especially in Machine Translation research, widely adopted a new tokenization method called Byte Pair encoding. It tackles the out-of-vocabulary problem by encoding rare and unknown words as sequences of sub-word units. The intuition behind the approach is that various classes of words in a language can be represented as a combination of several meaningful units. For example, the morphological, phonological and compositional rules of the language produce various cognates, compounds, and loanwords. BPE allows building a compact vocabulary capturing the most common sequences of characters in the language.

2.1.2 *Language as Probability Model*

The most general task of NLP deals with building the models of the language. This task is highly related to the abstractive text summarization problem and its understanding is essential for dealing with the

cutting-edge summarization models. Language modeling is usually seen by statistical-based NLP as unsupervised distribution estimation from a set of examples each composed of variable length sequences of tokens. Every token is assumed to be left-conditioned of the previous tokens forming the conditional distribution (as a rule normal) over vocabulary. Due to the natural sequential ordering, the joint probability is usually factorized as the product of these conditional probabilities [15, 16]:

$$p(x) = \prod_{i=1}^{n} p(s_i|s_1, ..., s_{i-1}), \tag{2.1}$$

where p() is the joint probability and s_i denoted the i-th token.

The language model is the key method for generating distributed representations of tokens (also known as embeddings) to be later used in specific NLP tasks. As every sequence in the natural language is conditioned in both left-to-right and right-to-left order, usually, two respective models are trained and concatenated to get a more accurate representation. However, recently, the idea of masking the conditioned tokens allowed the explicit bidirectional conditioning [14, 12]. This allows us to reformulate the previous formula in the following way:

$$p(x) = \prod_{i=1}^{n} p(s_i|s_1, ..., s_{i-1}, s_{i+1}, ..., s_n) \tag{2.2}$$

The most precise language representation can be formed by counting models. However, this would demand an unfeasible amount of memory and time. Hence, practically language models are always approximated by some complex functions trained on the corpus of texts. Recently, Deep Learning approaches have obtained very high performance in language modeling becoming a standard tool for this task.

2.1.3 *Neural networks*

Artificial neural networks are kind of systems inspired by the biological neural networks which considered to be the source of human intelligence and consciousness. These systems have a statistical nature and learn to perform a particular task (in general case to perform approximation) from examples without the necessity to be programmed explicitly as a system of rules. We can adapt the neural networks to deal with text pairs by coding the input and target texts in the corpus using the One Hot Encoding (which represents a word as a vector with 1 in one position and 0 in rest). That allows us to build sequence-to-sequence models that can process texts directly without any additional pre-processing except some basic cleaning and tokenization.

The first artificial neural network called Perceptron was developed in 50ś by Frank Rosenblatt. All state-of-the-art gigantic neural net-

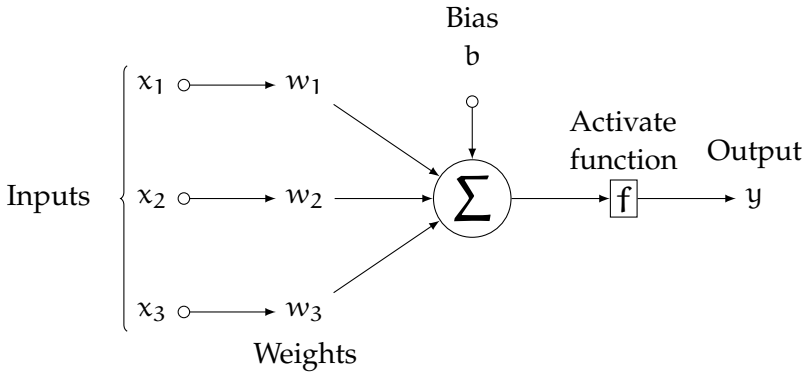


Figure 2.1: The general schema of an artificial neuron

works stem from this controversial work. Perceptron is a binary linear classifier which applies a threshold function on the linear combination of the input features. Perceptron with a threshold function replaced by some other non-linear activation functions (e. g. sygmoid, RELU, etc.) constitutes the basis for most modern neural networks (Figure 2.1).

The combination of perceptrons in connected layers resulted in a multilayer perceptron (MLP) which belongs to the class of free-forward artificial neural networks (networks which have no loops between its nodes). An MLP consists of at least three layers of parallel neurons: an input layer, a hidden layer with a nonlinear activation function at each neuron and an output layer (Figure 2.2).

The multilayer perceptron is proved mathematically be a universal function approximator. The universal approximation theorem states: Theorem

Theorem 2.1. A feed-forward network with a single hidden layer containing a finite number of neurons can approximate continuous functions on compact subsets of R^n, under mild assumptions on the activation function [2].

This theorem was formulated based on another theorem proven by George Cybenko in 1989 for sigmoid activation functions:

Theorem 2.2. Finite linear combinations of compositions of a fixed, univariate function and a set of affine functionals can uniformly approximate any continuous function of n real variables with support in the unit hypercube[17].

Considering a neural network as a classifier it means that, theoretically, with a big number of neurons and a learning algorithm achieving a global minimum, a free-forward neural network can in the most precise way distinguish not linear separable data. Practically, there does not exist an algorithm finding a global optimum for an arbitrary network. Free-forward neural networks usually are trained

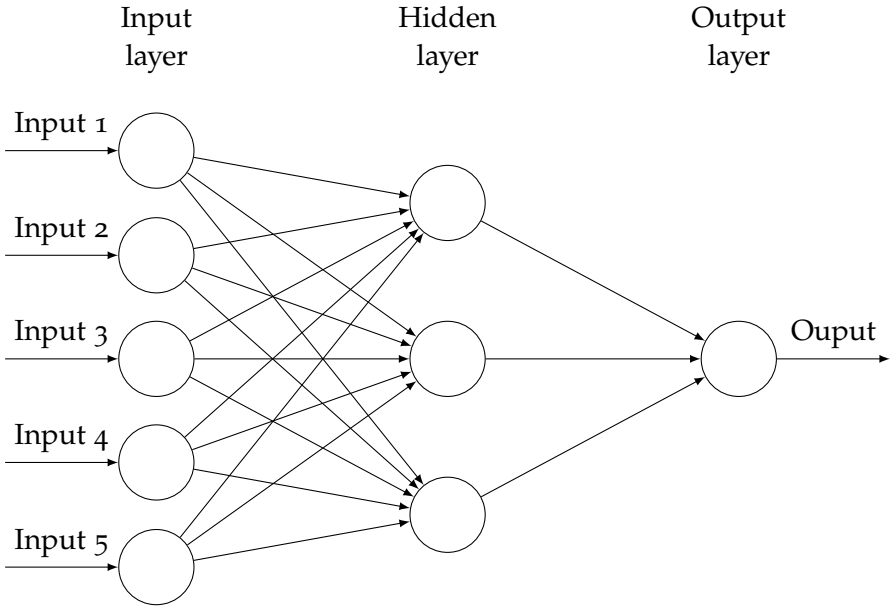


Figure 2.2: The schema of a multilayer perceptron

via a back-propagation algorithm that employs the gradient descent optimization of the loss function and chain rule to achieve a local minimum which is guaranteed to be close to the global minimum[18]. In such an algorithm, when the neural networks reach a particular depth they start suffering from the problem of vanishing gradient. The nature of the problem is the permanent decrease of the gradient for the deep layers preventing their weights from being updated. Many techniques were proposed to overcome the problem, among them there are receiver activation function, residual connections, and the hardware updates.

2.1.4 *Recurrent Neural Networks*

The standard free-forward neural network assumes the input to be of the same size and structured uniformly. This assumption becomes inadequate when data has a sequential nature and FNN could not anymore be used as a modeling tool. Sequential data exists in many forms: speech, video, time-series, text, or chain or events. Due to the nature of our task, we concentrate here only on text documents. They can be quite large consisting of thousands of words and have the variance in their length. In the inevitable lack of memory, time, and computational resources that prevents us from the usage of the standard FNN-based models. That led to the development of different neural network architectures eligible for sequential information processing. One of the most powerful models for processing of sequential infor-

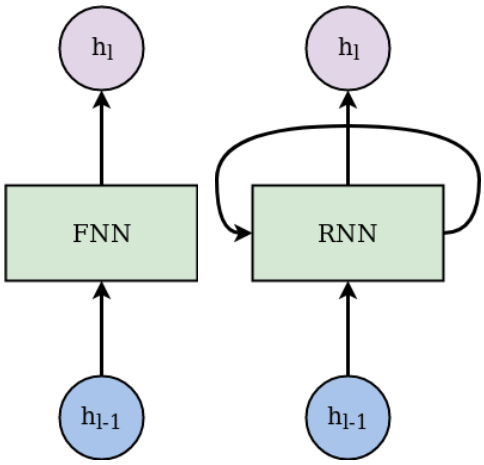


Figure 2.3: Architectures of the FNN and RNN.

mation is recurrent neural network. The main strength of RNN is its ability to memorize the results from the previous computation which allows modeling of the context dependencies in the input sequences of arbitrary length and remembering the elements' order. Every output of the network is conditioned on the previous computations through the combination of the hidden state from the previous iteration with the current hidden state. This procedure is repeated recursively for every token from the input sequence with the corresponding update of the network's weights. Hence, for classical RNNs, the deterministic state condition function is:

$$h_t^l = f(W_{n,n}^a h_t^{l-1} + W_{n,n}^b h_{t-1}^l), \tag{2.3}$$

where f is the non-linear function, l is the number of the layer and $W_{n,n}^a, W_{n,n}^b$ are the learnable matrices.
The Figure 2.3 illustrates the difference between FNN and RNN frameworks.

The drawback of the standard RNN is that it processes the sequence only in one direction. For every new token the network looks backward to the previous states to find some inter-tokens relationships. However, it is obvious that a sequence's elements can relate to the next tokens as well as to previous. Thus, in practice, bidirectional recurrent neural networks (BRNN) are utilized which analyze every text in both directions generating more informative contextual representations. The architecture for such networks represents two separate RNNs in which hidden states for every position in the sequence are concatenated into a final bidirectional representation.

The standard RNNs also suffer from the vanishing gradient problem. This implicitly prevents networks from learning long sequences. Another problem is akin to the gradient vanishing, but here we observe the vanishing influence of the past input, such that with the

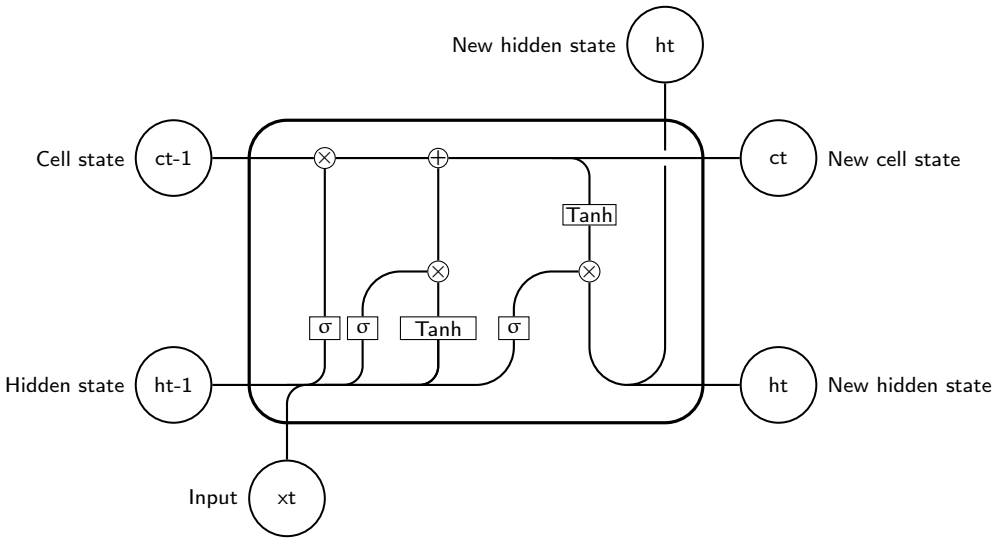


Figure 2.4: The schema of the LSTM cell

progress of the sequence the strength of the past tokens decreases. As a result, we can not nether train the network with long sequences nor generate long meaningful sequences. In other words, it can be said that traditional RNNs have a short-term memory. Hence, there was the idea to equip the standard neural networks with explicit long-termed memory. The most famous such architectures are long short-term memory (LSTM) and gated recurrent unit (GRU) [55]. This long-term memory is organized via the additional Cell state which is regulated by three "gates" (Figure 2.4). Respectively, the input gate controls the input flow into the cell, forget gate controls the extent of which we keep the previous information and the output gate controls the output flow from the cell. The recently proposed GRU is an LSTM without the output gate [56]. Such architectures proved to improve the ability to memorize information and were dominant in NLP research up to the invention of Attention-based models (look Section 2.1.7).

2.1.5 *Encoder-Decoder Framework*

The minimal architecture for a Sequence-to-sequence model consists of only one neural network where the input consists of the input text and the generated output draft summary separated by a special separation token. The problem here is that the length of the input and the output sequences vary considerably that undermines the efficiency of a simple use of one network. Therefore, most models are built upon the encoder-decoder framework. In this framework, we train two networks where an encoder is used to encode the input and a decoder to generate the text. In the first step, an input sequence is mapped by the embedding matrix to the distributed representations. Then the encoder transforms them into the contextual distributed representations (encoder hidden

states) which represent some information about the sequence in every vector. Encoders may be based on various architectures among those the most famous are variations of RNNs. However, to get rid of the sequential processing, currently, convolutional and attention-based models are replacing the traditional RNNs.

Decoder infers the output from the hidden states of the encoder. The decoding happens in a sequential manner where every newly generated token is conditioned on the sequence of the previously generated tokens. Its architecture usually is based on the same principle as the encoder. On the first step, the decoder, having the embedding vectors of the output draft and the set of hidden states of the encoder, calculates the vector which represents the information essential for the generation of a next token. Then, the generator layer maps this vector to the vector of probabilities over the vocabulary. Here, the normalization technique Softmax is used to restrict all the vector's components to the interval $(0, 1)$ and make them add up to 1 so that they can be interpreted as probabilities. The embedding of the most probable token is used as input in the next decoding iteration. The generation will stop after reaching the end of the sequence token.

2.1.6 *Attention*

Another recent idea which led to the significant improvement in the valorous NLP tasks is attention. The logic of this operation is to allow the generator at every iteration to consider all encoder hidden states while paying more attention to the most relevant tokens. In other words, it replaces the set of hidden states of the encoder by the set of weighted averages of these vectors The algorithm of attention is the following:

1. Map linearly the contextualized embedding vectors into the set of vectors called queries, values, and keys.
2. Apply the attention score function to the keys and queries to calculate the attention distribution. The distribution is normalized by the softmax function.
3. Using attention distribution calculate the context vector as a weighted sum of the values.

Equation 2.4 presents the algorithm in the matrix form:

$$\text{Attention}(Q, K, V) = \mathtt{softmax}(\mathtt{score}(K, Q))V \tag{2.4}$$

There exist many different score functions (Table 2.1). Among them, the most adopted are additive and dot-product functions. Although, they are similar in theoretical complexity dot-product outperforms other functions in terms of time and space efficiency.

Attention type	Score function	Citation
Content-base	$score(k_i, q_i) = cosine[k_i, q_i]$	[52]
Additive	$score(k_i, q_i) = \beta_a^T tanh(W_a[k_i, q_i])$	[53]
General	$score(k_i, q_i) = k_i^T W_a q_i$	[54]
Dot-Product	$score(k_i, q_i) = k_i^T q_i$	[54]
Scaled Dot-Product	$score(k_i, q_i) = \frac{k_i^T q_i}{\sqrt{d_k}}$	[3]

Table 2.1: Different attention score functions.

Attention type	Description	Citation
Global Attention	Attending to the whole target sequence	[54]
Local Attention	Attending to the target sub-sequence	[54]
Self-Attention	Attending to the source sequence	[3]

Table 2.2: Different types of attention.

Also, there exist three general approaches to the attention (Table 2.2). The global attention is the classical form of attention used in the most NLP models. However, local and self-attention recently gained much popularity as the tool to model the local dependencies ad as a replacement to the RNN and CNN networks respectively.

2.1.7 *Transformer*

The drawback of the traditional sequence-to-sequence models based primarily on the recurrent neural networks (even if they use attention, for example, [50]) is that they require sequential processing of the texts. This precludes the parallelization of the computations within training examples and respectively increases the training time needed for long sequences. To overcome this problem the new model Transformer relying entirely on the attention mechanism was proposed [3].

Transformer employs the encoder-decoder framework multi-layer architecture. Every Transformer layer consists of a stack of attention layer and a fully connected free-forward network followed by a normalization layer with residual connections. To encode the order of the tokens the special positional embeddings are calculated. Attention is calculated via the dot product and is used in two different ways. The first one, used only in the decoder, represents the typical attention mechanism which calculates the queries from the previous layer hidden state and values and keys from the encoder's output. The second type, called self-attention, calculates all of the keys, values, and queries from the hidden states of the previous Transformer layer. Also, the attention mechanism was applied in several threads (called heads)

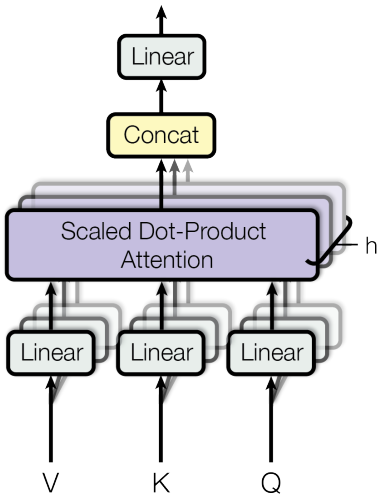


Figure 2.5: Multi-head attention. Source:[3]

on parallel which allows learning different contextual representation at each head (Figure 2.5). The Figure 2.6 presents the schema of the whole Transformer architecture.

This architecture reached a new state-of-the-art in machine translation task after which it was adopted by the majority of cutting edge models for various NLP tasks. However, as it compares with RNN-based and CNN-based models, Transformer, even often outperforming them, can not be considered superior in every case because all these models model different aspects of the data. As a result, many new ideas have been proposed to improve the baseline architecture by incorporation of convolution or recurrence in the architecture. In [31] it was shown that the main contribution of Transformer is its multi-layer architecture where self-attention can be replaced with recurrent or convolutional units without a significant drop in performance. [32] exploited this property and proposed the lightweight and dynamic convolution function that outperformed the self-attention baseline. Another task-specific problem is the fixed-length context in the language modeling task that prevents the model from learning the longer-term dependencies. A novel neural architecture Transformer-XL enabled learning dependency beyond a fixed length by introducing a segment-level recurrence mechanism and a novel positional encoding scheme into the vanilla Transformer [51].

2.1.8 *Pre-Trained Models*

The traditional machine learning-based NLP pipeline assumed that all models must be trained from scratch. Were it supervised machine learning methods over hand-crafted features or Deep Learning ap-

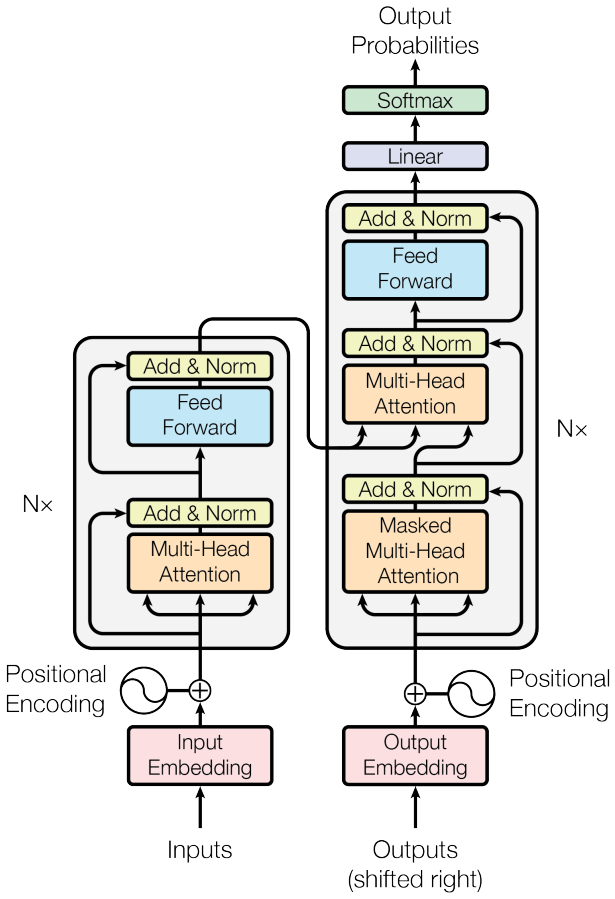


Figure 2.6: The architecture of Transformer. Source:[3]

proach they all did not employ any form of pre-training. As the training datasets for specific NLP tasks are relatively small this resulted in quite high bias.

This first successful attempt to change the situation is related to the concept of word embeddings. These are low dimensional distributed representations of words learned by neural models. This idea stems from the distributional hypothesis that states that the meaning of the word is determined by the context in which it occurs. The matrix mapping the high dimensional vectors of one-hot encoded tokens to the low dimensional vectors constitute the first layer of the neural models. Quickly people came to the idea of pre-training these embeddings on the vast amount of data instead of training them alongside the specific neural model on the tiny task-specific dataset. The specific popular variants of this approach include Word2Vec [43] and Glove [44]. These vectors can be downloaded and used directly at the bottom of the model.

The word embedding approach helped to significantly improve the accuracy of the sequence-to-sequence models but lacked the accountability of the local context. The meaning of the same word can be different in different contexts. This led to the idea of contextualized

word-embeddings which encode the meaning of the words within the sentences they appear. This is implemented as a language model where each token is either conditioned on the preceding or both preceding and successive tokens. Training of these models on the vast amount of texts makes the re-usability possible which means saving the time and resources required for training a task-specific model from scratch. Such language models can be used in two ways: to fine-tune for the specific task or to simply produce contextualized representation.

One of the first examples of such models that achieved state-of-the-art results on many tasks is ELMO [28]. It uses a bi-directional multi-layer LSTM where, as in the later proposed Transformer, the higher layers tend to model context-dependent aspects of meaning while lower layers model more syntactic information. Hence, for each input word, the model also learns the linear combination of the hidden states from different layers to account for different aspects of the word context in the most efficient way for a general language modeling task.

As Transformer deals with long-term dependencies better than LSTM many of the works use it as an LM. Currently the best model for the left-conditioned language modeling is GPT [6] (and its later version GPT-2[16]). It is based on the Transformer decoder which allows using the standard unmodified LM task.

GPT gave us the first fine-tunable Transformer based model. At the same time, it lost the bi-directionality provided by di-directional LSMT based models (e. g. ELMO). Here the idea of BERT [12] was born which imitates forward and backward conditioning of a sequence by using the Transformer encoder instead of the decoder. To do so BERT adopts the idea of masked language modeling when we mask the considered token conditioning it on all other tokens in the sequence. At the same time, the input still follows the left-right order. Besides, the model was also trained on the second task of predicting the probability the one sentence follows another. Such multi-task learning showed to give a strong improvement over the previous art models and helped to achieve state-of-the-art results in many various tasks.

The major problem with BERT is that it corrupts the input with masks. In the testing phase, the input is never masked whereas it is during training. To overcome this problem the XLNet [42] model was proposed. It is a generalized autoregressive model which instead of the fixed order modeling considers all the permutations of the sequence (so-called permutation language modeling). Therefore, masking becomes unnecessary and the model holds strong bi-directionality. Moreover, it also adopted the ideas of positional encoding and segment recurrence from the Transformer-XL language model to increase the memory capacity of the network. XLNet outperformed BERT in many tasks such as sentiment analysis, question answering, etc.

2.2 ABSTRACTIVE TEXT SUMMARIZATION WITH NEURAL NETWORKS

The task considered in this thesis is the abstractive text summarization of a single document with neural networks-based methods. Such an approach toward summarization was largely adopted by many state-of-the-art models in the recent past and raised the attention toward abstractive methods which previously always fell behind the extractive methods. Within the Deep Learning framework, this is a kind of sequence-to-sequence mapping task where we consider the article as input and the summary as output. Thus, we can use any model developed for the sequence processing including recurrent, convolutional and attention-based models. At the same time this task has several specific challenges:

- The length of the output is much smaller than the length of the input.
- The input and output texts should overlap on a large number of words and n-grams.
- The metrics used for evaluation is different from the loss we minimize.
- There are few big summarization datasets on a very limited number of domains.

Various state-of-the-art summarization models tackled to solve one or several of such issues by different modifications of the baseline NN architectures[19].

The first successful attempt to apply a modern neural network to abstractive text summarization was shown in [50]. This work achieved the state-of-the-art scores on Gigaword and DUC-2004 datasets and made the idea of attention ubiquitous in the field of summarization. The next breakthroughs became possible with the introduction of the new large-scale long text summarization datasets. The most used dataset nowadays is CNN / Daily Mail [10] based on the question-answering dataset by DeepMind [58].

The very significant contribution was the pointer-generator network [9]. It is a hybrid method that adds extractive capabilities into the sequence-to-sequence model allowing in addition to generation from vocabulary also to copy elements from the input text to the summary. Therefore, it also allows producing out-of-vocabulary words. To do so it calculates the generation probability p_{gen} (Equation 2.5) which is used as a switch between generation and copying of the token.

$$p_{gen} = \sigma(w_{h^*}^T h_t^* + w_s^T s_t + w_x^T x_t + b_{ptr}), \tag{2.5}$$

where h_t^* is the attention context vector, s_t is the decoder hidden state, x_t is the decoder input and w_{h^*}, w_s, w_x and b_{ptr} are learnable parameters.

Then, this probability is used to generate the final probability distribution over vocabulary where the copied word is sampled from the attention distribution (Equation 2.6).

$$P(w) = p_{gen} P_{vocab}(w) + (1 - p_{gen}) \sum_{i:w_i=w} a_i^t \tag{2.6}$$

Besides, it also uses the Coverage vector mechanism to pay less attention to tokens which were already paid enough attention in the previous iterations.

The idea of further hybridization with an extractive model was adopted by many consecutive works. [30] uses the extractive Key information guide network to guide the summary generation process. In [11] the Bottom-up summarization method was proposed. It uses the extractive model to increase the precision of the Pointer Generator mechanism. It predicts the probability for every token in the input text and triggers the pointer mechanism only for the high-scores tokens.

Another direction of the research is the modification of the neural models to cope with long texts exceeding the limits in length in which baseline models perform well. Some works target the sub-category of such texts which have an explicit internal structure. So, [34] presents the Discourse-Aware Attention model which introduces hierarchy in the attention mechanism via calculating an additional attention vector over the sections of the input text. Otherwise, the standard approach for really long texts is a hybrid two-stage summarization process where we first summarize text extractively and then apply the abstractive model on the intermediate summaries. Such an approach was used in [5] which uses the Transformer decoder with various attention compression techniques needed to increase the maximal size of the input sequence on top of the TF-IDF extractive system to summarize the Wikipedia articles. To our knowledge, it also was the first time when the Transformer model used in summarization. Another original approach is to perform summarization by the standard language model. So, [41] showed that the language model trained on the combination of the original text, an extractively generated summary, and the golden summary can achieve the results comparable with the standard encoder-decoder based summarization models.

Some works raised the problem of the discrepancy of the loss function and the evaluation function used in the summarization and adapted the reinforcement learning (RL) strategy (for example [7]). The usage of the ROUGE-based reward allowed to explicitly optimize the ROUGE score. The pure RL models showed high ROUGE-1 and ROUGE-L scores but, unfortunately, produced the summaries looking very unnatural. However, its combination with usual cross-entropy optimization achieved high scores eliminating the unreliability problem.

Currently, most of the cutting-edge models use some form of fine-tuning of the pre-trained language models. [25] incorporates BERT into the Transformer-based model in the two-stage process. First, the

combination of BERT and the Transformer decoder generates the intermediate summary. Then it exploits the masking learning strategy to rearrange the generated summary. The optimization uses both cross-entropy and RL losses. [46] and [45] using the similar approach present the latest state-of-the-art summarization model.

2.2.1 *Summarization Evaluation*

Evaluation is the crucial step for the summarization method progress. Currently, human evaluation is the method that allows getting the most valid assessment working for any domain, score range, and text size. Unfortunately, this method is very time consuming and expensive which prevents it from the ubiquitous adaptation. This encourages researchers to use various automatic evaluation techniques.

ROUGE [20] is the most adopted evaluation metric used in automatic text summarization research. The evaluation is made through a comparison of a set of system-generated candidate summaries with the gold standard summary. The availability of the accompanying software and its performance greatly contributed to its popularity [35]. Despite its adoption in many studies, the metric faced some key criticism.

The main criticism of ROUGE is that it does not take into account the meaning expressed in the sequences. The metric was developed based on the assumption that a high quality generated candidate summary should share many words with a single human-made gold standard summary. This assumption may be very relevant to the extractive summarization task but not to the abstractive task, where different terminology and paraphrasing can be used to express the same meaning [35]. This results in the metric assigning low scores to any summary not matching the gold standard at the appropriate level. This also allows cheating of the metric by generating ungrammatical and nonsensical summaries having very high ROUGE scores. [39] shows how this can be achieved by choosing the most frequent bigrams from the input document.

ROUGE adoption relies on its correlation with human assessment. In the first research on the DUC and TDT-3 datasets containing news articles ROUGE indeed showed a high correlation with the human judgments [20, 37]. However, more recent research questioned the suitability of ROUGE for various settings. It was shown that on DUC data the linguistic and responsiveness scores of some systems do not correspond to the high ROUGE scores [36]. In [35] it was demonstrated that for summarization of scientific texts, ROUGE-1 and ROUGE-L have very low correlations with the gold summaries. ROUGE-N correlates better but is still far from the ideal case. This follows the result of [38], showing that the unigram match between the candidate

summary and gold summary is not an accurate metric to asses quality on the meeting dataset.

Another problem is that the credibility of ROUGE was proven for the systems which operated in the low-scoring range. [49] shows that different summarization evaluation metrics correlate differently with human judgments for the higher-scoring range in which state-of-the-art systems now operate. Furthermore, improvements found for one metric do not necessarily lead to improvements when using the other ones.

This concern led to the development of many new evaluation metrics. [48] using the idea of Shannon's entropy defines metrics for important concepts with regard to summarization: Redundancy, Relevance, and Informativeness. From these definitions, they formulate a metric of Importance which better correlates to human judgments. [47] proposes the metric of Sentence Mover's Similarity which operates on the semantic level and also better correlates with human evaluation. A summarization model trained via Reinforcement Learning with this metric as reward achieved higher scores in both human and ROUGE-based evaluation.

Despite these drawbacks, the high adoption of ROUGE makes it the only way to compare the efficiency of our model with other state-of-the-art models. The evaluation of our system on the SwissData dataset confirms that its efficiency (in terms of ROUGE) is not restricted to CNN / Daily Mail data only.

3 REQUIREMENTS

3.1 DATA REQUIREMENTS

The developed service is supposed to be used for summarization of documents from very different domains and in various languages. The lack of publicly available data though does not allow us to develop a fully multilingual multidomain neural summarization system. Hence, under this work, we decide to restrict the experiments to only English and German languages. We believe the news and general concept descriptions domains would fit the goals the best way as such texts usually cover the maximal wide range of topics. The data should be split into the training, validation and testing subsets. All the data samples must be the first tokenized using the appropriate tokenizer and prepossessed for the sake of compatibility with the system being developed.

3.2 TECHNICAL REQUIREMENTS

3.2.1 *Computational Resources*

The modern Deep Learning methods require a lot of computations for the models to converge. As a result, the neural network-based models are almost always trained on the cluster of GPUs that accelerate the training time in several times. The majority of Deep Learning frameworks (such as Tensorflow or PyTorch) require the video cards with The support of Nvidia technology CUDA. Accordingly, we are required an HPC cluster with Nvidia Tesla or GTX video cards to finish the research in a reassemble time.

Besides, we are also required a web server to deploy the system as a microservice. The server should not necessary posses GPUs and can conduct all the computations on a CPU only.

3.2.2 *Programming Environment*

The major part of this work relies on the Deep Learning approach. In recent years, Python in combination with various Deep Learning libraries has become a standard tool for neural network development. Hence, the standard DL environment forms the main requirement for our work.

As the abstractive text summarization in Deep Learning approach belongs to a category of sequence-to-sequence tasks we consider it

reasonable to base the system on some general sequence-to-sequence NLP library to accelerate the development time.

The final system is supposed to be deployed as a microservice on a web server. To ensure the consistency of the software dependencies, the entire environment of the system needs to be virtualized by some containerization tool.

The final Web interface development lies outside of the scope of this research. Still, we need to develop a demo interface to interact with the summarization microservice. For this task any API testing tool, which supports the REST eligible APIs and HTTP requests, is suitable.

3.3 FUNCTIONAL REQUIREMENTS

3.3.1 *Extractive Summarization Module*

The recent developments in the area of Deep Learning allowed us to achieve very high efficiency in various NLP tasks. State-of-the-art neural summarization models already equal and often surpass the extractive models in the result accuracy. The major shortcoming of these models is that they implicitly (due to the memory limitations) allow only relatively small input texts. The standard approach to tackle this problem is to truncate all the input texts to the unite length. This strategy is quite efficient to the medium-length sequences but fails to achieve the appropriate results with bigger documents where the important information is spread uniformly over the text. That limitation highly restricts the approach's usability and industrial adoption. Thus, as the length of our target documents can reach quite large values we conclude the necessity of developing the extractive summarization module to be able to select the most meaningful sentences to use them as the input to the abstractive summarization module. This extraction tool is expected to deal with sequences of any length. Hence, it would be more logical to use some standard rule-based approach which does not tend to implicitly assume a particular sequence length (as happens when we train a model on a particular corpus).

Also, we assume that abstractive summarization system input should be semantically and syntactically consistent to increase the abstractiveness of the system. Thus, the extractive summarizer must produce the summaries which are also consistent.

3.3.2 *Abstractive Summarization Module*

The abstractive summarization module should output the final abstractive summary when fed the original text or the intermediate summary generated by the extractive summarizer. We decided to restrict our system to the models based on Transformer architecture only as it was shown that it achieves better accuracy in the summarization task

compared to the models based on pure recurrent neural networks [11]. We also decide to use an encoder-decoder framework, Pointer Generator and Beam search following the majority of the recent state-of-the-art works [19]. The model's performance must be evaluated using a standard for text summarisation metric ROUGE [20]. In the current work, we restrict the research to the two areas: conditioning of the Transformer on some pretrained LM and the replacement of the standard Self-Attention with its convolutional variation.

4

MODEL DESCRIPTION

4.1 CONSISTENT EXTRACTIVE SUMMARIZATION MODULE

As stated in the requirements chapter, our extractive text summarization system is supposed to be primarily rule-based with the additional property of generating a consistent summary. Taking this into account we decided to base our system on the simple TF-IDF (term frequency-inverse document frequency) algorithm following the work [5]. TF-IDF is a statistic reflecting the importance of the token to a document in the collection of documents. The final score can be found by multiplication of the following terms:

$$\mathrm{tf}(t, d) = \frac{n_t}{\sum_k n_k}, \tag{4.1}$$

where n_t is the number of occurrences of the token t in the document, and in the denominator there is the overall number of words in the document.

$$\mathrm{idf}(t, D) = \log \frac{|D|}{|\{ d_i \in D \mid t \in d_i \}|}, \tag{4.2}$$

where $|D|$ is the overall number of documents in the collection and in the denomenator there is the number of documents in the collection D where the token t appears.

Our realization of this algorithm receives the percentage of the information to keep as a parameter and returns the corresponding number of ordered by their TF-IDF score sentences.

We also modified the general TF-IDF algorithm to transfer a larger amount of useful information to the abstractive summarizer. Recent state-of-the-art summarization systems usually show performance close to (and often lower) the "lead-3" baseline which takes as a summary only the first 3 sentences of the text. Hence, we also decided always to add 3 first sentences to the summary even when they were scored low.

The second modification relates to our approach to increase summary consistency. As we can logically assume every human-generated text recursively depends on itself. Every token or sentence syntactically and semantically depends on other sentences and tokens in text. Respectively, the removal of one sentence from a text can cause information loss in other sentences depending on it. As a result, the TF-IDF algorithms (as any extractive summarization algorithm) can

produce summaries that are not consistent. To alleviate this problem we decided to use the property of the BERT model to predict the probability of one sentence going after another. Each adjacent pair of sentences from the given summaries were tested if they have positive BERT scores. For each negative pair of sentences (let call them original sentences) we look at other sentences from the original text placed between them and calculate for each the probabilities of being placed between original two sentences ignoring the other. If some sentences obtained a positive probability with both left and right original sentences we add the most probable one to the summary. This procedure can theoretically be recursively repeated to select the set of sentences having obtaining higher consistency. However, we decided to stop calculation only on one sentence for two reasons. First, to not increase the length of summaries too much and save the computational time. And second, because BERT sometimes also scores two adjacent sentences from the original text negatively that can lead to the uncertainty about the condition to stop the computation.

4.2 ABSTRACTIVE SUMMARIZATION MODULE

Our text summarization model is based on the Transformer architecture. The architecture repeats the original model [3]. On top of the decoder, we use a Pointer-Generator (formula 4.3) to increase the extractive capabilities of the network (we later refer to this architecture as CopyTransformer).

$$p(w) = p(z = 1)p_{copy}(w) + p(z = 0)p_{softmax}(w), \tag{4.3}$$

where $p(z = 1$ is the probability of coping, $p_{copy}(w)$ is the probability of copying the specific word from the source document and $p_{softmax}(w)$ is the probability calculated by the abstractive summarization model.

For a better understanding of the trained baseline model, we plotted the summed-up logarithmic probability distributions over the vocabulary within all positions in the random summary taken from the CNN / Daily Mail dataset. As expected for the text generation models, the final distribution is normal (Figure 4.1). To fulfill the requirements this vanilla model was extended by the application of convolutional self-attention and conditioned on the pre-trained LM. The following sections describe these improvements and their modifications in more detail.

4.2.1 *Convolutional Self-Attention*

Transformer, like any other self-attention network, has a hierarchical multi-layer architecture. In many experiments it was shown that this architecture tends to learn lexical information at the first layers,

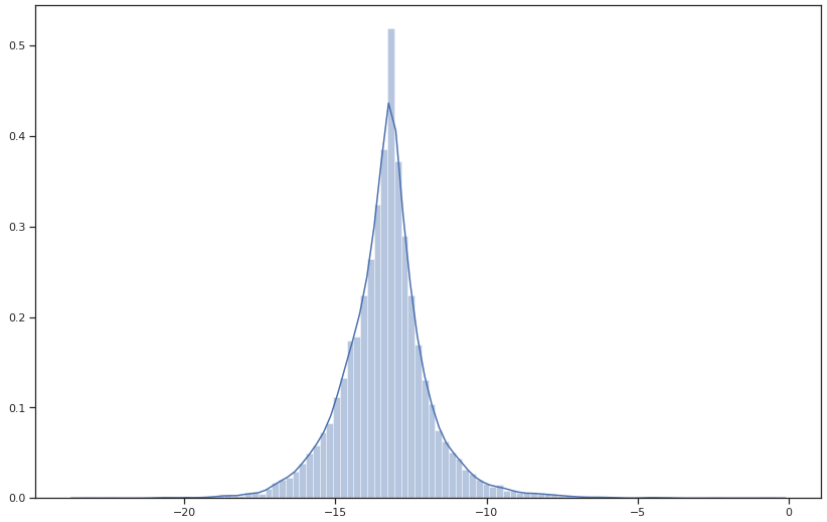


Figure 4.1: Baseline model. The probability distribution for a next output token.
X axis: log probabilities, Y axis: frequency

sentence-level patterns in the middle and the semantics at the upper layers [27],[33]. The disadvantage of this approach is that during the attention operation it considers all tokens as equally important, whereas syntactic information is mostly concentrated in the center of a local area. This problem is usually specified as the problem of locality modeling. As syntactic information can help in identifying more important words or phrases, it would be beneficial to allow the focus of attention on these regions.

A successful approach to the locality modeling task is the so-called convolutions (local) self-attention networks [26]. Essentially, the problem is dealt with by the application of a 1-dimensional convolution to the self-attention operation at the network's lower layers. This strengthens dependencies among neighboring elements and makes the model distance-aware when it searches for low-level patterns in a sequence. In other words, it restricts the attention scope to the window of neighboring elements. The 1D convolution applied to attention is illustrated in the Formulas 4.4, 4.5 and 4.6.

$$\widehat{\mathbf{K}}^h = \{\mathbf{k}^h_{i-\frac{M}{2}}, \ldots, \mathbf{k}^h_i, \ldots, \mathbf{k}^h_{i+\frac{M}{2}}\}, \tag{4.4}$$

$$\widehat{\mathbf{V}}^h = \{\mathbf{v}^h_{i-\frac{M}{2}}, \ldots, \mathbf{v}^h_i, \ldots, \mathbf{v}^h_{i+\frac{M}{2}}\}, \tag{4.5}$$

$$\mathbf{o}^h_i = \textsc{Att}(\mathbf{q}^h_i, \widehat{\mathbf{K}}^h)\widehat{\mathbf{V}}^h, \tag{4.6}$$

where $\mathbf{q}^h_i$ is the query and $M+1$ ($M \leqslant I$) is its attention region centered at the position i.

The convolution can be extended to the 2-dimensional area by taking interactions between features learned by the different attention heads of the Transformer into account. In the original Transformer each head independently models a distinct set of linguistic properties and dependencies among tokens [27]. By applying 2-dimensional convolution, where the second dimension is the index of attention head, we explicitly allow each head to interact with learned features for their adjacent sub-spaces. The shortcoming of the original implementation is that the first and the last heads do not interact as they are assumed not adjacent. Thus, we assume that considering the heads' sub-spaces periodic, we can increase the model's effectiveness by applying circular convolution to the second dimension. In Chapter 7, we evaluate both the original version and our modification.

$$\widetilde{\mathbf{K}}^h = \bigcup[\widehat{\mathbf{K}}^{h-\frac{N}{2}}, \ldots, \widehat{\mathbf{K}}^h, \ldots, \widehat{\mathbf{K}}^{h+\frac{N}{2}}], \tag{4.7}$$

$$\widetilde{\mathbf{V}}^h = \bigcup[\widehat{\mathbf{V}}^{h-\frac{N}{2}}, \ldots, \widehat{\mathbf{V}}^h, \ldots, \widehat{\mathbf{V}}^{h+\frac{N}{2}}], \tag{4.8}$$

$$\mathbf{o}_i^h = \textsc{Att}(\mathbf{q}_i^h, \widetilde{\mathbf{K}}^h)\widetilde{\mathbf{V}}^h, \tag{4.9}$$

where $(M+1)$ $(N \leqslant H)$ is the window region over heads and $\bigcup$ stands for the union of keys $\widehat{\mathbf{K}}^h$ and values $\widehat{\mathbf{V}}^h$ from different subspaces.

The convolutional self-attention has been shown to be very effective in Machine Translation and several other NLP tasks. However, to our knowledge, it was never applied to the text summarization problem. For the experiments reported in this thesis, we created our implementation of the local attention and the convolutional self-attention network (based on Transformer). It supports both 1D and 2D modes having the size of the kernels as system parameters. As in [26] we incorporate convolutional self-attention in the Transformer encoder by positioning it in the place of the self-attention in the lower layers. In Section 7.3, we show that the low-level modeling capabilities of our encoder provide a strong boost to the model's prediction accuracy in the text summarization task.

4.2.2 *Pre-Trained Language Models Comparison*

Our main contribution in this work is the usage of the pre-trained language model to condition our summarization model. To start with we decided to conduct a set of experiments to select the model which fits our need the most. We consider tree most new Transformer-based LMs: GPT2, BERT, and XLNet.

The knowledge we try to incorporate into the summarization system is the language model' property to predict the probability of a token going next to the previously generated sequence. To test and select

the model we develop a special procedure. We assume that the right language models should give a high probability to the true token when evaluated on the human-generated sequence and low probability when evaluated on the random-generated sequence. Putting some probability threshold we can classify the target token to be right or false. It gives us the ability to calculate the recall of the system output. To test the permutation detection ability of the system we also shuffle our test texts. Finally, the overall accuracy over the original text and shuffled is calculated. On the text box 7.1 we can you can see three texts used in our experiments. The first two are taken from the target of the dataset CNN/Daily News and the last one is a randomly generated text. Our primary goal is to select a model that has the highest original text recall and lowest shuffled and random text recall.

We use tree thresholds: 90%, 95% and 99% (percentage of tokens classified as false). The results of all the experiments are presented in the tables 4.1, 4.2 and 4.3.

marseille prosecutor says ‘ ‘ so far no videos were used in the crash investigation ’ ’ despite media reports . journalists at bild and paris match are ‘ ‘ very confident ’ ’ the video clip is real , an editor says . andreas lubitz had informed his lufthansa training school of an episode of severe depression , airline says .
membership gives the icc jurisdiction over alleged crimes committed in palestinian territories since last june . israel and the united states opposed the move , which could open the door to war crimes investigations against israelis .
new say horus gray say bear, yellow run discrepancy driver cd memes Androgine Germany , I apple Demolished. invoke Bottle ’

Text box 4.1: Sequennces used for the language models comparison. First two are the sample examples from the pre-processed and detokenized CNN / Daily Mail dataset and the last one is the randomly generated sequence of words

Based on the results we ranked all the models for all tree texts (figures in parentheses indicate the threshold):

- **1 natural text:** GPT2(99), BERT(99), GPT2(95), BERT(95), BERT(90), GPT2(90), XLNET(99), XLNET(95), XLNET(90)
- **2 natural text:** GPT2(99), XLNET(95), BERT(99), BERT(95), BERT(90), GPT2(90), XLNET(99), GPT(95), XLNET(90)
- **Random text:** XLNET(99), BERT(99), XLNET(95), GPT2(99), XLNET(90), GPT2(95), BERT(95), GPT2(90), BERT(90)

As we can see from the results for the random text XLNet(99) gives us the best overall accuracy followed by BERT(99) and XlNet(95). On the other hand, on the natural texts GPT2 scores higher. We assume

Percentile	Text №	Recall	Recall when shuffled	Accuracy
90%	1	85.9%	21.2%	53.5%
	2	90%	24%	57%
	3	47.9%	–	–
95%	1	83.5%	28.2%	55.9%
	2	90%	40%	65%
	3	62.5%	–	–
99%	1	54.9%	48.2%	57%
	2	70%	48%	59%
	3	71.9%	–	–

Table 4.1: Exploration of the XLNet model

Percentile	Text №	Recall	Recall when shuffled	Accuracy
90%	1	95.5%	25.4%	58.9%
	2	88.9%	33.3%	61.1%
	3	34.3%	–	–
95%	1	79.1%	38.8%	58.9%
	2	86.1%	38.9%	62.5%
	3	43.7%	–	–
99%	1	67.2%	59.7%	63.4%
	2	72.2%	52.8%	62.5%
	3	65.6%	–	–

Table 4.2: Exploration of the BERT model

it is the result of the permutation stability of the XLNet's training procedure which leads to the low recall on the shuffled text. The overall accuracy is proportional to percentile used, as with the rise of the strictness of the model the recall on shuffled texts also rises. Among the models giving the highest recall on the original text BERT(95) scores largest while in general BERT(99) always gets the second-third best score.

Summing it all up, we find here some kind of trade-off. XLNet, as the most complicated model, performs the best in finding not-natural structures. GPT2 performs on this task the worst but gives the best quality in the detection of the not-natural order (handling of shuffling) of the tokens. BERT lies in between giving not the best but relatively high result on both tasks and also performing best in finding mistakes

Percentile	Text №	Recall	Recall when shuffled	Accuracy
90%	1	98.7%	16%	57.3%
	2	97.8%	22.3%	60%
	3	35.7%	–	–
95%	1	93.3%	25.3%	59.3%
	2	93.3%	24.4%	58.9%
	3	46.4%	–	–
99%	1	85.3%	42.7%	64%
	2	88.9%	53.3%	71.1%
	3	57.1%	–	–

Table 4.3: Exploration of the GPT2 model

when the model is minimally strict to classify the largest number of target tokens as right.

In these considerations, we decided that BERT would be the best choice for our task providing the best performance in different circumstances with the highest level of acceptance for the tokens being generated.

4.2.3 *BERT-Conditioned Encoder*

The main task of the encoder is to remember all the semantic and syntactic information from the input text which should be used by the decoder to generate the output. Knowledge transfer from the language model should theoretically improve its ability to remember important information due to the much larger corpus used in its pre-training phase compared to the corpus used in the text summarization training phase. We thus condition our encoder on the BERT language model.

For the encoder conditioning, we used the most straightforward strategy recommended for the BERT based model: Placing the pre-trained language model in the encoder as an embedding layer. This should make the embeddings of the system context-dependent. We decided to not fine-tune BERT for the sake of memory and time economy. Instead, we follow the general recommendations by concatenating the hidden states of the last 4 layers of BERT into a 3072-dimensional embedding vector [12]. As it is different from the hidden size used in our experiments (Chapter 6) we need some kind of dimensionality reduction. In this work, we try two strategies: a simple linear transformation and a non-linear reduction via a free-forward neural layer with the RELU activation function. The first one, being a linear model,

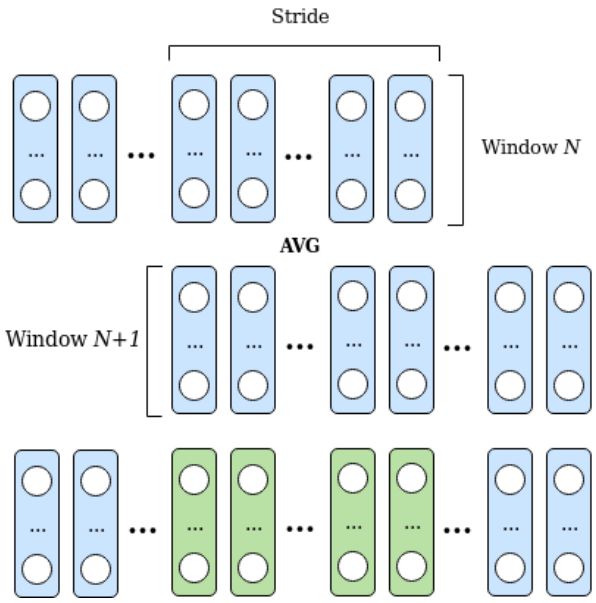


Figure 4.2: Integration of BERT-generated contextual representations from two windows

should converge faster losing some non-linear dependencies in the original vectors. The second can approximate the vector more precisely in the price of higher variance decreasing the model's accuracy.
The first model uses only BERT to encode the input sequence and the second model feeds BERT's generated embeddings into the vanilla Transformer encoder [25]. Here we can observe the same Bias-Variance trade-off as in the dimensionality reduction layer.

4.2.4 *BERT-Windowing*

One of the key features of our approach is its ability to overcome the length limitations of BERT, allowing it to deal with longer documents. The BERT's maximum supported sequence length is 512 tokens, which is smaller than the average size of texts used in most summarization datasets. The first most obvious approach to alleviate this problem is to truncate the input text to the required length. We compare this approach to our technique of windowing the BERT model to decide about the final model architecture.

Our method relies on the well-known method of windowing which to our knowledge was nevertheless never used before either in the BERT-based models nor in the abstractive text summarization research (Figure 4.2). We apply BERT to the windows of texts with strides and generate N matrices embedding each one window. Then we combine them by doing the reverse operation. The vectors at the overlapped positions are averaged (by summing and dividing by the number of overlapping vectors). As a result, we have the matrix of embeddings

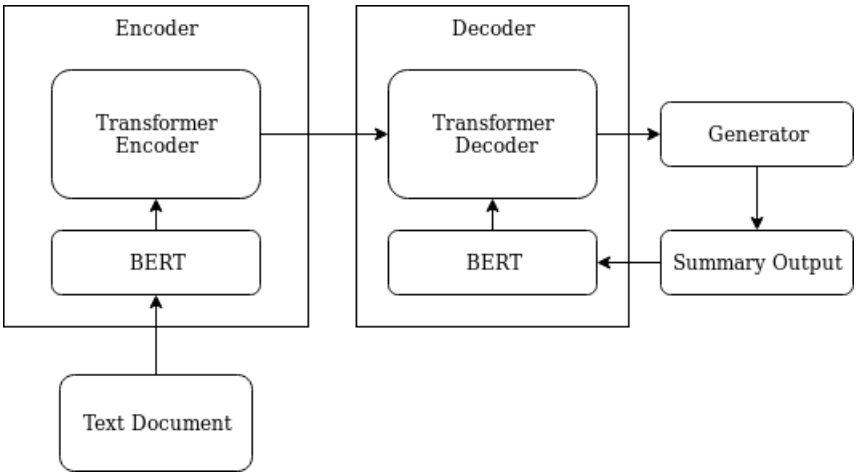


Figure 4.3: Model Overview

with the shape of the hidden size times the length of the text. The drawback of this approach is that we reduce the size of the context as each resulted vector is calculated based on maximum twice the window size number of tokens. Besides, the split of the text to equal size windows will aggravate the consistency of the input as some sentences will be split in an arbitrary manner between two adjacent windows. Despite this drawback, we assume that this procedure will nevertheless improve the accuracy of the encoder trained on the non-truncated texts. We set the window size to the maximum size of 512 tokens and the stride to 256. We consider this stride optimal as it provides the largest average context (768 tokens for all except the 256 initial and final tokens) with the minimal number of windows reducing the computational requirements of the model.

4.2.5 *BERT-Conditioned Decoder*

In the decoder, pre-training was applied similarly. The same approaches to embeddings construction, dimensionality reduction, and long text support were used. The main difference is that instead of the final output of BERT we use only its word embedding matrix (without positions). The reason behind this is that in the decoder the generated probability distribution is conditioned on the incomplete text (previous summary draft output) while BERT implicitly assumes consistent and completed input [25]. As context-independent embeddings are not enough to represent the minimum set of features to make a meaningful prediction the custom Transformer decoder is always stacked on top of BERT.

Our whole BERT-based model is similar to One-Stage BERT [25] and BertSumAbs [45] but differs in the usage of the four last hidden states of BERT to create contextualized representations, in presence of

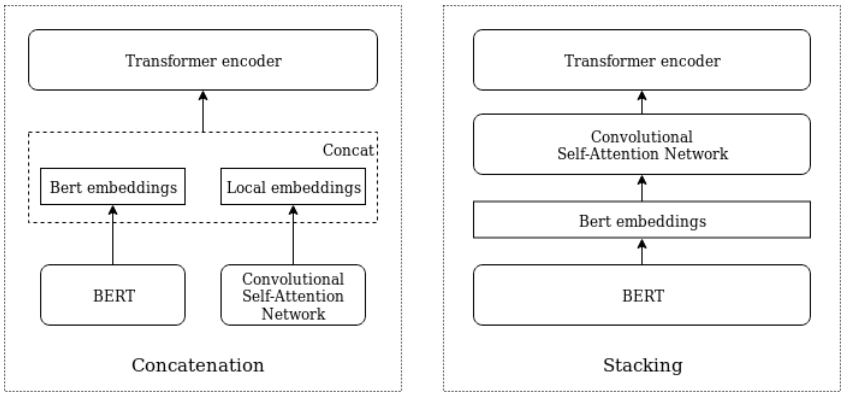


Figure 4.4: Two ways of the integration of the BERT-conditioning with the Convolutional Self-Attention

Pointer Generator, and capabilities to process long texts. In Figure 4.3 we show the schema of the basic model with the BERT-conditioned and decoder.

4.2.6 *Integration of BERT and Convolutional Self-Attention*

We evaluated two different ways to integrate the BERT-conditioning with the convolutional self-attention of the model's encoder (Figure 4.4).

Stacking. Our first approach comprises directly feeding the BERT-generated embeddings to the convolutional self-attention Transformer encoder. A potential problem with this approach is that convolutional self-attention is assumed to be beneficial when applied in the lower layers as its locality modeling feature should help in modeling of local dependencies (e. g. syntax). At the same time, BERT is a hierarchical model where the last layers target very high-level patterns in the sequences (e. g. semantics). Hence, we assume that the application of the network detecting the low-level patterns on BERT's output can undermine its generalization abilities.

Concatenation. Because of the considerations raised above, in parallel, we develop a second approach which we call Concatenation. We split the original convolutional self-attention Transformer encoder into two networks where the first one uses only convolutional self-attention and the second original self-attention (identical to the Transformer encoder). Then we feed the original sequences into BERT and the convolutional self-attention network in parallel. The resulting embedding vectors are concatenated and fed into the Transformer encoder. In this way, we model the locality at the lower layers of the encoder at the cost of a smaller depth of the network (assuming the same number of layers).

4.2.7 *BERT-Conditioned Generator*

Different from the encoder and decoder, the conditioning of the generator is a more complicated task. The analogous logic would be to use the generator layer of BERT instead of the one trained from scratch. However, we assume that the generator weights are very sensible to the text domain, hence, the usage of the pre-trained decoder will most likely decrease the accuracy. We decided to follow another approach which can be generally called the modification of the predicted probability distribution. From Section 4.2.2 we know that Language models are prone to score the naturally generated texts as more probable compared to random and shuffled text. The hypothesis that we want to test is that this property of the language models may be used for the enhancement of the total accuracy via the modification of the generator output.

Two different strategies are possible to accomplish it: post-processing and training-based. First, we consider the post-processing approach as more easy and intuitive. The algorithm is based on the passing of previous summary draft outputs through BERT and calculation of the probability distribution for the next token. The BERT's result is compared with the distribution generated by our decoder. All tokens which get a BERT score smaller the particular threshold (calculated as the percentile from the LM distribution) are multiplied by a parameter α ($0 < \alpha < 1$). This procedure should force the trained model to generate only tokens which were decided by a language model to be human-like. In the real implementation, we restricted this procedure only to N tokens with the highest probability to be the next token. This trick will not change the generated summary if N is large but will significantly reduce the computational time.

The second strategy is to apply a similar procedure while training. Different from the post-processing case, here we increment by some factor the probability of tokens which gained the high BERT score. As for language models, the next token probability is usually higher for the natural tokens (our target) the loss should be less when the model's output is close to the global optimum and larger when is it far from it. We assume that this can improve model convergence.

5 DATASETS DESCRIPTION

Dataset	Size	Vocab.	Src. len.	Tgt. len.	src-tgt ratio
CNN/DM	312,084	27,696	895	63	14.2
SwissText	100,000	46,433	918	54	17

Table 5.1: General CNN / Daily Mail and SwissText datasets statistics

We aim to develop a system that works in a language-independent way, i. e. it assumes the upstream components being available in the respective language or being a language-independent, such as the multi-lingual version of BERT which does not need language-specific tuning. However, since most summarization datasets are in English, we use English as the default language for evaluation. Additionally, we include German for validation purposes.

5.1 CNN / DAILY MAIL

In this work, we decided to use the CNN / Daily Mail dataset [8, 10]. It contains the collection of news articles paired with multi-sentence summaries published on the `CNN.com` and `dailymail.com` websites. Our logic behind this decision is that this is the de facto standard for training summarization models which allows us to compare our results with the previous studies. Besides, the news articles are one of the main domains where our summarization system is to be used.

This dataset exists in two versions: original data (non-anonymized version) and anonymized which replaces every named entity with an anonymous identifier. We use the non-anonymized data as our application requires the model to be able to copy the named entities from the original texts and as was used for training of the most recent state-of-the-art models [9]. The raw dataset consists of separate text files each representing a single article or a summary. In this work, we used already pre-processed data supplied by [11]. It has 287,226 training pairs, 13,368 validation pairs, and 11,490 test pairs. with the average lengths of an article and a summary 781 tokens and 56 tokens respectively (Source-Target ratio 13.9).

To align the data with the vocabulary of BERT used in the language conditioning experiments we tokenized it using the BPE-based Word-Piece tokenizer [12]. That increased the average length of source texts by 14.6% and the length of summaries by 12.5% which led to the

slightly higher Source-Target ration. We assume in this work this does not complicate the task dramatically and any drop in the model's performance is to be alleviated by our contributions to the model architecture. As all samples in BERT's training data are prepended the special token "[CLS]", we follow the same strategy and add it to every source text in out dataset. You can see the example of the training pair on the text box 5.1. Then we preprocessed this data to the OpenNMT format using the standard preprocessing scripts of the library that created the vocabulary consisting of all the tokens in the training and validation datasets (Table 5.1). Due to the high variance in the length of the text that can worsen the model's convergence and the limitation of BERT accepting only sequences of the maximum length 512 tokens we also prepared the clipped version of the training and validation datasets with each article truncated to 512 tokens. In the experiments on BERT windowing, we use the full-text version.

[CLS] london - l ##rb - cnn - rr ##b - a 19 - year - old man was charged wednesday with terror offenses after he was arrested as he returned to britain from turkey , london ' s metropolitan police said . ya ##hya rashid , a uk national from northwest london , was detained at luton airport on tuesday after he arrived on a flight from istanbul , police said . he ' s been charged with engaging in conduct in preparation of acts of terrorism , and with engaging in conduct with the intention of assisting others to commit acts of terrorism . both charges relate to the period between november 1 and march 31 . rashid is due to appear in westminster magistrates ' court on wednesday , police said . cnn ' s lindsay isaac contributed to this report .	london ' s metropolitan police say the man was arrested at luton airport after landing on a flight from istanbul . he ' s been charged with terror offenses allegedly committed since the start of november .

Text box 5.1: Sample example of the input text(left) and summary(right) from the preprocessed and tokenized CNN / Daily Mail dataset

5.2 SWISSTEXT DATASET

To evaluate the efficiency of the model in the multi-lingual multidomain environment in addition to the CNN / Daily Mail dataset we conducted some experiments on the German SwissText dataset. This dataset was created for the 1st German Text Summarization Challenge at 4th Swiss Text Analytics Conference - SwissText 2019 [13]. It was designed to explore different ideas and solutions regarding abstractive summarization of German texts. To the best of our knowledge, it is the first long-text summarization dataset in the German language publicly available. The data was extracted from the German Wikipedia and represents mostly autobiography articles and definitions of various concepts.

The dataset was tokenized by the multilingual WordPiece tokenizer [12] and preprocessed by the OpenNMT library in the same way as the CNN / Daily Mail dataset. It was split into the training, validation and testing datasets containing 90,000, 5,000 and 5,000 samples respectively. From the general statistics (Table 5.1) we can see that even having less number of samples that the CNN / Daily Mail dataset the preprocessing results in a larger vocabulary. We assume this is the result of the morphological richness of the German language which leads to the higher word split rate (how often a word is split into the pieces). Without taking into account the domain of the data this property makes the German summarization task more complex. Besides, the dataset has a higher length of input texts and lower length of the target summaries which results in a higher Source-Target ratio. That makes this dataset very suitable for our experiments on windowing long texts, otherwise, texts are truncated to 512 tokens.

6 IMPLEMENTATION

Based on the requirements and the model description we implemented our summarization system. It is based on the PyTorch and the OpenNMT NLP library [23]. The implemented extractive and abstractive summarization modules were combined into a microservice tool using the Flask web framework. The extractive system shorts any text to 512 tokens which are later fed into the abstractive system to generate the final summary in the Turtle format. The system was containerized using Docker and deployed on the DEMO server which communicates with the Postman interface via POST requests.

The extractive summarizer is currently implemented for the English language only. It uses the uncased BERT-base model and the custom-made sentence level tokenizer.

The abstractive summarizer supports now two languages using the respective realizations of BERT: base uncased and base multilingual cased. For the experiments on the Generator conditioning, we use the version of BERT with the generator layer trained on the Masked language modeling task. For ROUGE evaluation we use the files2rouge library and calculate ROUGE recall, precision and F-score for unigrams, bigrams, and Longest Common Subsequences (LCS). In the experimental part, we report only F-scores as the most adapted metric by the state-of-the-art summarization research that depends on both recall and precision.

Bellow, we provide a general description of the environments and the values of hyper-parameters used in our experiments.

6.1 ENVIRONMENT

Computational Environment:

- **Main:** server with 2 Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz, 377 GB RAM and 7 NVIDIA GeForce GTX 1080 Ti
- **Additional:** An HPC of 20x GPU-Nodes each with Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz, 512 GB RAM and 2 NVIDIA Tesla P100

Programming Environment:

- Ubuntu 16.04
- Python 3.6
- Pytorch 1.0.1
- Transformers 1.1.0
- Pyrouge 0.1.3

- files2rouge 2.0.0
- Numpy 1.16.3
- Flask 1.1.1
- Postman
- Various additional libraries fully specified in the requirements.txt file

6.2 PROJECT STRUCTURE

The implementation is separated into 2 distinguished summarizing systems synchronized via Flask API. The extractive summarization system consists of two scripts: one for calculating all the TF-IDF scores for a given corpus and another for the generating the summaries according to the algorithm described in Chapter 4. The abstractive summarization system is built upon the OpenNMT library with several modifications to fulfill our goals. The following classes were added to the OpenNMT library:

- **BertEncoder** – BERT-based encoder.
- **BertTransformerEncoder** – BERT combined with a regular Transformer encoder.
- **BertTransformerDecoder** – the embedding layer of BERT combined with a regular Transformer Decoder.
- **ConvMultiHeadedAttention** – the realisation of self-attention with 1D and 2D convolution.
- **ConvTransformerEncoderLayer** – the layer of Transformer encoder with convolutonal Self-Attention.
- **ConvTransformerEncoder** – the Transformer encoder with the convolutional Self-Attention in the first layers.
- **BertGenerator** – BERT based generator without pointer.
- **GlobalModel** – the global class storing the deployed BERT model and the vocabulary.
- **BertConvTransformerEncoder** – encoder combining BERT with Convolutional Self-Attention.

Also, the following files were modified:

- **opts.py** – to add new terminal parameters to support the aforementioned classes.
- **model_builder.py** – to initialize the properties of the **GlobalModel** class.

- **translator.py** – to connect the BERT-based Generator to the BERT-based decoder and run the post-processing Generator conditioning experiments.

6.3 EXPERIMENTAL SETUP

Most of the experiments were conducted using the hyperparameters specified bellow. They were primarily taken from the recommendations of OpenNLP documentation for the summarization task [21]. The depth, size of the hidden state and the number of training steps were shrunk to accelerate the training procedure and a model's convergence. Different form [9] here we do not reuse some attention layer from the Decoder but train a new copy attention layer from scratch. For training, we used Adam optimizer with the Noam decay method [22]. For the regularization, we use dropout and label smoothing. Each neural network was trained on one of the GPUs specified in the Section 6.1 for 90,000 training steps (around 2-3 days). The generation of the summary is made via the Beam search algorithm.

Bellow, we provide a general description of the environments and the values of hyper-parameters used in our experiments.

Main hyperparameters:

- Shared embeddings
- Truncation length = 512
- Batch size = 4096 tokens
- Validation batch size = 1 sample
- Hidden state size = 256
- Word vector size = 256
- Number of layers in Encoder = 3
- Number of layers in Decoder = 3
- Number of Self-Attention heads = 4
- Maximal number of generator batches = 2
- Number of training steps = 90,000
- Validation every 2,000 steps
- Label smoothing = 0.1
- Dropout rate 0.2
- Adam beta1 = 0.9, beta2 = 0.998
- Learning rate = 2 (initial)
- Number of warmup steps = 8000
- Beam size = 15 or 4
- Minimal summary length = 30
- Maximal summary length = 512
- Stepwise coverage penalty (alpha=0.9, beta=5)

For the final comparison of our model with other state-of-the-art models the hidden state size was set to 512, the number of Transformer layers to 6 and the number of self-attention heads to 8 to adjust the architecture to out main baseline (Transformer with Pointer-Generator and Coverage Penalty [11]). The models were trained for 200,000 training steps.

7 EXPERIMENTAL RESULTS

7.1 BASELINE MODEL

Model	R-1	R-2	R-L
Transformer	24.82	6.27	22.99
Transformer + Beam Search	25.04	7.83	23.08
Transformer + Pointer Gen.	31.95	14.49	30.02
Transformer + Pointer Gen. + Beam Search	**36.04**	**17.14**	**33.34**

Table 7.1: Effect of Pointer Generator and beam search on Transformer on the CNN / Daily Mail dataset.

The results on the SwissText dataset (Table 7.2) shows the same tendency and confirms the importance of Pointer Generator and beam search. Later, we use these models as our baselines. In the first set of experiments, we evaluate the effect of the copy mechanism and the beam search on the vanilla Transformer model. The beam size was set to 15. The results on the CNN / Daily Mail dataset is presented in Table 7.1. First, we see that both the Pointer Generator and beam search lead to an increase in the ROUGE scores. The effect of Pointer Generator is very large which is the evidence of its much importance for our task.

7.2 EXTRACTIVE STAGE

The main purpose of the extracting module is to be used with log texts which can not be correctly handled by the neural networks. However, we assumed that the application of such extractive pre-stage on the smaller texts can also increase the total scores. We pre-processed the CNN / Daily Mail training and testing datasets keeping 50 and 70 percent of the sentences. We trained Transformer with Pointer Generator (CopyTransformer) models following 2 strategies: applying extraction procedure only in testing and in both testing and training. The evaluation of the models was made via the beam search (Table 7.3).

The model trained on the full texts and evaluated on extractively summarized texts shows the lowest ROUGE scores. The usage of extraction in training achieves the higher scores which are still smaller than the full-text baseline. Here we observe the particular dynamics

Model	R-1	R-2	R-L
Transformer	36.40	20.69	34.14
Transformer + Beam Search	37.37	22.58	35.08
Transformer + Pointer Gen.	39.44	25.11	37.16
Transformer + Pointer Gen. + Beam Search	**40.59**	**26.33**	**37.58**

Table 7.2: Effect of Pointer Generator and beam search on Transformer on the SwissText dataset.

Model	R-1	R-2	R-L
CopyTransformer	**36.04**	**17.14**	**33.34**
+Extraction in training time (50%)	35.30	16.72	31.98
+Extraction in training time (70%)	35.68	16.65	32.91
+Extraction in testing time (50%)	34.76	16.14	32.11
+Extraction in testing time (70%)	33.65	15.46	31.09

Table 7.3: Effect of the extraction pre-stage on the ROUGE scores on the CNN / Daily Mail dataset.

that ROUGE increases with the increase of the input text length. We also observed that in the case of 70 percent extraction rate the ROUGE precision scores were the highest, but, as F-scores are still lower than the baseline, we consider this model inferior. That proves that on the CNN / Daily Mail dataset it is more beneficial to not use any extraction and rely only on the abstractive methods. Hence, we conclude that the extractive summarization module should be mainly used when the text is the size not processable by neural models.

7.3 LOCALITY MODELING

To evaluate the effect of convolution on self-attention we introduce it in the first layer of the encoder. We use the same kernel sizes as in [26]. As a baseline, we use our implementation of CopyTransformer.

The results are presented in Table 7.4. We see that both convolutions over tokens and over attention heads improve the ROUGE scores. Standard convolution outperformed circular convolution on ROUGE-1, ROUGE-2, and ROUGE-L by 0.06, 0.13 and 0.09 percent, respectively. We also investigated the effect of the window size of the 1-dimensional convolution on ROUGE scores (Figure 7.1). In contrast to findings in Machine Translation, we found that size 13 returns the best result for the summarization task.

Model	R-1	R-2	R-L
CopyTransformer	31.95	14.49	30.02
+ 1D conv.	32.62	14.99	30.74
+ 2D conv.	**32.72**	**15.12**	**30.85**
+ 2D round conv.	32.68	15.01	30.76

Table 7.4: Ablation study of model with convolutional self-attention on the CNN / Daily Mail dataset. The kernel sizes are 11 and 3.

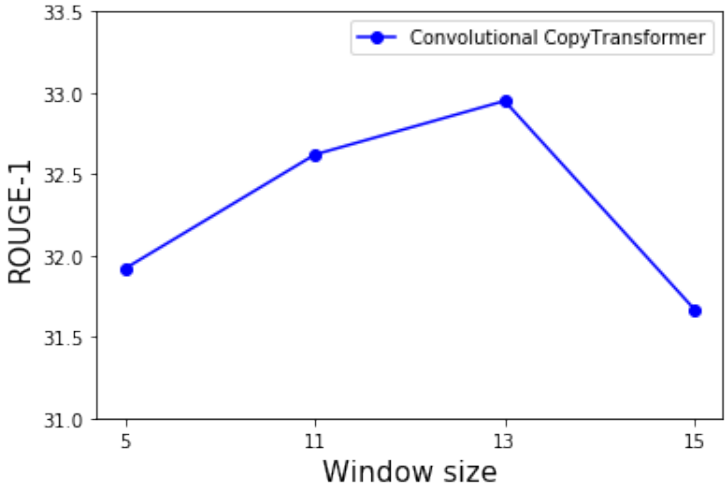


Figure 7.1: Effect of the window size on ROUGE-1

7.4 LANGUAGE MODEL CONDITIONING

To find the optimal architecture of the BERT-based abstractive summarizer we conducted an ablation study (Table 7.5). On CNN/Daily Main dataset we test 3 different models: BERT encoder+Transformer decoder, BERT-Transformer encoder + Transformer decoder and BERT-Transformer encoder+BERT-Transformer decoder. The version of BERT used in the experiments is BERT-base. As the baseline, we use Transformer without the Pointer Generator. From the results, we observe that BERT improves the efficiency of the model when it is used in both encoder and decoder. Besides, the linear dimensionality reduction from the BERT hidden size to the hidden size of our model showed to work better than non-linear transformation via the RELU function. Furthermore, BERT in the encoder is more effective when it is used to produce embeddings to be used by the standard Transformer encoder than when it is used solely as an encoder. Even without the Pointer Generator, our model outperformed the CopyTransformer baseline by 1.28, 0.5, and 1.24 on ROUGE-1, ROUGE-2, and ROUGE-L

Model	R-1	R-2	R-L
Transformer	24.82	6.27	22.99
CopyTransformer	31.95	14.49	30.02
Bert Enc. + Transformer Dec.	31.3	13.37	29.46
Bert Enc. with RELU + Transformer Dec.	29.84	12.3	28.17
Bert-Transformer Enc. + Transformer Dec.	32.5	14.68	30.68
Bert-Transformer Enc. and Dec.	**33.23**	**14.99**	**31.26**
Transformer (full)	23.18	5.15	21.48
Bert-Transformer Enc. + Trans. Dec. (full)	**31.51**	**14.1**	**29.77**

Table 7.5: Ablation study of the BERT-based model on truncated and original CNN / Daily Mail dataset.

Model	R-1	R-2	R-L
Transformer	36.40	20.69	34.14
CopyTransformer	39.44	25.11	37.16
Bert-transformer Enc. + Transformer Dec.	**44.01**	**29.60**	**41.65**
Bert-transformer Enc. and Dec.	43.22	29.01	40.84
Transformer (full)	34.76	18.65	32.61
Bert-transformer Enc. + Trans. Dec. (full)	**45**	**30.49**	**42.64**

Table 7.6: Ablation study of the BERT-based model on the truncated and original SwissText dataset.

To evaluate our BERT-windowing method, we conducted the experiments on the full non-truncated text. Our approach outperforms the baseline, which proves that the method can be successfully applied to the texts longer than 512 tokens. The final performance of this model is still lower than of the model trained on the truncated text, but, as the same pattern can be observed for the baselines, we assume this relates to the specifics of the given dataset that is prone to have important information in the first sentences of a text.

On SwissText data we evaluated 2 models with the Bert-Transformer encoder and Transformer and BERT-Transformer decoders (Table 7.6). The introduction of BERT into Transformer increased the ROUGE-1, ROUGE-2 and ROUGE-L scores by 7.21, 8.91, 7.51 percent. At the same time, the usage of BERT in the decoder decreased the overall score. We assume that the reason behind this is that in multilingual BERT, due to its language-independence, the embedding matrix outputs less precise

Model	α parameter	R-1	R-2	R-L
Transformer	1.0	**24.82**	**6.27**	**22.99**
Transformer+BERT-generator	0.9	24.25	5.91	22.44
	0.7	23.47	5.35	21.72

Table 7.7: Experimental results on the CNN / Daily Mail dataset for post-processing generator conditioning approach.

Model	Usage strategy	R-1	R-2	R-L
Transformer	-	**24.82**	**6.27**	**22.99**
Transformer+BERT-generator	train	23.9	5.87	22.17
	train and test	23.62	5.73	21.94

Table 7.8: Experimental results on the CNN / Daily Mail dataset for the approach of generator conditioning during training.

contextualized representations which undermines their benefits for the summarization task.

On the non-truncated texts, usage of the Bert-transformer encoder increased the ROUGE scores by 10.23, 11.84 and 10.03 percent. Furthermore, it gives us higher scores compared to the same model trained on truncated texts. This shows that information in this dataset is spread more uniformly than in the CNN / Daily Mail dataset and proves the high efficiency of our BERT-windowing method in the detection of this information.

In our next experiments, we evaluate our two strategies of generator conditioning. Table 7.7 presents the results of the application of the post-processing to the baseline's output. First, we found that the number of tokens N and the percentile do not affect the results much, as we had the same results with different values of these parameters. Then, we found that, unfortunately, post-processing conditioning worsens the model's performance and the ROUGE score is higher, the closer α parameter to 1.

Then, we applied our conditioning procedure during training (Table 7.8). This approach performs better when we use conditioned generator also in the testing phase. Unfortunately, this method also did not bring any improvement. Hence, we conclude that generator-conditioning should be dismissed and only encoder and decoder conditioning is to be used in the final model.

Integration	Model	R-1	R-2	R-L
Stacking	BERT+Transformer	35.28	**17.12**	33.31
	BERT+Conv. Transformer	**35.4**	16.82	**33.31**
Concatenation	BERT+Transformer	34.82	16.46	32.79
	BERT+Conv. Transformer	**35.26**	**16.79**	**33.22**

Table 7.9: Experimental results on the CNN / Daily Mail dataset for different strategies of integration of pre-trained models with convolutional Self-Attention

7.5 INTEGRATION STRATEGIES

For the evaluation of the integration strategies, we trained two models with the respective BERT-based baselines. Both models have in their encoder two Transformer layers and one convolutional Transformer layer placed on top of BERT or in parallel, respectively (Table 7.9).

The method of stacking does not provide any significant improvement. With the introduction of the convolutional self-attention, only ROUGE-1 increased by 0.12 percent, while ROUGE-2 dropped by 0.3 and ROUGE-L remained the same. Considering that in many domains ROUGE-2 maximally correlates with human assessment (see Section 2.2.1), we dismiss this method. The concatenation strategy convolution is shown to be much more efficient, increasing ROUGE scores by 0.44,0.33 and 0.43 percent. This confirms our hypothesis that locality modeling is the most efficient when applied at the bottom on the non-contextualized word representations. Unfortunately, this model failed to outperform the stacking baseline. We conclude that the concatenating architecture undermines the performance of the Transformer model, and the convolutional self-attention is not beneficial when used together with pre-trained language models. Hence, we decide to train our two final models separately.

7.6 MODELS COMPARISON

For the final comparison of our model with other state-of-the-art methods we set the hidden state to 512, the number of Transformer layers in the encoder and layers to six and the number of self-attention heads to 8. Hence, our baseline is smaller compared to the original CopyTransformer [11], which may be the reason why it performs slightly worse (Table 7.10). BERT-conditioning was used in both encoder and decoder. The generation of the summary is made via the beam search algorithm with the beam size set to four. Finally, the generated summaries were detokenized back to the sequences of words separated by spaces.

Method	R-1	R-2	R-L
BiLSTM + Pointer-Generator + Coverage [9]	39.53	17.28	36.38
ML + Intra-Attention [7]	38.30	14.81	35.49
CopyTransformer [11]	39.25	17.54	36.45
Bottom-Up Summarization [11]	41.22	18.68	38.34
One-Stage BERT [25]	39.50	17.87	36.65
Two-Stage BERT [25]	41.38	19.34	38.37
ML + Intra-Attention + RL [7]	39.87	15.82	36.90
Key information guide network [30]	38.95	17.12	35.68
Sentence Rewriting [29]	40.88	17.80	38.54
BertSumAbs [45]	**41.72**	**19.39**	**38.76**
CopyTransformer (our implementation)	38.73	17.28	35.85
Convolutional CopyTransformer	38.98	17.69	35.97
BERT+CopyTransformer	**40**	**18.42**	**37.15**

Table 7.10: ROUGE scores for various models on the CNN / Daily Mail test set.The first section shows different state-of-the-art models. The second section presents our models and the main baseline.

Method	R-1	R-2	R-L
CopyTransformer (our implementation)	39.5	22.36	36.97
Convolutional CopyTransformer	40.54	23.62	38.06
BERT+CopyTransformer (enc.)	**42.61**	**25.25**	**39.85**

Table 7.11: ROUGE scores for our models on the SwissText test set.

For the BERT-based model, we set the minimum length of a generated summary to 55, as we found that without such restriction the model was prone to generate shorter sequences than in the test dataset. The model outperformed the baseline by 1.27 on ROUGE-1, 1.14 on ROUGE-2 and 1.3 on ROUGE-L. This is larger than scores of One-Stage BERT but still less than the two-stage and BertSumAbs models. For the convolutional CopyTransformer we use convolutional self-attention in the first three layers of the encoder. It increased ROUGE-1, ROUGE-2 and ROUGE-L by 0.25, 0.41 and 0.12.

Furthermore, we present, to our knowledge, the first publicly available result for the SwissData dataset (Table 7.11). All parameters are equal to the CNN / Daily Mail baseline. BERT-conditioning was used only in the encoder. The networks were trained on the truncated texts in 90,000 training steps. From the results we see that the convolutional CopyTransformer showed much more efficiency than on CNN / Daily

Mail dataset, outperforming the baseline by 1.04 percent on ROUGE-1, 1.26 on ROUGE-2 and 1.09 on ROUGE-L. The BERT-based model again achieved the highest ROUGE scores outperforming the baseline for more than 2 percent.

7.7 QUALITATIVE ANALYSIS

To compare the trained models we conducted a qualitative analysis comparing the generated summaries. Text box 7.1 includes the reference summary and those generated by different models. Comparing the first sentence we see that the vanilla Transformer model performed the worse by copying only part of the original sentence omitting some characters in the word "meteorological". The model with convolution has copied the whole sentence but still made a spelling error. Finally, only the BERT-based model succeeded to generate the right token "meteorological". Also, we see that while the BERT-based model's summary conveys the same meaning as the gold summary, the convolutional Transformer generates one and Transformer two sentences with the information not present in the gold summary. Overall, on the given example all models provided a summary of extractive nature and only the BERT-based model shows some level of abstractiveness merging parts of the two sentences into the single one (in the second summary's sentence). This is far from the gold summary where every sentence in some way paraphrases the original text. Hence, given this particular example, our models demonstrate some explicit improvements, but the abstractive summarization task remains challenging.

researchers are developing a computer that can write weather forecasts . it takes meteorological data and writes a report designed to mimic a human . this process is known as ‘ natural language generation ’ - lrb - nlg - rrb - . a prototype system will be tested on the bbc website later this year .

researchers from london and edinburgh have developed a computer that can collateological information . these computer - generated weather updates are being tested by scientists at heriot - watt university and university college london . if the project is successful , a prototype system will be tested by generating local weather reports on the bbc ’ s website . currently , the bbc website features 10 reports written by meteorologists .

researchers from london and edinburgh have developed a computer that can collate meterological information and then produce forecasts as if they were written by a human . it uses a process known as ‘ natural language generation ’ - lrb - nlg - rrb - . these computer - generated weather updates are being tested by scientists at heriot - watt university and university college london . if the project is successful , a prototype system will be tested by generating local weather reports on the bbc ’ s website .

researchers from london and edinburgh have developed a computer that can collate meteorological information and produce forecasts as if they were written by a human . using met office data , it uses a process known as ‘ natural language generation ’ - lrb - nlg - rrb - . if the project is successful , a prototype system will be tested by generating local weather reports on the bbc ’ s website .

Text box 7.1: Comparison of the output of models on an example form CNN / Daily Mail testset. Surface realisation mistakes are highlighted in green and a typical abstractive feature, illustrating re-arranging of the sentence is highlighted in blue. From top to bottom: gold summary, Transformer, convolutional Transformer, BERT-transformer.

8 CONCLUSIONS

In this thesis, we developed an abstractive text summarization system based on neural networks and exploiting several new ideas to increase its accuracy. The system represents the modification of the OpenNMT library built upon the PyTorch framework and can summarize texts of any length. The source code of our system is publicly available.[1] A functional service based on the model is currently being integrated, as a summarization service, in the platforms Lynx [62], QURATOR [59] and European Language Grid [61]. Besides, based on the results obtained in this thesis we published a paper on the LREC 2020 conference. [60]

The main dataset used for the training of our models is CNN / Daily Mail. To establish the suitability of our model to languages other than English and domains other than news, we also trained and tested our model on the German SwissText dataset. All developed models were evaluated and compared with a baseline and competing state-of-the-art models using the ROUGE scores evaluation. Our results on the SwissText dataset constitutes one of the first available results for this dataset.

In this thesis, we did several major scientific contributions. First, we presented several new abstractive text summarization models incorporating the idea of conditioning of encoder, decoder, and generator on the pre-trained language model. In our analysis, we found that the model BERT fits our needs the best. The BERT conditioning showed huge improvement when used in encoder and decoder but was not useful for generator conditioning

Second, we developed a model explicitly introducing locality modeling. It uses the convolutional self-attention in the encoder instead of the vanilla self-attention to better model local dependencies. We tested several strategies to combine it with the BERT-based model but found that as both of them are useful only at the very bottom layer their integration does not bring any gain. As the BERT-based model showed better results than convolutional self-attention-based we decided to use it in the release version of our summarization system.

Third, we proposed a method to alleviate the BERT's input size limitation by setting the maximal length of a sequence to 512 tokens. Our method processes the input sequence in windows, the resulted distributional representations are then combined to be used by the rest part of the network. The evaluation demonstrated its efficiency in the processing of relatively long input texts.

1 https://github.com/axenov/BERT-Summ-OpenNMT

And last but not the least, to be able to handle significantly long documents (that are much larger than 512 tokens) we developed the extractive sentence-level summarization module. This module is based on the TF-IDF sentence-level summarization and uses BERT's next sentence prediction capability to increase the consistency of the result summaries. In the deployed version of the system, we summarize any long text to approximately 512 tokens and feed it into the abstractive model to get the final output.

Therefore, in this work, we achieved all the objectives stated in the introduction chapter. Based on the results obtained in this work we are going to extend this system to the broader scope. First of all, we will continue our research on the developed model to find new more successful strategies of generator conditioning and integration of language model conditioning with locality modeling. Second, we will adapt this system to the multi-document summarization. The particular challenge we want to solve is to summarise several documents on the same topic while keeping the low redundancy and high consistency in the summary. As the multi-document summarization task lacks big open-source datasets, we think that the extension and adaptation of our extractive module would be the most promising strategy. Apart from that, the exploration of the system's hyperparameters is needed to make the combination of extractive module and abstractive module the most efficient. Within this work, we also would like to explore the usefulness of semantic methods and knowledge graphs for summarization. Finally, we plan to adapt the system to the structured input such as XML or RDF documents, possibly, via incorporation of the hierarchical attention methods.

BIBLIOGRAPHY

[1] Jie Hao, Xing Wang, Baosong Yang, Longyue Wang, Jinfeng Zhang, and Zhaopeng Tu. "Modeling Recurrence for Transformer." In: (Apr. 5, 2019). arXiv: http://arxiv.org/abs/1904.03092v1 [cs.CL].

[2] Universal approximation theorem. *Universal approximation theorem — Wikipedia, The Free Encyclopedia*. [Online; accessed 15-October-2019]. 2019 (cit. on p. 10).

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Kaiser, and Illia Polosukhin. "Attention is All You Need." In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010. ISBN: 978-1-5108-6096-4 (cit. on pp. 2, 15–17, 28).

[4] Felix Wu, Angela Fan, Alexei Baevski, Yann N. Dauphin, and Michael Auli. *Pay Less Attention with Lightweight and Dynamic Convolutions*. Available online (arXiv). 2019. arXiv: 1901.10430 [cs.CL].

[5] Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. "Generating Wikipedia by Summarizing Long Sequences." In: *International Conference on Learning Representations*. 2018 (cit. on pp. 20, 27).

[6] Alec Radford. "Improving Language Understanding by Generative Pre-Training." In: Available online. 2018 (cit. on p. 18).

[7] Romain Paulus, Caiming Xiong, and Richard Socher. "A Deep Reinforced Model for Abstractive Summarization." In: *International Conference on Learning Representations*. 2018 (cit. on pp. 20, 53).

[8] Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. "Teaching Machines to Read and Comprehend." In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'15. Montreal, Canada: MIT Press, 2015, pp. 1693–1701 (cit. on p. 39).

[9] Abigail See, Peter J. Liu, and Christopher D. Manning. "Get To The Point: Summarization with Pointer-Generator Networks." In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (2017) (cit. on pp. 19, 39, 45, 53).

[10] Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gülçehre, and Bing Xiang. "Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond." In: *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 280–290 (cit. on pp. 19, 39).

[11] Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. "Bottom-Up Abstractive Summarization." In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2018, pp. 4098–4109 (cit. on pp. 20, 25, 39, 46, 52, 53).

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186 (cit. on pp. 9, 18, 33, 39, 41).

[13] "Swiss Text 2019: German Text Summarization Challenge." In: Available online. 2019 (cit. on p. 41).

[14] Wilson L. Taylor. "C̈loze Procedure̋: A New Tool for Measuring Readability." In: *Journalism Bulletin* 30.4 (1953), pp. 415–433. eprint: https://doi.org/10.1177/107769905303000401 (cit. on p. 9).

[15] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. "A Neural Probabilistic Language Model." In: *J. Mach. Learn. Res.* 3 (Mar. 2003), pp. 1137–1155. ISSN: 1532-4435 (cit. on p. 9).

[16] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. "Language Models are Unsupervised Multitask Learners." In: Available online. 2018 (cit. on pp. 9, 18).

[17] G. Cybenko. "Approximation by superpositions of a sigmoidal function." In: *Mathematics of Control, Signals and Systems* 2.4 (1989), pp. 303–314. ISSN: 1435-568X (cit. on p. 10).

[18] Yann LeCun, Y. Bengio, and Geoffrey Hinton. "Deep Learning." In: *Nature* 521 (May 2015), pp. 436–44 (cit. on p. 11).

[19] Tian Shi, Yaser Keneshloo, Naren Ramakrishnan, and Chandan K. Reddy. *Neural Abstractive Text Summarization with Sequence-to-Sequence Models*. Available online (arXiv). 2018. arXiv: 1812.02303 [cs.CL] (cit. on pp. 19, 25).

[20] Chin-Yew Lin. "ROUGE: A Package for Automatic Evaluation of Summaries." In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 74–81 (cit. on pp. 21, 25).

[21] Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. *OpenNMT-py: Summarization*. [Online; accessed 15-October-2019]. 2018 (cit. on p. 45).

[22] Diederik Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization." In: *International Conference on Learning Representations* (Dec. 2014) (cit. on p. 45).

[23] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. "OpenNMT: Open-Source Toolkit for Neural Machine Translation." In: *Proceedings of ACL 2017, System Demonstrations*. Vancouver, Canada: Association for Computational Linguistics, July 2017, pp. 67–72 (cit. on p. 43).

[24] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[25] Haoyu Zhang, Jingjing Cai, Jianjun Xu, and Ji Wang. "Pretraining-Based Natural Language Generation for Text Summarization." In: *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 789–797 (cit. on pp. 20, 34, 35, 53).

[26] Baosong Yang, Longyue Wang, Derek F. Wong, Lidia S. Chao, and Zhaopeng Tu. "Convolutional Self-Attention Networks." In: *Proceedings of the 2019 Conference of the North* (2019) (cit. on pp. 2, 29, 30, 48).

[27] Alessandro Raganato and Jörg Tiedemann. "An Analysis of Encoder Representations in Transformer-Based Machine Translation." In: *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 287–297 (cit. on pp. 29, 30).

[28] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. "Deep Contextualized Word Representations." In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 2227–2237 (cit. on p. 18).

[29] Yen-Chun Chen and Mohit Bansal. "Fast Abstractive Summarization with Reinforce-Selected Sentence Rewriting." In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 675–686 (cit. on p. 53).

[30] Chenliang Li, Weiran Xu, Si Li, and Sheng Gao. "Guiding Generation for Abstractive Text Summarization Based on Key Information Guide Network." In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 55–60 (cit. on pp. 20, 53).

[31] Tobias Domhan. "How Much Attention Do You Need? A Granular Analysis of Neural Machine Translation Architectures." In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 1799–1808 (cit. on pp. 2, 16).

[32] Felix Wu, Angela Fan, Alexei Baevski, Yann Dauphin, and Michael Auli. "Pay Less Attention with Lightweight and Dynamic Convolutions." In: *International Conference on Learning Representations*. 2019 (cit. on pp. 2, 16).

[33] Ian Tenney, Dipanjan Das, and Ellie Pavlick. "BERT Rediscovers the Classical NLP Pipeline." In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (2019) (cit. on p. 29).

[34] Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. "A Discourse-Aware Attention Model for Abstractive Summarization of Long Documents." In: *NAACL-HLT*. 2018 (cit. on p. 20).

[35] Arman Cohan and Nazli Goharian. *Revisiting Summarization Evaluation for Scientific Articles*. Available online (arXiv). 2016. arXiv: 1604.00400 [cs.CL] (cit. on p. 21).

[36] John M. Conroy and Hoa Trang Dang. "Mind the Gap: Dangers of Divorcing Evaluations of Summary Content from Linguistic Quality." In: *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*. Manchester, UK: Coling 2008 Organizing Committee, Aug. 2008, pp. 145–152 (cit. on p. 21).

[37] Bonnie Dorr, Christof Monz, Stacy President, Richard Schwartz, and David Zajic. "A Methodology for Extrinsic Evaluation of Text Summarization: Does ROUGE Correlate?" In: *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for*

Machine Translation and/or Summarization. Ann Arbor, Michigan: Association for Computational Linguistics, June 2005, pp. 1–8 (cit. on p. 21).

[38] Gabriel Murray, Steve Renals, and Jean Carletta. "Extractive summarization of meeting recordings." In: *INTERSPEECH 2005 - Eurospeech, 9th European Conference on Speech Communication and Technology, Lisbon, Portugal, September 4-8, 2005*. 2005, pp. 593–596 (cit. on p. 21).

[39] Jonas Sjöbergh. "Older versions of the ROUGEeval summarization evaluation system were easier to fool." In: *Information Processing & Management* 43.6 (2007). Text Summarization, pp. 1500 –1505. ISSN: 0306-4573 (cit. on p. 21).

[40] Elena Lloret, Laura Plaza, and Ahmet Aker. "The challenging task of summary evaluation: an overview." In: *Language Resources and Evaluation* 52.1 (2018), pp. 101–148. ISSN: 1574-0218.

[41] Sandeep Subramanian, Raymond Li, Jonathan Pilault, and Christopher Pal. *On Extractive and Abstractive Neural Document Summarization with Transformer Language Models*. Available online (arXiv). 2019. arXiv: 1909.03186 [cs.CL] (cit. on p. 20).

[42] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. "XLNet: Generalized Autoregressive Pretraining for Language Understanding." In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., 2019, pp. 5754–5764 (cit. on p. 18).

[43] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. "Distributed Representations of Words and Phrases and their Compositionality." In: *Advances in Neural Information Processing Systems 26*. Ed. by C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger. Curran Associates, Inc., 2013, pp. 3111–3119 (cit. on p. 17).

[44] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "GloVe: Global Vectors for Word Representation." In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543 (cit. on p. 17).

[45] Yang Liu and Mirella Lapata. "Text Summarization with Pretrained Encoders." In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (2019) (cit. on pp. 21, 35, 53).

[46] Yang Liu. *Fine-tune BERT for Extractive Summarization*. Available online (arXiv). 2019. arXiv: 1903.10318 [cs.CL] (cit. on p. 21).

[47] Elizabeth Clark, Asli Celikyilmaz, and Noah A. Smith. "Sentence Mover's Similarity: Automatic Evaluation for Multi-Sentence Texts." In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 2748–2760 (cit. on p. 22).

[48] Maxime Peyrard. "A Simple Theoretical Model of Importance for Summarization." In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 1059–1073 (cit. on p. 22).

[49] Maxime Peyrard. "Studying Summarization Evaluation Metrics in the Appropriate Scoring Range." In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 5093–5100 (cit. on p. 22).

[50] Alexander M. Rush, Sumit Chopra, and Jason Weston. "A Neural Attention Model for Abstractive Sentence Summarization." In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sept. 2015, pp. 379–389 (cit. on pp. 15, 19).

[51] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. "Transformer-XL: Attentive Language Models beyond a Fixed-Length Context." In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (2019) (cit. on p. 16).

[52] Alex Graves, Greg Wayne, and Ivo Danihelka. *Neural Turing Machines*. Available online (arXiv). 2014. arXiv: 1410.5401 [cs.NE] (cit. on p. 15).

[53] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. Available online (arXiv). 2014. arXiv: 1409.0473 [cs.CL] (cit. on p. 15).

[54] Thang Luong, Hieu Pham, and Christopher D. Manning. "Effective Approaches to Attention-based Neural Machine Translation." In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sept. 2015, pp. 1412–1421 (cit. on p. 15).

[55] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-term Memory." In: *Neural computation* 9 (Dec. 1997), pp. 1735–80 (cit. on p. 13).

[56] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation." In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (2014) (cit. on p. 13).

[57] Julian Moreno Schneider and Georg Rehm. "Curation Technologies for the Construction and Utilisation of Legal Knowledge Graphs." In: *Proceedings of the LREC 2018 Workshop on Language Resources and Technologies for the Legal Knowledge Graph. International Conference on Language Resources and Evaluation (LREC-2018), Miyazaki, Japan*. Ed. by Georg Rehm, Víctor Rodríguez-Doncel, and Julian Moreno Schneider. Springer, May 2018 (cit. on p. 2).

[58] Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. *Teaching Machines to Read and Comprehend*. Available online (arXiv). 2015. arXiv: 1506.03340 [cs.CL] (cit. on p. 19).

[59] Georg Rehm, Peter Bourgonje, Stefanie Hegele, Florian Kintzel, Julián Moreno Schneider, Malte Ostendorff, Karolina Zaczynska, Armin Berger, Stefan Grill, Sören Räuchle, Jens Rauenbusch, Lisa Rutenburg, André Schmidt, Mikka Wild, Henry Hoffmann, Julian Fink, Sarah Schulz, Jurica Seva, Joachim Quantz, Joachim Böttger, Josefine Matthey, Rolf Fricke, Jan Thomsen, Adrian Paschke, Jamal Al Qundus, Thomas Hoppe, Naouel Karam, Frauke Weichhardt, Christian Fillies, Clemens Neudecker, Mike Gerber, Kai Labusch, Vahid Rezanezhad, Robin Schaefer, David Zellhöfer, Daniel Siewert, Patrick Bunk, Lydia Pintscher, Elena Aleynikova, and Franziska Heine. "QURATOR: Innovative Technologies for Content and Data Curation." In: *Proceedings of QURATOR 2020 – The conference for intelligent content solutions*. Ed. by Adrian Paschke, Clemens Neudecker, Lydia Pintscher, Jamal Al Qundus, and Georg Rehm. Accepted for presentation. 20/21 January 2020. Berin, Germany, Feb. 2020 (cit. on pp. 2, 57).

[60] Dmitrii Aksenov, Julin Moreno-Schneider, Peter Bourgonje, Robert Schwarzenberg, Leonhard Hennig, and Georg Rehm. "Abstractive Text Summarization based on Language Model Conditioning and Locality Modeling." In: *Proceedings of the 12th Language Resources and Evaluation Conference (LREC 2020)*. Ed. by Nicoletta Calzolari, Fr©d©ric B©chet, Philippe Blache, Christopher Cieri, Khalid Choukri, Thierry Declerck, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis. Accepted for publication. Final camera-ready version available as preprint. Marseille, France: European Language Resources Association (ELRA), May 2020 (cit. on pp. 4, 57).

[61] Georg Rehm, Maria Berger, Ela Elsholz, Stefanie Hegele, Florian Kintzel, Katrin Marheinecke, Stelios Piperidis, Miltos Deligiannis, Dimitris Galanis, Katerina Gkirtzou, Penny Labropoulou, Kalina Bontcheva, David Jones, Ian Roberts, Jan Hajic, Jana Hamrlová, Lukáš Kačena, Khalid Choukri, Victoria Arranz, Andrejs Vasiljevs, Orians Anvari, Andis Lagzdiņš, Jūlija Meļņika, Gerhard Backfried, Erinç Dikici, Miroslav Janosik, Katja Prinz, Christoph Prinz, Severin Stampler, Dorothea Thomas-Aniola, José Manuel Gómez Pérez, Andres Garcia Silva, Christian Berrío, Ulrich Germann, Steve Renals, and Ondrej Klejch. "European Language Grid: An Overview." In: *Proceedings of the 12th Language Resources and Evaluation Conference (LREC 2020)*. Ed. by Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Christopher Cieri, Khalid Choukri, Thierry Declerck, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis. Accepted for publication. Marseille, France: European Language Resources Association (ELRA), May 2020 (cit. on p. 57).

[62] Julián Moreno-Schneider, Georg Rehm, Elena Montiel-Ponsoda, Víctor Rodriguez-Doncel, Artem Revenko, Sotirios Karampatakis, Maria Khvalchik, Christian Sageder, Jorge Gracia, and Filippo Maganza. "Orchestrating NLP Services for the Legal Domain." In: *Proceedings of the 12th Language Resources and Evaluation Conference (LREC 2020)*. Ed. by Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Christopher Cieri, Khalid Choukri, Thierry Declerck, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis. Accepted for publication. Submitted version available as preprint. Marseille, France: European Language Resources Association (ELRA), May 2020 (cit. on p. 57).

Part I

APPENDIX

Abstractive Text Summarization based on Language Model Conditioning and Locality Modeling

Dmitrii Aksenov, Julián Moreno-Schneider, Peter Bourgonje, Robert Schwarzenberg, Leonhard Hennig, Georg Rehm

DFKI GmbH, Alt-Moabit 91c, 10559 Berlin, Germany
{firstname.lastname}@dfki.de

Abstract

We explore to what extent knowledge about the pre-trained language model that is used is beneficial for the task of abstractive summarization. To this end, we experiment with conditioning the encoder and decoder of a Transformer-based neural model on the BERT language model. In addition, we propose a new method of BERT-windowing, which allows chunk-wise processing of texts longer than the BERT window size. We also explore how locality modeling, i. e., the explicit restriction of calculations to the local context, can affect the summarization ability of the Transformer. This is done by introducing 2-dimensional convolutional self-attention into the first layers of the encoder. The results of our models are compared to a baseline and the state-of-the-art models on the CNN/Daily Mail dataset. We additionally train our model on the SwissText dataset to demonstrate usability on German. Both models outperform the baseline in ROUGE scores on two datasets and show its superiority in a manual qualitative analysis.

Keywords: Summarisation, Language Modeling, Information Extraction, Information Retrieval, BERT, Locality Modeling

1. Introduction

Text summarization is an NLP task with many real-world applications. The ever-increasing amount of unstructured information in text form calls for methods to automatically extract the relevant information from documents and present it in condensed form. Within the field of summarization, different paradigms are recognised in two dimensions: extractive vs. abstractive, and single-document vs. multi-document. In extractive summarization, those sentences or words are extracted from a text which carry the most important information, directly presenting the result of this as the summary. Abstractive summarization methods paraphrase the text, and by changing the text aim to generate more flexible and consistent summaries. Furthermore, single-document summarization works on single documents, while multi-document summarization deals with multiple documents at once and produces a single summary. In this paper, we concentrate on single-document abstractive summarization. Most recent abstractive models utilize the neural network-based sequence-to-sequence approach. During training, such models calculate the conditional probability of a summary given the input sequence by maximizing the loss function (typically cross-entropy). Most approaches are based on the encoder-decoder framework where the encoder encodes the input sequence into a vector representation and the decoder produces a new summary given the draft summary (which is the part of the summary generated during previous iterations). The last layer of a decoder, the generator, maps hidden states to token probabilities. We use a state-of-the-art Transformer for sequence-to-sequence tasks which is built primarily on the attention mechanism (Vaswani et al., 2017).

We attempt to improve performance of abstractive text summarization by improving the language encoding capabilities of the model. Recent results have shown that the main contribution of the Transformer is its multi-layer architecture, allowing Self-Attention to be replaced with some other technique without a significant drop in performance (Domhan, 2018; Wu et al., 2019). Following this strategy, we develop a model that introduces convolution into the vanilla Self-Attention, allowing to better encode the local dependencies between tokens. To overcome the data sparsity problem, we use a pre-trained language model for the encoding part of the encoder-decoder setup, which creates a contextualized representation of the input sequence. Specifically, we use BERT due to its bi-directional context conditioning, multilingualism and state-of-the-art scores on many other tasks (Devlin et al., 2019). Furthermore, we propose a new method which allows applying BERT on longer texts. The main contributions of this paper are: (1) Designing two new abstractive text summarization models based on the ideas of conditioning on the pre-trained language model and application of convolutional self-attention at the bottom layers of the encoder. (2) Proposing a method of encoding the input sequence in windows which alleviates BERT's input limitations[1] and allows the processing of longer input texts. (3) Evaluating the performance of our models on the English and German language by conducting an ablation study on CNN/Dail Mail and SwissText datasets and comparing it with other state-of-the-art methods.

2. Related Work

2.1. Pre-trained Language Models

Traditionally, non-contextualized embedding vectors were used for pre-training neural-based NLP models (Mikolov et al., 2013; Pennington et al., 2014). Recently, pre-trained language models exploiting contextualized embeddings, such as ELMo, GPT-2, BERT and XLNet raised the bar in many NLP tasks (Peters et al., 2018; Radford et al., 2018; Devlin et al., 2019; Yang et al., 2019b). Recent attempts to use these models for text summarization demon-

[1] BERT can process sequences with a maximum of 512 tokens.

strated their suitability by achieving new state-of-the-art results (Zhang et al., 2019; Liu, 2019; Liu and Lapata, 2019).

2.2. Neural Abstractive Text Summarization

The neural approach toward abstractive summarization was largely adopted by state-of-the-art models (Shi et al., 2018). A significant contribution was the pointer Generator Network (See et al., 2017). It uses a special layer on top of the decoder network to be able to both generate tokens from the dictionary and extract them from the input text. It uses the coverage vector mechanism to pay less attention to tokens already covered by previous iterations. An example of earlier work adapting Reinforcement Learning (RL) is described by Paulus et al. (2018). The pure RL model achieved high ROUGE-1 and ROUGE-L scores but produced unreadable summaries. Its combination with typical cross-entropy optimization achieved high scores eliminating the unreliability problem. Liu et al. (2018), to the best of our knowledge, were the first to use the Transformer model for summarization. It was only used in the decoder on top of the extraction model with various attention compression techniques to increase the size of the input sequence. Zhang et al. (2019) incorporate BERT into the Transformer-based model. They use a two-stage procedure exploiting the mask learning strategy. Others attempt to improve their abstractive summarization models by incorporating an extractive model. For example, Li et al. (2018) use the Key information guide network to guide the summary generation process. In Bottom-up summarization (Gehrmann et al., 2018) the extractive model is used to increase the precision of the Pointer Generator mechanism. Another strand of research adapts existing models to cope with long text. Cohan et al. (2018) present the Discourse-Aware Attention model which introduces hierarchy in the attention mechanism via calculating an additional attention vector over the sections of the input text. Subramanian et al. (2019) showed that the language model trained on the combination of the original text, extractive summaries generated by the model and the golden summary can achieve results comparable to standard encoder-decoder based summarization models.

3. Approach

Our text summarization model is based on the Transformer architecture. This architecture adopts the original model of Vaswani et al. (2017). On top of the decoder, we use a Pointer-Generator (Formula 1) to increase the extractive capabilities of the network (we later refer to this architecture as CopyTransformer).

$$p(w) = p_{gen}P_{copy}(w) + (1 - p_{gen})P_{softmax}(w), \quad (1)$$

where $P_{copy}(w)$ is the probability of copying a specific word w from the source document, $P_{softmax}(w)$ is the probability of generation a word calculated by the abstractive summarization model and p_{gen} is the probability of copying instead of generation.

3.1. Convolutional Self-Attention

The Transformer, like any other self-attention network, has a hierarchical multi-layer architecture. In many experiments it was shown that this architecture tends to learn lexical information in the first layers, sentence-level patterns in the middle and the semantics in the upper layers (Raganato and Tiedemann, 2018; Tenney et al., 2019). The disadvantage of this approach is that during the attention operation it considers all tokens as equally important, whereas syntactic information is mostly concentrated in certain local areas. This problem is usually specified as the problem of locality modeling. As syntactic information can help in identifying more important words or phrases, it could be beneficial to focus attention on these regions.

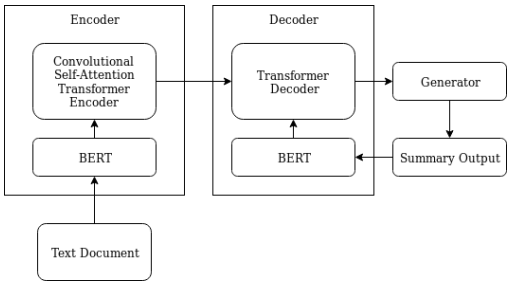


Figure 1: Model overview

A successful approach to the locality modeling task are the so-called convolutions (local) self-attention networks (Yang et al., 2019a). Essentially, the problem is dealt with by the application of a 1-dimensional convolution to the self-attention operation at the network's lower layers. This strengthens dependencies among neighboring elements and makes the model distance-aware when it searches for low-level patterns in a sequence. In other words, it restricts the attention scope to the window of neighboring elements. The 1D convolution applied to attention is illustrated in Formulas 2, 3 and 4.

$$\widehat{\mathbf{K}}^h = \{\mathbf{k}^h_{i-\frac{M}{2}}, \ldots, \mathbf{k}^h_i, \ldots, \mathbf{k}^h_{i+\frac{M}{2}}\}, \quad (2)$$

$$\widehat{\mathbf{V}}^h = \{\mathbf{v}^h_{i-\frac{M}{2}}, \ldots, \mathbf{v}^h_i, \ldots, \mathbf{v}^h_{i+\frac{M}{2}}\}, \quad (3)$$

$$\mathbf{o}^h_i = \textsc{Att}(\mathbf{q}^h_i, \widehat{\mathbf{K}}^h)\widehat{\mathbf{V}}^h, \quad (4)$$

where $\mathbf{q}^h_i$ is the query and $M + 1$ ($M \leq I$) is its attention region centered at the position i.

The convolution can be extended to the 2-dimensional area by taking interactions between features learned by the different attention heads of the Transformer into account. In the original Transformer each head independently models a distinct set of linguistic properties and dependencies among tokens (Raganato and Tiedemann, 2018). By applying 2-dimensional convolution, where the second dimension is the index of attention head, we explicitly allow each head to interact with learned features for their adjacent sub-spaces. The shortcoming of the original implementation is that the first and the last heads do not interact as they are assumed not to be adjacent. Thus, we assume that considering the heads' sub-spaces periodically, we can increase the model's effectiveness by applying circular convolution to the second dimension. In Section 5, we evaluate both the original version and our modification.

$$\widetilde{\mathbf{K}}^h = \bigcup [\widehat{\mathbf{K}}^{h-\frac{N}{2}}, \ldots, \widehat{\mathbf{K}}^h, \ldots, \widehat{\mathbf{K}}^{h+\frac{N}{2}}], \quad (5)$$

$$\widetilde{\mathbf{V}}^h = \bigcup [\widehat{\mathbf{V}}^{h-\frac{N}{2}}, \ldots, \widehat{\mathbf{V}}^h, \ldots, \widehat{\mathbf{V}}^{h+\frac{N}{2}}], \quad (6)$$

$$\mathbf{o}_i^h = \textsc{Att}(\mathbf{q}_i^h, \widetilde{\mathbf{K}}^h)\widetilde{\mathbf{V}}^h, \quad (7)$$

where $(M+1)$ $(N \leq H)$ is the window region over heads and $\bigcup$ stands for the union of keys $\widehat{\mathbf{K}}^h$ and values $\widehat{\mathbf{V}}^h$ from different subspaces.

The convolutional self-attention has been shown to be very effective in Machine Translation and several other NLP tasks. However, to our knowledge, it was never applied to the text summarization problem. For the experiments reported on in this paper, we created our implementation of the local attention and the convolutional self-attention network (Transformer). It supports both 1D and 2D modes having the size of the kernels as system parameters. As in Yang et al. (2019a) we incorporate convolutional self-attention in the Transformer encoder by positioning it in the place of the self-attention in the lower layers. In Section 5, we show that the low-level modeling capabilities of our encoder provides a strong boost to the model's prediction accuracy in the text summarization task.

3.2. BERT-Conditioned Encoder

The main task of the encoder is to remember all the semantic and syntactic information from the input text which should be used by the decoder to generate the output. Knowledge transfer from the language model should theoretically improve its ability to remember the important information due to the much larger corpus used in its pre-training phase compared to the corpus used in the text summarization training phase. We thus condition our encoder on the BERT language model.

For the encoder conditioning, we used the most straightforward strategy recommended for the BERT based model: placing the pre-trained language model in the encoder as an embeddings layer. This should make the embeddings of the system context-dependent. We decided not to fine-tune the encoder on BERT for the sake of memory and time economy. Instead, we follow the general recommendations by concatenating the hidden states of the last four layers of BERT into a 3072-dimensional embedding vector (Devlin et al., 2019). We use two variations of the BERT-based encoder. The first model uses only BERT to encode the input sequence and the second model feeds BERT's generated embeddings into the vanilla Transformer encoder.

3.3. BERT-Windowing

One of the key features of our approach is its ability to overcome the length limitations of BERT, allowing it to deal with longer documents. BERT's maximum supported sequence length is 512 tokens[2], which is smaller than the average size of texts used in most summarization datasets. Our method relies on the well-known method of windowing which to our knowledge was never used before neither in the BERT-based models nor in abstractive text summarization research (Figure 2). We apply BERT to the windows of texts with strides and generate N matrices, every matrix embedding one window. Then we combine them by doing the reverse operation. The vectors at the overlapping positions are averaged (by summing and dividing by the number of overlapping vectors). As a result, we have the matrix of embeddings with the shape of the hidden size times the length of the text. The drawback of this approach is that we reduce the size of the context as each resulted vector is calculated based on maximum twice the window size number of tokens. Besides, the split of the text to equal size windows will aggravate the consistency of the input as some sentences will be split in an arbitrary manner between two adjacent windows. Despite this drawback, we assume that this procedure will nevertheless improve the accuracy of the encoder trained on the non-truncated texts. We set the window size to the maximum size of 512 tokens and the stride to 256. We consider this stride size optimal due to a trade-off between the average context size and computational requirements of the model (number of windows). By this trade we ensure every token to have a 768 tokens-context except for the 256 initial and final tokens, that only have 512 tokens-context.

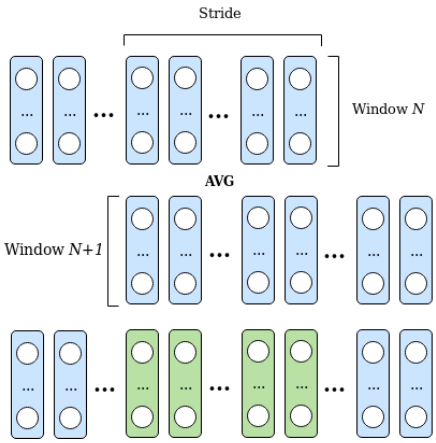


Figure 2: Integration of BERT-generated contextual representations from two windows

3.4. BERT-Conditioned Decoder

In the decoder, pre-training was applied in a similar way. The main difference is that instead of the final output of BERT we use only its word embedding matrix (without positions). The reason behind this is that in the decoder the generated probability distribution is conditioned on the incomplete text (previous summary draft output) while BERT implicitly assumes consistent and completed input (Zhang et al., 2019). As context-independent embeddings are not enough to represent the minimum set of features to make a meaningful prediction, the custom Transformer decoder is always stacked on top of BERT.

Our whole BERT-based model is similar to One-Stage BERT (Zhang et al., 2019) and BertSumAbs (Liu and Lapata, 2019) but differs in the usage of the four last hidden states of BERT to create contextualized representation, in presence of Pointer Generator and capabilities to process

[2]These are not tokens in the traditional sense, but so-called WordPiece tokens, see Devlin et al. (2019).

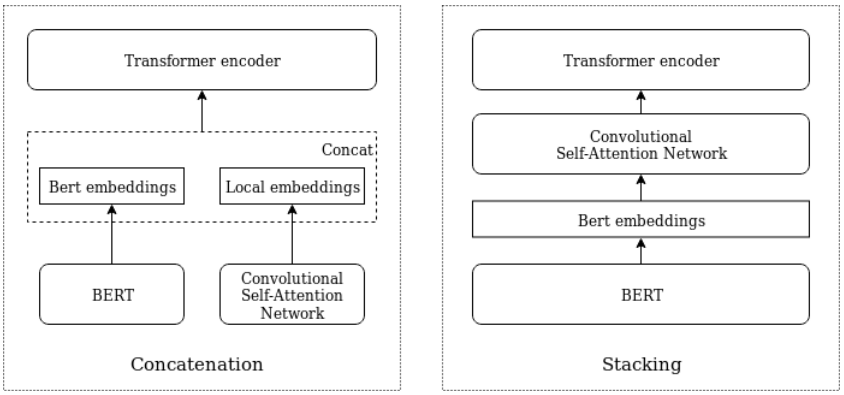


Figure 3: Two different approaches for the integration of the BERT-conditioning with Convolutional Self-Attention

Method	ROUGE-1	ROUGE-2	ROUGE-L
CopyTransformer	31.95	14.49	30.02
+ 1D conv.	32.62	14.99	30.74
+ 2D conv.	**32.72**	**15.12**	**30.85**
+ 2D Circular conv.	32.68	15.01	30.76

Table 1: Ablation study of model with Convolutional Self-Attention on the CNN/Daily Mail dataset (kernel sizes are 11 and 3)

long texts. In Figure 1 we show the schema of the basic model with the BERT-conditioned convolutional self-attention encoder and BERT-conditioned decoder.

3.5. Integration of BERT and Convolutional Self-Attention

We evaluated two different ways of integrating the BERT-conditioning with the convolutional self-attention of the model's encoder (Figure 3).

Stacking This approach comprises feeding the BERT-generated embeddings to the convolutional self-attention Transformer encoder. A potential problem with this approach is that convolutional self-attention is assumed to be beneficial when applied in the lower layers as its locality modeling feature should help in modeling of local dependencies (e. g., syntax). At the same time, BERT is a hierarchical model where the last layers target high-level patterns in the sequences (e. g., semantics). We assume that the application of the network detecting the low-level patterns on BERT's output can undermine its generalization abilities.

Concatenation Because of the considerations raised above, we also develop a second approach which we call Concatenation. We split the convolutional self-attention Transformer encoder into two networks where the first one uses only convolutional self-attention and the second original self-attention (identical to the Transformer encoder). Then we feed the original sequences into BERT and into the convolutional self-attention network in parallel. The resulting embedding vectors are concatenated and fed into the Transformer encoder. In this way, we model the locality at the lower layers of the encoder at the cost of a smaller depth of the network (assuming the same number of layers).

4. Datasets

We aim to develop a system that works in a language-independent way. It assumes that either the upstream components are available in the respective language, or they are themselves language-independent, such as the multi-lingual version of BERT. Since most summarization datasets are in English however, we use English for the evaluation and additionally include German to check if of our model can be applied to another language.

4.1. CNN/Daily Mail

Our experiments are mainly conducted on the CNN/Daily Mail dataset (Hermann et al., 2015; Nallapati et al., 2016). It contains a collection of news articles paired with multi-sentence summaries published on the CNN and Daily Mail websites. This dataset is the de facto standard for training summarization models. We use the non-anonymized data as was used for training of the most recent state-of-the-art models (e. g., See et al. (2017)). The raw dataset consists of separate text files each representing a single article or a summary. We use the data in its preprocessed version as provided by Gehrmann et al. (2018). It has 287,226 training pairs, 13,368 validation pairs and 11,490 test pairs. To align the data with the vocabulary of BERT we tokenized it using the BPE-based WordPiece tokenizer (Devlin et al., 2019). As all samples in BERT's training data are prepended with the special token "[CLS]", we follow

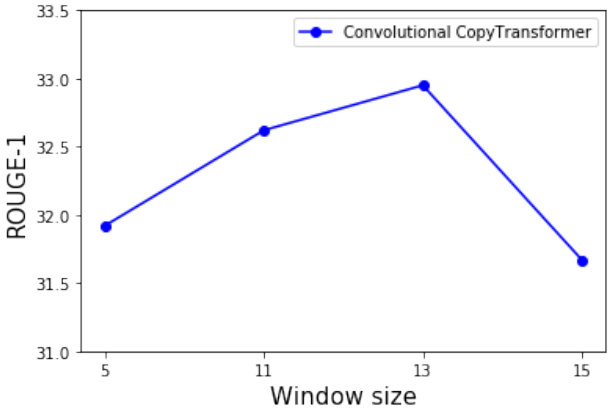


Figure 4: Effect of the window size on ROUGE-1

Model	ROUGE-1	ROUGE-2	ROUGE-L
Transformer	24.82	6.27	22.99
CopyTransformer	31.95	14.49	30.02
Bert Encoder + Transformer Decoder	31.3	13.37	29.46
Bert-transformer Encoder + Transformer Decoder	32.5	14.68	30.68
Bert-transformer Encoder + Bert-transformer Decoder	**33.23**	**14.99**	**31.26**
Transformer (full text)	23.18	5.15	21.48
Bert-transformer Encoder + Transformer Decoder (full text)	**31.51**	**14.1**	**29.77**

Table 2: Ablation study of the BERT-based model on truncated and original CNN/Daily Mail dataset

Model	ROUGE-1	ROUGE-2	ROUGE-L
Transformer	36.40	20.69	34.14
CopyTransformer	39.44	25.11	37.16
Bert-transformer Encoder + Transformer Decoder	**44.01**	**29.60**	**41.65**
Bert-transformer Encoder + Bert-transformer Decoder	43.22	29.01	40.84
Transformer (full text)	34.76	18.65	32.61
Bert-transformer Encoder + Transformer Decoder (full text)	**45**	**30.49**	**42.64**

Table 3: Ablation study of the BERT-based model on the truncated and original SwissText dataset

this and add it to every source text in our dataset. In the resulting dataset, the average lengths of an article and a summary are 895 and 63 tokens, respectively. In most of our experiments, we use the clipped version of the training and validation datasets with each article truncated to 512 tokens. In the experiments on BERT windowing, we use the full-text version.

4.2. SwissText Dataset

To evaluate the efficiency of the model in a multi-lingual, multi-domain environment we conduct a series of experiments on the German SwissText dataset. This dataset was created for the 1st German Text Summarization Challenge at the 4th Swiss Text Analytics Conference – SwissText 2019 (ZHAW, 2019). It was designed to explore different ideas and solutions regarding abstractive summarization of German texts. To the best of our knowledge, it is the first long document summarization dataset in the German language that is publicly available. The data was extracted from the German Wikipedia and represents mostly biographical articles and definitions of various concepts.
The dataset was tokenized by the multilingual WordPiece tokenizer (Devlin et al., 2019) and preprocessed in the same way as the CNN/Daily Mail dataset. It was split into the training, validation and testing sets containing 90,000, 5,000 and 5,000 samples, respectively. The average length of a source sequence is 918 tokens, which makes this dataset suitable for our experiments on windowing.

5. Experiments

Our system is built on the OpenNMT library. For training, we use cross-entropy loss and the Adam optimizer with the Noam decay method (Kingma and Ba, 2014). Regularization is made via dropout and label smoothing. For evaluation, we calculate the F1-scores for ROUGE using the files2rouge library. The ROUGE evaluation is made on the sequences of WordPiece tokens.

5.1. Locality Modeling

To evaluate the effect of convolution on self-attention we introduce it in the first layer of the encoder. We use the same kernel sizes as in Yang et al. (2019a). In these experiments, to accelerate the training process, we use a small model with a hidden size of 256, four self-attention heads and three layers in the encoder and decoder. All models are trained for 90,000 training steps with the Coverage Penalty. As a baseline, we use our implementation of CopyTransformer. In contrast to See et al. (2017), we do not re-use the attention layer for the decoder but train a new Pointer-Generator layer from scratch.
The results are presented in Table 1. We see that both convolutions over tokens and over attention heads improve the ROUGE scores. Standard convolution outperformed circular convolution on ROUGE-1, ROUGE-2 and ROUGE-L by 0.06, 0.13 and 0.09 percent, respectively.
We also investigated the effect of the window size of the 1-dimensional convolution on ROUGE scores (Figure 4). In contrast to findings in Machine Translation, we found that size 13 returns the best result for the summarization task.

5.2. BERT Conditioning

To find the optimal architecture of the BERT-based abstractive summarizer we conducted an ablation study (Table 2). All hyperparameters were set equal to the ones in experiments in convolutional self-attention. On CNN/Daily Main dataset we test three different models: BERT encoder+Transformer Decoder, BERT-Transformer encoder+Transformer decoder and BERT-Transformer encoder+BERT-Transformer decoder. The version of BERT used in the experiments is BERT-Base. As the baseline, we use the Transformer without Pointer Generator. From the results, we observe that BERT improves the efficiency of the model when it is used in both encoder and decoder. Besides, BERT in the encoder is more effective when it is used to produce embeddings to be used by the

standard Transformer encoder than when it is used solely as an encoder. Even without a Pointer Generator, our model outperformed the CopyTransformer baseline by 1.28, 0.5 and 1.24 on ROUGE-1, ROUGE-2 and ROUGE-L.

To evaluate our BERT-windowing method we conducted the experiments on the full text. Our approach outperforms the baseline, which proves that the method can be successfully applied to texts longer than 512 tokens. The final performance of this model is still lower than that of the model trained on the truncated text, but as the same pattern can be observed for the baselines we assumed this relates to the specifics of the dataset that is prone to having important information in the first sentence of a text.

On SwissText data we use the multilingual version of BERT-Base. We evaluated two models with Bert-transformer encoder and Transformer and BERT-Transformer decoders (Table 3). The introduction of BERT into the transformer increased the ROUGE-1, ROUGE-2 and ROUGE-L scores by 7.21, 8.91 and 7.51 percent, respectively. At the same time, the usage of BERT in the decoder decreased the overall score. We assume that the reason behind this is that in multilingual BERT, due to its language-independence, the embedding matrix outputs less precise contextualized representations which undermines their benefits for the summarization task.

On the non-truncated texts, usage of the Bert-transformer encoder increased the ROUGE scores by 10.23, 11.84 and 10.03 percent. Furthermore, it gives us higher scores compared to the same model on truncated texts. This demonstrates the usability of BERT-windowing for this particular dataset. We assume that the difference in performance on the CNN/Daily Mail datasets reflects the difference in distribution of the useful information within the text. Particularly, that in the SwissText dataset, it is spread more uniformly than in the CNN/Daily Mail dataset. We conducted a small experiment comparing the average ROUGE score between a golden summary and the head and the tail of a document (taking the first or last n sentences, where n correlates to the length of the gold summary) on both datasets. The difference between taking the head and a tail on the SwissText dataset (ROUGE-L of 34.79 vs. 20.15, respectively) was much smaller than on CNN / Daily Mail (ROUGE-L of 16.95 vs. 12.27, respectively) which confirms our hypothesis.

5.3. Integration Strategies

To evaluate the integration strategies, we trained two models with the respective BERT-based baselines. Both models have in their encoder two Transformer layers and one Convolutional Transformer layer placed on top of BERT or in parallel, respectively (Table 4).

The method of stacking does not provide any significant improvement. With the introduction of convolutional self-attention only ROUGE-1 increased by 0.12 percent, while ROUGE-2 dropped by 0.3 and ROUGE-L remained the same. Considering that in many domains ROUGE-2 maximally correlates with human assessment (see Section 7), we dismiss this method. The concatenation strategy convolution is shown to be much more efficient, increasing ROUGE scores by 0.44,0.33 and 0.43 percent. This confirms our hypothesis that locality modeling is the most efficient when applied at the bottom on the non-contextualized word representations. Unfortunately, this model failed to outperform the stacking baseline. We conclude that the concatenating architecture undermines the performance of the Transformer model, and the convolutional self-attention is not beneficial when used together with pre-trained language models. Hence, we decided to train our two final models separately.

5.4. Model Comparison

For the final comparison of our model to other state-of-the-art methods we conducted experiments on the CNN/Daily Mail dataset. We set the hidden state to 512, the number of Transformer layers in the encoder and layers to six and the number of self-attention heads to eight. Hence, our baseline is smaller than the original CopyTransformer (Gehrmann et al., 2018), which may be the reason why it performs slightly worse (Table 5). BERT-conditioning was used in both the encoder and decoder. The sizes of convolution kernels are set to 13 and three. The networks were trained for 200,000 training steps on a single NVIDIA GeForce GTX 1080 Ti. The generation of the summary was made via the Beam search algorithm with the Beam size set to four. Finally, the generated summaries were detokenized back to the sequences of words separated by spaces.

For the BERT-based model, we set the minimum length of a generated summary to 55 as we found that without such restriction the model was prone to generate shorter sequences than in the test dataset. The model outperformed the baseline by 1.27 on ROUGE-1, 1.14 on ROUGE-2 and 1.3 on ROUGE-L. This is better than the scores of One-Stage BERT but still worse than the two-stage and Bert-SumAbs models.

For the convolutional CopyTransformer we use convolutional self-attention in the first three layers of the encoder. It increased ROUGE-1, ROUGE-2 and ROUGE-L by 0.25, 0.41 and 0.12.

Furthermore, we present the first publicly available benchmark for the SwissData dataset (Table 6).[3] All param-

[3]For comparability with our other model we include results

Method of Integration	Model	ROUGE-1	ROUGE-2	ROUGE-L
Stacking	BERT+CopyTransformer	35.28	**17.12**	33.31
	BERT+Convolutional CopyTransformer	**35.4**	16.82	**33.31**
Concatenation	BERT+CopyTransformer	34.82	16.46	32.79
	BERT+Convolutional CopyTransformer	**35.26**	**16.79**	**33.22**

Table 4: Different strategies for integrating language models with convolutional Self-Attention (CNN/Daily Mail dataset)

Method	ROUGE-1	ROUGE-2	ROUGE-L
BiLSTM + Pointer-Generator + Coverage (See et al., 2017)	39.53	17.28	36.38
ML + Intra-Attention (Paulus et al., 2018)	38.30	14.81	35.49
CopyTransformer (Gehrmann et al., 2018)	39.25	17.54	36.45
Bottom-Up Summarization (Gehrmann et al., 2018)	41.22	18.68	38.34
One-Stage BERT (Zhang et al., 2019)	39.50	17.87	36.65
Two-Stage BERT (Zhang et al., 2019)	41.38	19.34	38.37
ML + Intra-Attention + RL (Paulus et al., 2018)	39.87	15.82	36.90
Key information guide network (Li et al., 2018)	38.95	17.12	35.68
Sentence Rewriting (Chen and Bansal, 2018)	40.88	17.80	38.54
BertSumAbs (Liu and Lapata, 2019)	**41.72**	**19.39**	**38.76**
CopyTransformer (our implementation)	38.73	17.28	35.85
Convolutional CopyTransformer	38.98	17.69	35.97
BERT+CopyTransformer (enc., dec.)	**40**	**18.42**	**37.15**

Table 5: ROUGE scores for various models on the CNN/Daily Mail test set. The first section shows different state-of-the-art models, the second section presents our models and baseline.

Method	ROUGE-1	ROUGE-2	ROUGE-L
CopyTransformer (our implementation)	39.5	22.36	36.97
Convolutional CopyTransformer	40.54	23.62	38.06
BERT+CopyTransformer (enc.)	**42.61**	**25.25**	**39.85**

Table 6: ROUGE scores for our models on the SwissText test set

eters are equal to the CNN/Daily Mail baseline. BERT-conditioning was used only in the encoder. The networks were trained on the truncated texts in 90,000 training steps. From the results we see that the convolutional CopyTransformer showed much more efficiency than on CNN/Daily Mail dataset, outperforming the baseline by 1.04 percent on ROUGE-1, 1.26 on ROUGE-2 and 1.09 on ROUGE-L. The BERT-based model achieved the highest scores.

6. Qualitative Analysis

As ROUGE evaluation is not always a valid method for quality assessment we perceive the need for an additional, manual evaluation. The best solution would be to conduct a fine-grained study of the models' outputs by manually ranking them in terms of semantic coherence, grammaticality, etc. However, due to the time-consuming nature of such an evaluation, we reverted to a qualitative analysis comparing several summaries generated by different models. Figure 5 includes the reference summary and those generated by the different models. Comparing the first sentence we see that the vanilla Transformer model performed worse by copying only part of the original sentence omitting some characters in the word "meteorological". The model with convolution has copied the whole sentence but still made a spelling error. Finally, only the BERT-based model succeeded to generate the right token, "meteorological". Also, we see that while the BERT-based model's summary conveys the same meaning as the gold summary, the convolutional Transformer generates one and Transformer two sentences that are not present in the gold summary. Overall, on the given example all models provided a summary of extractive nature and only the BERT-based model shows some level of abstractiveness merging parts of the two sentences into the single one (in the second summary's sentence). This is far from the gold summary where every sentence in some way paraphrases the original text. Hence, given this particular example, our models demonstrate some explicit improvements. Still, abstractive summarization remains challenging. The paraphrasing capabilities of all state-of-the-art systems are low and the models are not guaranteed to produce summaries which follow the initial order of the sequence of events.

7. Discussion: Summarization Evaluation

ROUGE (Lin, 2004) is the most widely adopted metric used for evaluating automatic text summarization approaches. The evaluation is made though comparison of a set of system-generated candidate summaries with a gold standard summary. The availability of the corresponding software and its performance contributed to its popularity (Cohan and Goharian, 2016). Despite its adoption in many studies, the metric faced some key criticisms.

The main criticism of ROUGE is that it does not take into account the meaning expressed in the sequences. The metric was developed based on the assumption that a high quality generated candidate summary should share many words with a single human-made gold standard summary. This assumption may be very relevant to extractive, but not to abstractive summarization, where different terminology and paraphrasing can be used to express the same meaning (Cohan and Goharian, 2016). This results in the metric assigning low scores to any summary not matching the gold standard on the surface level. This also allows cheating the metric by generating ungrammatical and nonsensical sum-

for the bigger BERT+CopyTransformer model. At the same time, we found that the smaller model without the copy mechanism achieved higher scores with 45.12 ROUGE-1, 28.38 ROUGE-2 and 42.99 ROUGE-L. This needs to be explored in future work.

Gold summary: researchers are developing a computer that can write weather forecasts . it takes meteorological data and writes a report designed to mimic a human . this process is known as ‘ natural language generation ’ - lrb - nlg - rrb - . a prototype system will be tested on the bbc website later this year .

Transformer: researchers from london and edinburgh have developed a computer that can collateological information . these computer - generated weather updates are being tested by scientists at heriot - watt university and university college london . if the project is successful , a prototype system will be tested by generating local weather reports on the bbc ’ s website . currently , the bbc website features 10 reports written by meteorologists .

Convolutional Transformer: researchers from london and edinburgh have developed a computer that can collate meterological information and then produce forecasts as if they were written by a human . it uses a process known as ‘ natural language generation ’ - lrb - nlg - rrb - . these computer - generated weather updates are being tested by scientists at heriot - watt university and university college london . if the project is successful , a prototype system will be tested by generating local weather reports on the bbc ’ s website .

BERT-Transformer: researchers from london and edinburgh have developed a computer that can collate meteorological information and produce forecasts as if they were written by a human . using met office data , it uses a process known as ‘ natural language generation ’ - lrb - nlg - rrb - . if the project is successful , a prototype system will be tested by generating local weather reports on the bbc ’ s website .

Figure 5: Comparison of the output of models on an example form CNN/Daily Mail testset. Surface realisation mistakes are highlighted in green and a typical abstractive feature, illustrating re-arranging of the sentence is highlighted in blue.

maries having very high ROUGE scores. Sjöbergh (2007) show how this can be achieved by choosing the most frequent bigrams from the input document.

ROUGE adoption relies on its correlation with human assessment. In the first research on the DUC and TDT-3 datasets containing news articles, ROUGE indeed showed a high correlation with the human judgments (Lin, 2004; Dorr et al., 2005). However, more recent research questions the suitability of ROUGE for various settings. Conroy and Dang (2008) show that on DUC data the linguistic and responsiveness scores of some systems do not correspond to the high ROUGE scores. Cohan and Goharian (2016) demonstrate that for summarization of scientific texts, ROUGE-1 and ROUGE-L have very low correlations with the gold summaries. ROUGE-N correlates better but is still far from the ideal case. This follows the result of Murray et al. (2005), showing that the unigram match between the candidate summary and gold summary is not an accurate metric to assess quality.

Another problem is that the credibility of ROUGE was demonstrated for the systems which operated in the low-scoring range. Peyrard (2019b) show that different summarization evaluation metrics correlate differently with human judgements for the higher-scoring range in which state-of-the-art systems now operate. Furthermore, improvements measured with one metric do not necessarily lead to improvements when using others.

This concern led to the development of new evaluation metrics. Peyrard (2019a) define metrics for important concepts with regard to summariazion: Redundancy, Relevance, and Informativeness in line with Shannon’s entropy. From these definitions they formulate a metric of Importance which better correlates to human judgments. Clark et al. (2019) propose the metric of Sentence Mover’s Similarity which operates on the semantic level and also better correlates with human evaluation. A summarization model trained via Reinforcement Learning with this metric as reward achieved higher scores in both human and ROUGE-based evaluation.

Despite these drawbacks, the broad adoption of ROUGE makes it the only way to compare the efficiency of our model with other state-of-the-art models. The evaluation of our system on the SwissData dataset confirms that its efficiency (in terms of ROUGE) is not restricted to CNN/Daily Mail data only.

8. Conclusion

We present a new abstractive text summarization model which incorporates convolutional self-attention in BERT. We compare the performance of our system to a baseline and to competing systems on the CNN/Daily Mail data set for English and report an improvement over state-of-the-art results using ROUGE scores. To establish suitability of our model to languages other than English and domains other than that of the CNN/Daily Mail data set, we apply our model to the German SwissText data set and present scores on this setup. A key contribution of our model is the ability to deal with texts longer than BERT’s window size which is limited to 512 WordPiece tokens. We present a cascading approach and evaluate this on texts longer than this window size, and demonstrate its performance when dealing with longer input texts.

The source code of our system is publicly available.[4] A functional service based on the model is currently being integrated, as a summarization service, in the platforms Lynx (Moreno-Schneider et al., 2020), QURATOR (Rehm et al., 2020b) and European Language Grid (Rehm et al., 2020a).

Acknowledgements

The work presented in this paper has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement no. 780602 (Lynx) and from the German Federal Ministry of Education and Research (BMBF) through the project QURATOR (Wachstumskern no. 03WKDA1A).

[4] https://github.com/axenov/BERT-Summ-OpenNMT

9. Bibliographical References

Chen, Y.-C. and Bansal, M. (2018). Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 675–686, Melbourne, Australia, July. Association for Computational Linguistics.

Clark, E., Celikyilmaz, A., and Smith, N. A. (2019). Sentence mover's similarity: Automatic evaluation for multi-sentence texts. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2748–2760, Florence, Italy, July. Association for Computational Linguistics.

Cohan, A. and Goharian, N. (2016). Revisiting summarization evaluation for scientific articles. Available online (arXiv).

Cohan, A., Dernoncourt, F., Kim, D. S., Bui, T., Kim, S., Chang, W., and Goharian, N. (2018). A discourse-aware attention model for abstractive summarization of long documents. In *NAACL-HLT*.

Conroy, J. M. and Dang, H. T. (2008). Mind the gap: Dangers of divorcing evaluations of summary content from linguistic quality. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 145–152, Manchester, UK, August. Coling 2008 Organizing Committee.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Domhan, T. (2018). How much attention do you need? a granular analysis of neural machine translation architectures. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1799–1808, Melbourne, Australia, July. Association for Computational Linguistics.

Dorr, B., Monz, C., President, S., Schwartz, R., and Zajic, D. (2005). A methodology for extrinsic evaluation of text summarization: Does ROUGE correlate? In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 1–8, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Gehrmann, S., Deng, Y., and Rush, A. (2018). Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109.

Hermann, K. M., Kočiský, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., and Blunsom, P. (2015). Teaching machines to read and comprehend. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 1693–1701, Cambridge, MA, USA. MIT Press.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12.

Li, C., Xu, W., Li, S., and Gao, S. (2018). Guiding generation for abstractive text summarization based on key information guide network. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 55–60, New Orleans, Louisiana, June. Association for Computational Linguistics.

Lin, C.-Y. (2004). ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July. Association for Computational Linguistics.

Liu, Y. and Lapata, M. (2019). Text summarization with pretrained encoders. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Liu, P. J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L., and Shazeer, N. (2018). Generating wikipedia by summarizing long sequences. In *International Conference on Learning Representations*.

Liu, Y. (2019). Fine-tune bert for extractive summarization. Available online (arXiv).

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, et al., editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Moreno-Schneider, J., Rehm, G., Montiel-Ponsoda, E., Rodriguez-Doncel, V., Revenko, A., Karampatakis, S., Khvalchik, M., Sageder, C., Gracia, J., and Maganza, F. (2020). Orchestrating NLP Services for the Legal Domain. In Nicoletta Calzolari, et al., editors, *Proceedings of the 12th Language Resources and Evaluation Conference (LREC 2020)*, Marseille, France, 5. European Language Resources Association (ELRA). Accepted for publication. Submitted version available as preprint.

Murray, G., Renals, S., and Carletta, J. (2005). Extractive summarization of meeting recordings. In *INTERSPEECH 2005 - Eurospeech, 9th European Conference on Speech Communication and Technology, Lisbon, Portugal, September 4-8, 2005*, pages 593–596.

Nallapati, R., Zhou, B., dos Santos, C., Gülçehre, Ç., and Xiang, B. (2016). Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany, August. Association for Computational Linguistics.

Paulus, R., Xiong, C., and Socher, R. (2018). A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June. Association for Computational Linguistics.

Peyrard, M. (2019a). A simple theoretical model of importance for summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1059–1073, Florence, Italy, July. Association for Computational Linguistics.

Peyrard, M. (2019b). Studying summarization evaluation metrics in the appropriate scoring range. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5093–5100, Florence, Italy, July. Association for Computational Linguistics.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2018). Language models are unsupervised multitask learners. Available online.

Raganato, A. and Tiedemann, J. (2018). An analysis of encoder representations in transformer-based machine translation. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 287–297, Brussels, Belgium, November. Association for Computational Linguistics.

Rehm, G., Berger, M., Elsholz, E., Hegele, S., Kintzel, F., Marheinecke, K., Piperidis, S., Deligiannis, M., Galanis, D., Gkirtzou, K., Labropoulou, P., Bontcheva, K., Jones, D., Roberts, I., Hajic, J., Hamrlová, J., Kačena, L., Choukri, K., Arranz, V., Vasiļjevs, A., Anvari, O., Lagzdiņš, A., Meļņika, J., Backfried, G., Dikici, E., Janosik, M., Prinz, K., Prinz, C., Stampler, S., Thomas-Aniola, D., Pérez, J. M. G., Silva, A. G., Berrío, C., Germann, U., Renals, S., and Klejch, O. (2020a). European Language Grid: An Overview. In Nicoletta Calzolari, et al., editors, *Proceedings of the 12th Language Resources and Evaluation Conference (LREC 2020)*, Marseille, France, 5. European Language Resources Association (ELRA). Accepted for publication.

Rehm, G., Bourgonje, P., Hegele, S., Kintzel, F., Schneider, J. M., Ostendorff, M., Zaczynska, K., Berger, A., Grill, S., Räuchle, S., Rauenbusch, J., Rutenburg, L., Schmidt, A., Wild, M., Hoffmann, H., Fink, J., Schulz, S., Seva, J., Quantz, J., Böttger, J., Matthey, J., Fricke, R., Thomsen, J., Paschke, A., Qundus, J. A., Hoppe, T., Karam, N., Weichhardt, F., Fillies, C., Neudecker, C., Gerber, M., Labusch, K., Rezanezhad, V., Schaefer, R., Zellhöfer, D., Siewert, D., Bunk, P., Pintscher, L., Aleynikova, E., and Heine, F. (2020b). QURATOR: Innovative Technologies for Content and Data Curation. In Adrian Paschke, et al., editors, *Proceedings of QURATOR 2020 – The conference for intelligent content solutions*, Berin, Germany, 02. CEUR Workshop Proceedings, Volume 2535. 20/21 January 2020.

See, A., Liu, P. J., and Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Shi, T., Keneshloo, Y., Ramakrishnan, N., and Reddy, C. K. (2018). Neural abstractive text summarization with sequence-to-sequence models. Available online (arXiv).

Sjöbergh, J. (2007). Older versions of the rougeeval summarization evaluation system were easier to fool. *Information Processing & Management*, 43(6):1500 – 1505. Text Summarization.

Subramanian, S., Li, R., Pilault, J., and Pal, C. (2019). On extractive and abstractive neural document summarization with transformer language models. Available online (arXiv).

Tenney, I., Das, D., and Pavlick, E. (2019). Bert rediscovers the classical nlp pipeline. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 6000–6010, USA. Curran Associates Inc.

Wu, F., Fan, A., Baevski, A., Dauphin, Y., and Auli, M. (2019). Pay less attention with lightweight and dynamic convolutions. In *International Conference on Learning Representations*.

Yang, B., Wang, L., Wong, D. F., Chao, L. S., and Tu, Z. (2019a). Convolutional self-attention networks. *Proceedings of the 2019 Conference of the North*.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. (2019b). Xlnet: Generalized autoregressive pretraining for language understanding. In H. Wallach, et al., editors, *Advances in Neural Information Processing Systems 32*, pages 5754–5764. Curran Associates, Inc.

Zhang, H., Cai, J., Xu, J., and Wang, J. (2019). Pretraining-based natural language generation for text summarization. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 789–797, Hong Kong, China, November. Association for Computational Linguistics.

ZHAW. (2019). German text summarization challenge. Swiss Text Analytics Conference. Available online.