HAW HAMBURG

Masterarbeit

Matthias Nitsche

Towards German Abstractive Text Summarization using Deep Learning

Fakultät Technik und Informatik
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

Matthias Nitsche

Towards German Abstractive Text Summarization using Deep Learning

Mastertarbeit eingereicht im Rahmen der Masterprüfung
im Studiengang Master of Science Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Kai von Luck
Zweitgutachter: Prof. Dr.-Ing. Marina Tropmann-Frick

Eingereicht am: 29. August 2019

Matthias Nitsche

Title of Thesis

Towards German Abstractive Text Summarization using Deep Learning

Keywords

Neural Networks, NLP, Summarization, Transformers, Language Modelling, ELMo, BERT, Transfer Learning

Abstract

Text summarization is an established sequence learning problem divided into extractive and abstractive models. While extractive models learn to only rank words and sentences, abstractive models learn to generate language as well. The great success of deep learning algorithms on sequence learning tasks led to an increase in sequence to sequence learning algorithms with an attention mechanism. At the same time, the research area around transfer learning, transferring knowledge different domains, languages and tasks, increased over the past years. Word embeddings like GloVe or Word2Vec, are still useful for practical purposes, but were overtaken by a new generation of language models. In this thesis we explore two of the most prominent language models named ELMo and BERT, applying them to the extractive summarization task. We contribute a new ensemble model between abstractive and extractive summarization achieving, a new state-of-the-art on the English CNN/DM dataset. Instead of only working with an academic English dataset, we introduce a new dataset in German from the Deutsche Presse Agentur (DPA). This poses a challenge since real world datasets in German have less available resources through pretrained language models and inhibit more noise. We establish several abstractive and extractive summarization baselines.

Matthias Nitsche

Thema der Arbeit

Ansätze zu Deutscher, Abstrahierender Textzusammenfassung mit Deep Learning

Stichworte
Neural Networks, NLP, Summarization, Transformers, Language Modelling, ELMo, BERT, Transfer Learning

Kurzzusammenfassung

Textzusammenfassung ist ein etabliertes Sequenzproblem, unterschieden durch extraktive und abstrahierende Modelle. Extraktive Modelle lernen, Wörter und Sätze zu ordnen, wobei irrelevante Informationen eliminiert werden. Bei abstrahierenden Modellen kommt zusätzlich die Aufgabe der Sprachgenerierung hinzu. Durch den großen Erfolg von Deep Learning Methoden nahmen sequence-to-sequence Algorithmen zu, insbesondere mit dem sogenannten "Attention"-Mechanismus. Hinzu kommt ein größerer Fokus auf Wissenstransfer zwischen verschiedenen Domänen, Sprachen und unterschiedlichen Aufgaben. Word Embeddings wie GloVe oder Word2Vec sind nach wie vor nützlich, wurden aber durch eine neue Generation von Sprachmodellen abgelöst. In dieser Arbeit untersuchen wir zwei der bekanntesten Sprachmodelle namens ELMo und BERT und wenden sie auf die extraktive Zusammenfassung an. Wir bringen ein neues Ensemble-Modell zwischen abstrahierender und extraktiver Zusammenfassung ein, das einen neuen Stand der Technik auf dem englischen CNN/DM-Datensatz erreicht. Anstatt nur mit einem akademischen englischen Datensatz zu arbeiten, führen wir zusätzlich einen neuartigen Datensatz in deutscher Sprache ein, bereitgestellt von der Deutschen Presse Agentur (DPA). Dies stellt eine Herausforderung dar, da reale Datensätze in deutscher Sprache über weniger verfügbare Ressourcen durch vortrainierte Sprachmodelle verfügen und mehr Rauschen beinhaltet.

Contents

List of Figures

List of Tables

1 Introduction

Learning to generate fluent language is an interesting problem with a lot of different use cases. Generating document summaries from hundreds of articles, generating fake stories to influence voters, interactive dialogue agents for support or simply building a form of art that fascinates and potentially writes short stories. Natural language processing (NLP) had some high points in the last few years. The rise of neural language models and ever more complex neural network architectures covered a lot of ground, especially for language generation. Since the arrival of GPT-2, a transformer model by Radford et al. (2019) with 1.5 billion parameters, there is hope for neural models to write entire non trivial texts in the near future. See for instance the following generated summary:

> *The original site in Vallon-Pont-D'arc in Southern France is a Unesco World Heritage site and is the oldest known and the best preserved cave decorated by man. The replica cave was built a few miles from the original site in Vallon-Pont-D'Arc in Southern France. The cave contains images of 14 different species of animals including woolly rhinoceros, mammoths, and big cats.*

The summary is concise, grammatically correct, contains almost no repetitions and is factually correct. While impressive, the model itself is closed source and not re-trainable without computational power of a large GPU cluster. Models like these certainly pose some challenges and dangers. In the wrong hands, these models can easily generate entire campaigns of fake news or tweets in a matter of minutes. For automated journalism, however, this could mean condensing a collection of articles into one which could be published with minimal editorial effort. While not on the same level as works of professional journalists, it can also be helpful for automatically generated dossiers. Another German based summarization by our BertSum model based on Liu (2019):

> *Pompeo und kim yong chol wollen einen möglichen gipfel zwischen us-präsident donald trump und dem nordkoreanischen machthaber kim jong un*

vorbereiten. Nach einem abendessen am mittwoch soll es heute weitere gespräche geben.

The above summary is extractive in nature, ranking the source sentences and rearrange the best ones that describe the text perfectly. Since most academic endeavors in NLP are primarily concerned with English language, we try to transfer the used models to a German language corpus. The goal is to have a single document summarization system that can generate summaries from a range of news topics.

1.1 Research Question and Contributions

The research question and contributions revolve around an ensemble model of two state-of-the-art summarization algorithms and introducing a German dataset provided by the Deutsche Presse Agentur (DPA). The primary goal for this thesis is to answer questions around the task of summarizing documents with a German non-academic language corpus. The three questions this thesis tries to answer are:

1. How to establish a summarization baseline with a new German dataset using deep learning algorithms?
2. Do fine-tuning and feature extraction techniques improve performance for summarization?
3. Given that the non-academic dataset contains comparatively more linguistic noise, are we able to generate fluent summaries?

In NLP, fine-tuning and feature extraction are the most prominent two routes to go, since they focus on transferring highly performing systems trained on out of domain data. The leading discussion of this thesis is concerned with algorithms, focusing on language modelling and transfer learning in combination with sequence-to-sequence learning. The major contributions of this master thesis are as follows:

1. Introducing **two new datasets**, DPA Descriptions (116.603 text description pairs) with longer target texts and the DPA Headlines (199.328 text headline pairs) with shorter target texts.

2. Establishing a **baseline for German document summarization** using the ROUGE metric. The baseline includes extractive and abstractive state-of-the-art models by Peters et al. (2018); Gehrmann et al. (2018); Wu et al. (2019) and Liu (2019).

3. Introducing an ensemble model based on the copy transformer by Gehrmann et al. (2018), combining it with the fine-tuned BERT extractor introduced by Liu (2019), scoring **ROUGE R1 score of 41.75** on the CNN/DM corpus beating current state-of-the-art models.

First, the ensemble model is a proof of concept that given complex established ideas, we can easily push the ROUGE evaluation metric by a few decimals. Second, introducing a new real world dataset that is not extractive in nature in a different language poses a real challenge.

1.2 Chapter Overview

The thesis is split into six chapters. Neural document summarization is mostly based on sequence-to-sequence learning with attention. Thus the **first chapter** deals with the theory of advanced neural network architectures, focusing on sequence-to-sequence models, attention, recurrent neural networks and new self attention networks.

The **second chapter** focuses on language modelling, a rising field in NLP since 2018. It will cover transfer learning, word embeddings and deep language representations, concentrating on the rise of models like ELMo by Peters et al. (2018) and BERT by Devlin et al. (2018). Most of the currently used models for text summarization are based on ELMo and BERT.

In the **third chapter** our area of attention will be text summarization, covering topics like the ROUGE evaluation, followed by a discussion of abstractive and extractive summarization. In order to establish our ensemble model we will present two models, namely Bottom-Up summarization by Gehrmann et al. (2018) and fine-tuning BERT for extractive summarization by Liu (2019). The goal is to derive a simple ensemble called Bottom-Up BERT using both ideas.

The **fourth chapter** focuses on the results of Bottom-Up BERT on the CNN/DM corpus, comparing them to current state-of-the-art models. In order to run experiments on the

DPA dataset preprocessing and filtering must be transparent. At last we will present the results of several state-of-the-art algorithms on the DPA Descriptions and Headlines dataset.

The **fifth chapter** is a discusson on what we achieved, what we could not answer and critically reflect upon the limits and weaknesses of our approach.

The **last chapter** is a presentation of topics and ideas that will be relevant in the upcoming years. Specific topics include transfer learning, reinforcement learning and multi-document summarization.

2 Neural Networks

The basic building blocks for modern NLP models are neural networks. Their power lies in capturing a lot of subtleties classical approaches cannot. Most baselines use long-short term memory networks (LSTMs) Hochreiter and Schmidhuber (1997) while advanced architectures include the transformer Vaswani et al. (2017) and convolutional neural networks Ott et al. (2019) with positional embeddings to capture time dependencies. In this section we will give an overview of recurrent neural networks (RNNs), the transformer, convolutions and the general sequence-to-sequence framework. Vital to understanding all the modern architectures is the attention mechanism that was first coined in the context of neural machine translation systems.

2.1 Sequence-to-sequence Models

Sequence-to-sequence modelling is a general framework first established by Sutskever et al. (2014) in the area of neural machine translations (NMTs). Soon after Luong et al. (2015) created a working version of recurrent nets with attention. Since then sequence-to-sequence models are a standard architecture in the realm of translation, reconstruction and compression. One of the big differences between NMTs and document summarization is that NMTs need to attend to all the content and translate it in full. Summarization is a compression problem aligning long with short sequences. Most encoder decoder architectures in summarization build on the break through of Li et al. (2015); Chopra et al. (2015) and Nallapati et al. (2016) created what is now to be known as the encoder decoder with attention architecture for sequence-to-sequence models. sequence-to-sequence modelling is a general framework that consists of an encoder consuming a source sequence x consisting of tokens and a decoder consuming a target sequence y

$$x = [x_1, x_2, \ldots, x_n] \tag{2.1}$$

$$y = [y_1, y_2, \ldots, y_m] \tag{2.2}$$

where n and m are arbitrary lengths. In summarization $n > m$ typically holds, since compressing information is the main goal. The goal of the encoder-decoder is to model the probability $p(Y|X)$, meaning how likely is the sequence y given sequence x. Both sequences x and y are embedded into fixed length context vectors per token. Prior initialization of embeddings is an active research area, the assumption for now is that all vectors are initially drawn randomly from a uniform distribution. Given a set of sequences X and Y forming a unique vocabulary index $V : \mathit{Symbol} \to \mathbb{N}$ that embeds each token

$$\underset{1\times n}{\forall x \in X : V(x)} \to \mathbb{R}^{n\times k} \tag{2.3}$$

$$\underset{1\times m}{\forall y \in Y : V(y)} \to \mathbb{R}^{m\times k} \tag{2.4}$$

where k is some dimension and V a generalized mapping for a vector of symbols. Embeddings can be jointly trained in the model or held fixed using pre-trained embeddings. Learning a general mapping between two sequences and translating or aligning them can be expressed as

$$h_{enc} = \mathit{encoder}(X) \tag{2.5}$$

$$\mathit{align} = f(h_{enc}) \tag{2.6}$$

$$h_{dec} = \mathit{decoder}(Y, \mathit{align}) \tag{2.7}$$

Note that the encoder and decoder can be arbitrary differential functions, for the general notation we will sometimes fall back to LSTMs for illustration purposes. The concrete case for the a bidirectional LSTM forward pass

$$h_i = [\vec{h}_i^T; \vec{h}_i^T]^T, i = 1, \ldots, n \tag{2.8}$$

$$c_t = \sum_{i=1}^{n} \mathit{align}(y_t, x_i) h_i \tag{2.9}$$

$$s_t = f(s_{t-1}, y_{t-1}, c_t), t = 1, \ldots, m \tag{2.10}$$

Where h_i are the hidden states of the encoder and c_t an alignment function, commonly today attention, connected to a decoder s_t at time step t and the target label sequence y. The decoder needs to consume an aligned form that fits the dimensionality of an arbitrary target length, learning the probability of a given target token y_t. The objective

cost functions in neural language models are based on next word probabilities for each timestep. This is commonly with a softmax over all words V

$$\hat{y} = softmax(W \cdot s_t + b) \tag{2.11}$$

Where W is a weight matrix and b a bias vector to be learned and s_t is the decoder at timestep t. To train this function end to end the most common loss function for a softmax classification problem with $|V|$ classes is the cross-entropy loss

$$CE(\hat{y}, g) = -g \cdot log(\hat{y}) - (1 - g) \cdot log(1 - \hat{y}) \tag{2.12}$$

Where $\hat{y}$ is the expected per token probability and g the actual probability, e.g. a one hot vector of the word in the current sequence. The total error is then propagated back through the network with respect to all the gradients for each parameter and updated using gradient descent, also known as back-propagation. Some notes on the complexity of computing the softmax when vocabulary sizes grow very large. Using the Markov assumption that the current token is only dependant on the last c before going tokens which then yields the conditional probability

$$P(y_t|y_{t-1}, \ldots, y_{t-c+1}) = \prod_{t=1}^{n} P(y_t \mid y_1, \ldots, y_{c-1}) \tag{2.13}$$

Which can be computed with feed-forward neural network and a softmax, a function summing to 1 providing probabilities of the current token with respect to all other tokens.

$$P(y_t \mid y_{hist}) = \frac{exp(y_{hist}^T \cdot w_t)}{\sum_{i=1}^{|V|} exp(y_{hist}^T \cdot w_i)} \tag{2.14}$$

Where y_t is the current word embedding and y_{hist} the history vector so far of the whole vocabulary V (denominator) of the corpus and the embedding weight matrix W. When the vocabulary grows large the normalization term in the denominator even with windows is intractable to compute. Optimizations of the softmax include the hierarchical softmax or negative sampling.

A fully trained network is only a proxy for sequence generation. Since neural networks provide probabilities of possible next words, algorithms like greedy selection, viterbi

and beam search exists to generate sequences of words. In practice all the selection algorithms work in combination with the attention mechanism. In the subsequent section is a detailed presentation of the core ideas surrounding attention.

2.2 Attention

The motivation behind attention is to some extent inspired by how humans visualize images or text. Instead of focusing on one word at a time, humans pay attention to salient information of a sentence and whole paragraphs at once. Humans often view certain information as more relevant and highlight them immediately. When considering certain words like "playing" the expectation is that phrases like "the guitar" or "a video game" follow. It is a combination of what makes sense and what humans expect to happen next. In figure 2.1 is the first depiction of attention in RNNs introduced by Bahdanau et al. (2014).

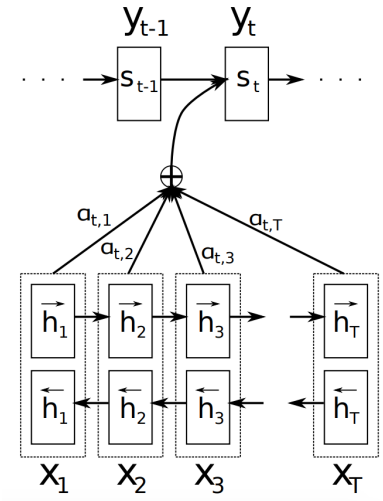


Figure 2.1: Attention as depicted in Bahdanau et al. (2014)

Similarly attention in deep learning is nothing but a vector of probabilities assigned to each word at the current stage. Attention is needed when sequences are longer than just a few words, since the last state of an encoder quickly forgets what it has processed in the beginning due to the vanishing gradient problem. Attention helps with keeping the salient focal points of tokens across longer sequences and helps the decoder to attend to

important tokens. Formally this means learning a probability for each source token

$$c_t = \sum_{i=1}^{n} \alpha_{t,i} h_i \tag{2.15}$$

$$\alpha_{t,i} = align(y_t, x_i) \tag{2.16}$$

Where the alignment function is a softmax connected with a feed forward network and a tanh non-linear function which can be an arbitrary alignment function, which in the simplest case is just the last encoder state

$$\alpha_{t,i} = \frac{exp(score(s_{t-1}, h_i))}{\sum_{j=1}^{n} exp(score(s_{t-1}, h_j))} \tag{2.17}$$

With the scoring function

$$score(s_t, h_i) = v_\alpha^T tanh(W_\alpha[s_t; h_i]) \tag{2.18}$$

In this case v_α and W_α are learned through end to end training of the entire network. Since then there has been wide research Xu et al. (2015); Luong et al. (2015); Cheng et al. (2016) distinguishing attention mechanisms with attributes such as local/global, soft/hard as well as self attention or intra-attention. Before going into detail into these distinctions let us give a brief overview of different attention scoring functions in order of year

$score(s_t, h_i) = cosine([s_t; h_i])$	Content based Graves et al. (2014)	(2.19)
$score(s_t, h_i) = v_\alpha^T tanh(W_\alpha[s_t; h_i])$	Additive Bahdanau et al. (2014)	(2.20)
$\alpha_{t,i} = softmax(W_\alpha s_t)$	Location based Luong et al. (2015)	(2.21)
$score(s_t, h_i) = s_t^T W_\alpha h_i$	General Luong et al. (2015)	(2.22)
$score(s_t, h_i) = s_t^T h_i$	Dot product Luong et al. (2015)	(2.23)
$score(s_t, h_i) = \frac{s_t^T h_i}{\sqrt{n}}$	Scaled dot product Vaswani et al. (2017)	(2.24)

The first attention mechanism by Graves et al. (2014) aligned each decoder step with an encoder by a simple cosine measure. Afterwards the scaling got a little more complicated with a linear layer and more free learnable parameters.

Self attention means that instead of relating important parts of the encoder to the decoder, each word in a sentence or document is attended to its own sub sequences. The original idea comes from Xu et al. (2015) doing this on images by first running a CNN over images and generating sequences as descriptions with a RNN. In 2.2 we can see how self attention learns correlations of the current head token and previous tokens.

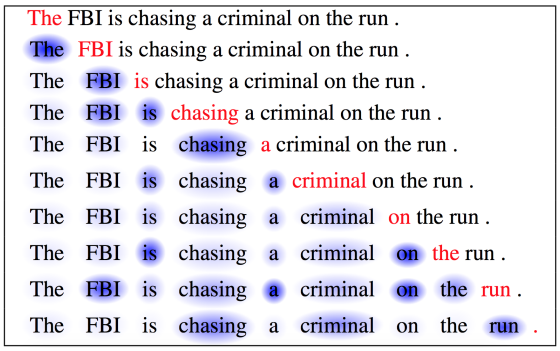


Figure 2.2: Attention activations as depicted in Cheng et al. (2016)

Soft/global and hard/local attention is distinguished by the simple fact that soft attention attends to all outputs of the encoder, while hard attention focuses on smaller patches, e.g. windows. Both have their place but hard attention is not fully differentiable with lesser computation overhead. Soft attention, proposed by Bahdanau et al. (2014), can get increasingly expensive the longer the sequences actually are sinc all tokens are considered at each time step. Global attention is somewhat similar to the soft attention. However local attention refers to the possibility to do hard attention with local patches keeping the model fully differentiable. Here, local patches mean, learning a position of the target word by sliding a context windows over the source.

Nowadays it is standard practice using attention and thus sequence-to-sequence learning evolved around the idea of attentional encoder-decoders. While LSTMS and CNNs use self attention to enhance their feasibility, the transformer model is entirely based on self attention with a so called multi head attention mechanism.

2.3 Recurrent Neural Networks

Recurrent neural networks (RNNs) are unrolled feed-forward networks over discrete timesteps. They are most useful when dealing with sequential information such as text

and have shown great success in language modelling, question answering and text summarization. In practice there are two feasible variants, long-short term memory networks (LSTMs) Hochreiter and Schmidhuber (1997) and gated recurrent units (GRUs) Chung et al. (2014). The problems with vanilla RNNs is that they suffer from the vanishing gradient problem, where for each timestep the gradient get vanishingly small and forgets what it learned. In figure 2.3 each LSTM cell at each timestep is a very complicated differential equation containing multiple ideas on how to deal with old and new knowledge

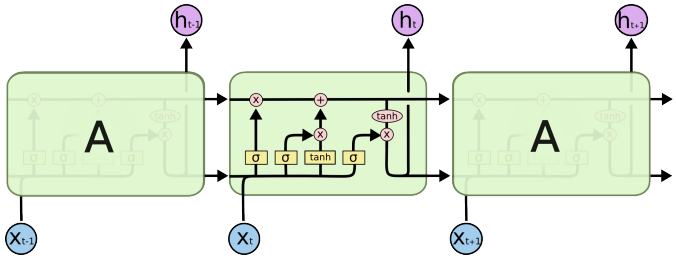


Figure 2.3: LSTM visualization by Olah (2015)

Formally LSTMs consist of a gating system filtering information by usefulness of the incoming memory and the current input word.

$$i^{(t)} = \sigma(W^{(i)} \cdot x^{(t)} + U^{(i)} \cdot h^{(t-1)}) \tag{2.25}$$
$$f^{(t)} = \sigma(W^{(f)} \cdot x^{(t)} + U^{(f)} \cdot h^{(t-1)}) \tag{2.26}$$
$$o^{(t)} = \sigma(W^{(o)} \cdot x^{(t)} + U^{(o)} \cdot h^{(t-1)}) \tag{2.27}$$
$$\tilde{c}^{(t)} = tanh(W^{(c)} \cdot x^{(t)} + U^{(c)} \cdot h^{(t-1)}) \tag{2.28}$$
$$c^{(t)} = f^{(t)} \circ \tilde{c}^{(t-1)} + i^{(t)} \circ \tilde{c}^{(t)} \tag{2.29}$$
$$h^{(t)} = o^{(t)} \circ tanh(c^{(t)}) \tag{2.30}$$

The hidden layer of the step before $h^{(t-1)}$ in combination with the current word x_t generates a new memory $\tilde{c}^{(t)}$ which is the function of the input gate $i^{(t)}$. The forget gate $f^{(t)}$ is there to asses whether the memory is useful and what parts can be dropped from it. The next step calculates what can be forgotten and what parts of the current input is used for the new memory $c^{(t)}$. The final hidden gate is there to filter out the unnecessary parts of $c^{(t)}$ making a final judgment resulting in the output for the next hidden layer. GRUs on the other hand are simpler versions of LSTMs by only keeping the importance

and forget gate

$$z_t = \sigma(W^z x_t + U^z h_{t-1} + b^z) \tag{2.31}$$
$$r_t = \sigma(W^r x_t + U^r h_{t-1} + b^r) \tag{2.32}$$
$$\tilde{h}_t = tanh(W^h x_t + U^h h_{t-1} \circ r_t + b^h) \tag{2.33}$$
$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \tilde{h}_t \tag{2.34}$$

In practice they have been proven to show strong results in sequence-to-sequence learning and neural machine translations in particular.

2.4 Self Attention Networks

The transformer introduced by Vaswani et al. (2017) is a new architecture and has become one of the standards in NLP. Since then there have been a lot of variations. Firstly, this section covers the original architecture and then highlights some improvements. Secondly it is explained how bidirectional encoder representations from transformers (BERT) by Devlin et al. (2018) are used for fine-tuning and pre-training. The novelty about transformer networks, as depicted in figure 2.4, is that it uses a new multi head attention mechanism in combination with positional embeddings and simple feed forward networks.

As explained before, each input sequence is embedded. Since the transformer attends to all information at once, it has no positional reasoning unlike LSTMs. The authors introduced positional embeddings and concatenate it with the source embeddings $inp_{emb} \in \mathbb{R}^{n \times d_{model}}$ and initialize the transformer

$$pe(pos)_{even} = sin(pos\ /10000^{2i\ /\ d_{model}}) \tag{2.35}$$
$$pe(pos)_{odd} = cos(pos\ /10000^{2i+1\ /\ d_{model}}) \tag{2.36}$$
$$pos_{enc} = [pe_{even}, pe_{odd}] \in R^{n \times d_{model}} \tag{2.37}$$
$$inp = [inp_{emb}; pos_{enc}] \tag{2.38}$$

Where $pe(pos)_{even}$ is a sine wave for each dimension with d_model when the position is even and $pe(pos)_{odd}$ is a cosine wave for each dimension with d_model when the position is odd, resulting in pos_{enc}. The result is the concatenation of the input embedding and

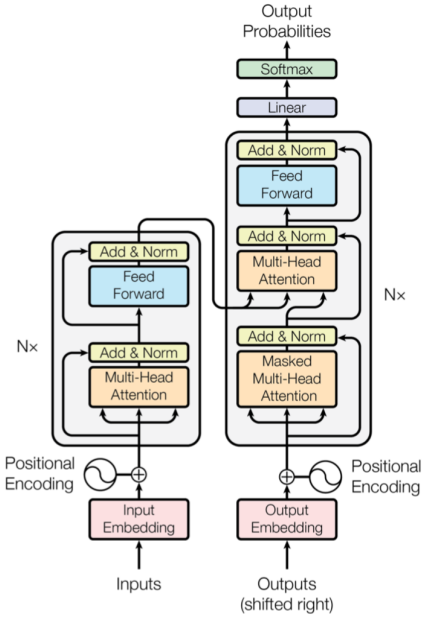
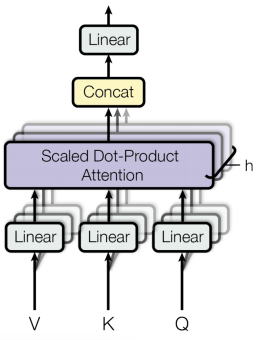
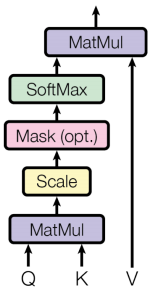


Figure 2.4: Transformer architecture in Vaswani et al. (2017)

the positional embedding. The input is then split into key, query and value and feed into a multi head attention function. Each of a total of N so called heads processes all the different parts of the sequence and feeds it into the decoder. As depicted in figure 2.5, multi head attention is a linearized self attentive function using scaled dot product attention, which is one of the before mentioned attention mechanisms.



(a) Multi head attention



(b) Scaled dot product

Figure 2.5: Transformer from Vaswani et al. (2017)

The interesting part is actually how multi head attention works. Formally

$$MultiHead(Q, K, V) = [head_1; \ldots; head_h]W^O \tag{2.39}$$

$$head_i = softmax(\frac{(QW_i^Q)(KW_i^K)^T}{\sqrt{d_k}})(VW_i^V) \tag{2.40}$$

where learnable parameters are $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$ and $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$. Each head is the result of a split of the whole sequence. Since the base model has a dimension of 512 and 8 heads, the resulting base dimensions d_k, d_v etc. have dimensionality $512/8 = 64$. Each head attends to different parts of the sequence and concatenated into the original dimension of 512. This concatenated scaled dot product attention is fed to a 2 layer feed forward network and resembles the linear combination of all heads. The heads basically act as a contextual window function.

$$\textit{ffn}(x) = ReLU(xW_1 + b_1)W_2 + b_2 \tag{2.41}$$

The *ffn* recombines all the head mechanisms projecting them to a final softmax layer over the whole vocabulary. For a language modelling task the standard cross-entropy loss is used to back propagate the error. The attention mechanism aligns a transformer encoder with a transformer decoder even in cases where there are unequal lengths source and target dimensions. Note that transformers are not sequential models, instead building up a whole context and recombining the whole sequence at once. Ideas by Radford and Salimans (2018) combined recurrent neural networks with transformers, eliminating the vanishing gradient problem and improving performance. They are however c onstrained to sequence sizes that are as long as the model dimension, in the standard case 512. Models like the GPT-2 by Radford et al. (2019), use a 4096 model size with much more and larger heads. This leads to improved performance on much larger sequences.

Transformer models are very sensitive to datasets and hyperparameters. We used Popel and Bojar (2018) as a reference when tuning specific parameters. Special care must be taken when choosing the following parameters

1. Learning rate + warm-up steps (Noam scheme)
2. Number of layers
3. Number of attention heads

4. Feed-forward connections
5. Parameter priors - Xavier uniform
6. Residual Dropout and Label smoothing
7. Batch size

In the original paper the transformer adheres to the Noam training scheme, choosing a small learning rate at the beginning and then increase it linearly for a few thousand warm up steps. When hitting the warm up steps limit the learning rate slowly starts to linearly decrease until training finishes as depicted in figure 2.6.

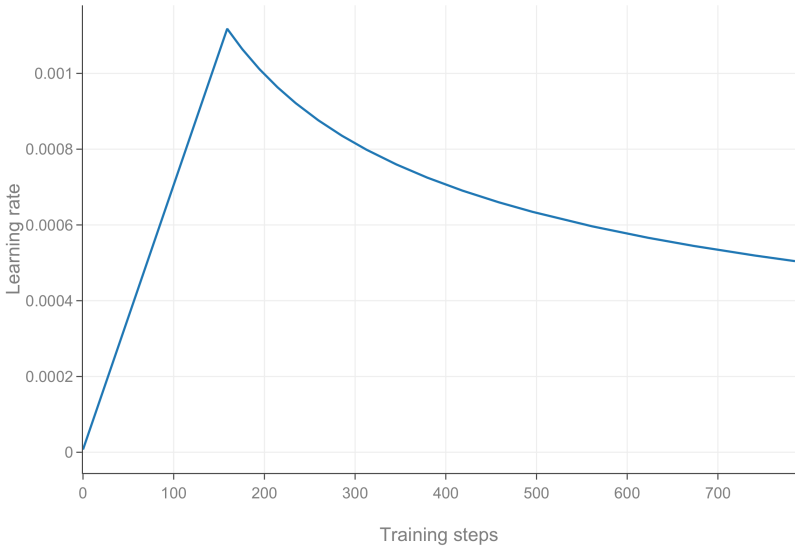


Figure 2.6: Noam learning rate during training

The motivation behind this is that very large updates at the beginning helps the model to learn more quickly. After a while the learning rate is too large and needs to slowly decrease to not overshoot the goal. This can lead to zero gradients or a sudden increase of the cross-entropy loss.

Another recent idea stemming from the self attentive transformer model is using CNNs. Gehring et al. (2017) introduced CNNs to sequence-to-sequence modelling. Based on the idea of sequence-to-sequence CNNs stems the idea of dynamic self attentive convolutions

by Wu et al. (2019). Instead of recombining three linear queries, only one linear query with a convolutional kernel in the center is used. There are two key concepts, the light convolution and dynamic convolutions with an additional set of dynamic weights. The formulation is as follows

$$LightConv(X, W_{[\frac{cH}{d}]}, i, c) = DepthwiseConv(X, softmax(W_{[\frac{cH}{d}]}), i, c) \tag{2.42}$$

$$DepthwiseConv(X, W_c, i, c) = \sum_{j=1}^{k} W_{c,j} \cdot X_{(i+j-[k+1]),c} \tag{2.43}$$

Where the light convolutions is a parameter tied, by a factor of $\frac{cH}{d}$, depth-wise convolution that reduces the number of parameters by several orders of magnitude. Light convolutions unlike the scaled dot product self attention do reuse the same parameter context. The problem of reusing the same context is that it is not updated per timestep. This problem however is addressed with dynamic convolutions that tie an additional linear layer to the light convolution. This linear layer is essentially a time dependant function that acts on the before going learned convolution. It still does not require to take the whole context of the sequence into account. We will use dynamic convolutions using the sequence-to-sequence toolkit by Ott et al. (2019) to establish a baseline.

2.5 Discussion

In this chapter we gave a detailed overview of the seq2seq framework aligning a source and target sequence via attention using a wide variety of models with a particular focus on self attention networks. For neural text summarization, language generation is the hardest part. Classical approaches via symbolic execution work well however neural approaches spark a little more randomness and surprise into the equation. This often leads to much more fluent language.

Summarization in essence is a sequence alignment task with a language generator on top. Learning a compression from long to short documents by elimination of source words and minimize the cross-entropy loss for next word prediction is at the center of these algorithms. Self attention networks are especially relevant since they have shown superior results to comparable models. The transformer is not a sequential network, relying on positional embeddings to model range dependencies. Moreover the transformer is very prone to hyperparameter tuning which makes them difficult to train correctly. Radford

et al. (2019) have shown that transformer models are mostly never at full capacity, either because the datasets are too small and therefore over-parameterized. The copy transformer by Gehrmann et al. (2018) is the primary model in the experiment section. The model is based on a sequence-to-sequence transformer with a few additional perks, like copying, reducing repetition and length constraints.

A special focus on attention is important since all of the following models will use different forms of it. It first appeared in 2014 introduced by Bahdanau et al. (2014) and was gradually improved over time. Attention can be used to focus on salient words in the documents, tackling the vanishing gradient problem by not forgetting any of the inputs, while aligning to sequences of unequal lengths. The downside is that attention greatly improves runtimes since for each query vector there is an additional linear layer added.

3 Language Modelling and Embeddings

Language modelling is a long standing NLP problem and has recently gotten a lot of attention due to its success in downstream applications. Neural text summarization is no exception to that. Liu (2019); Gehrmann et al. (2018); Edunov et al. (2019) have established several models beating state-of-the-art systems using relatively simple techniques based on fine-tuning and pre-training to language models such as BERT or ELMo. The main idea was first introduced by Bengio et al. (2001) contributing a mathematical formulation for a neural language model that is end to end trainable. In this chapter we will discuss general language modelling objectives and recent developments on state-of-the-art language models such as ELMo by Peters et al. (2018) or BERT by Devlin et al. (2018).

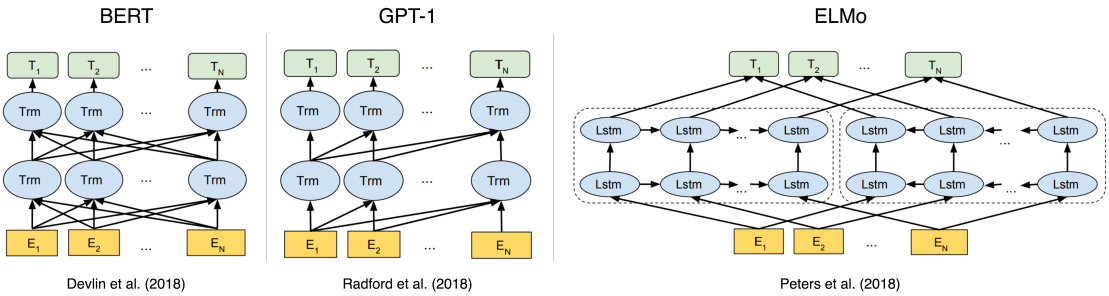


Figure 3.1: Pre-training architectures adapted from Devlin et al. (2018)

In language modelling there are two principal distinctions, language model fine-tuning and language modelling as feature extraction (embeddings). For fine-tuning the first step is pre-training a general language model. Using this pre-ptrained language model to then fine-tune to a specific target task. The tasks at hand range from question answering to text summarization and are often pluggable given the correct objective. The pre-trained model must learn general feature spaces about language such as part-of-speech and its relations between words, word hierarchies, groups of similar words and next token or sentence probabilities. In feature extraction, often synonymous for embeddings, word/characters/sentences or entire document representations are extracted as

fixed length context vectors with dimensionality $\mathbb{R}^{n \times k}$ where k is a hyperparameter. Each dimension in an embedding of a word is said to be a specific context of that word given all the other words. Levy and Goldberg (2014) call it an optimal point-wise mutual information (PMI) matrix. Formally the language modelling objective is defined in terms of the most likely probability of the before going words $[x_1, \ldots, x_{i-1}]$ given the current word x_i. By the chain rule of factoring a probability by their conditional probabilities it follows

$$p(x_1, \ldots, x_n) = \prod_{i=1}^{n} p(x_i \mid x_1, \ldots, x_{i-1}) \tag{3.1}$$

For practical reasons it was shown that it is better to do a forward language model and backwards language model

$$p(x_1, \ldots, x_n) = \prod_{i=1}^{n} p(x_i \mid x_{i+1}, \ldots, x_n) \tag{3.2}$$

concatenating both to have a sense of contextualization. The objective in language modelling is to calulate the conditional joint probability of a given token, followed or following a sequence. The common language model with LSTMs minimizes the negative log likelihood loss of next and previous token probabilities

$$\mathcal{L} = -\sum_{i=1}^{n} log\ p(x_i \mid x_1, \ldots, x_{i-1}) + log\ p(x_i \mid x_{i+1}, \ldots, x_n) \tag{3.3}$$

$\mathcal{L}$ is the negative log likelihood of a forward and backward language model. Using log probabilities makes the procedure numerically more stable. The biLSTM for language modelling was introduced in several instances such as the AWD-LSTM by Merity et al. (2017) or ELMo by Peters et al. (2018) both achieving state-of-the-art results. The AWD-LSTM builds an ensemble of two language models, since a next word probability cannot be learned when the next coming word is already defined.

In the following sections we will discuss transfer learning, when to use feature extraction and when to use fine-tuning. This entails to draw out a rough theoretical foundation about pre-training, fine-tuning and feature extraction, using examples from recent language models like ELMo and BERT. First let us quickly review the most occurring cost functions and evaluation metrics for language models and embeddings.

3.1 Evaluation

The most common metrics for measuring a language models prediction capacities is the cross-entropy and perplexity. Most language models objectives require a function that measures the next word probability given the current word, based on a context the word is in. Recall that the softmax is

$$p_\theta(x \mid c) = \frac{exp(h_c^T w_x)}{\sum_{i=1}^{|V|} exp(h_c^T w_i))} \tag{3.4}$$

where c is the context of x, h_c and w_i are functions of c and x parameterized by θ and $h_c^T w_x$ is the unnormalized logit. The softmax yields a probability distribution summing to 1. The softmax is an expensive function when the vocabulary size grows. In language modelling, keeping as many vocabulary words V as possible is important which entails using approximations of the softmax like hierarchical softmax or negative sampling.

Cross-entropy over words is defined over two distributions p, the true distribution as observed in the training data and q the learned distribution. Expanding this idea to language modelling this measures the true probability of the current word x and the context c as described above

$$H(p, q) = -\sum_x p(x)\ log\ q(x) \tag{3.5}$$

$$\mathcal{L}_\theta = -\sum_{i=1}^{|V|} y_i\ log\ p(x_i|c_i) \tag{3.6}$$

$$= -log\ \frac{exp(h_c^T w_x)}{\sum_{i=1}^{|V|} exp(h_c^T w_i))} \tag{3.7}$$

Since x_i is the current word it is at the same time the label as a one hot encoded vector and c_i is our prediction. The cross-entropy can be defined in terms of the negative log-likelihood of the softmax probabilities. Minimizing the cross-entropy loss results in high probabilities for correct next words.

Perplexity is defined over a discrete probability distribution p and is a measurement of how well it learns the context of the dataset. The perplexity is defined in terms of the

exponentiation of the entropy $H(p)$.

$$2^{H(p)} = 2^{-\sum_x p(x)\ log2\ p(x)} \tag{3.8}$$

where x are the words and p the probability distribution. In the case of a text looping over all words and measuring the probability of each word results in

$$2^{H(p)} = \sum_{i=1}^{N} p(w_i)^{-\frac{1}{N}} \tag{3.9}$$

In general, the lower the perplexity the higher the individual probability of each word. A good language model predicts high probabilities for a word when it actually occurs.

3.2 Transfer Learning

The following section is heavily based on the thesis of Ruder (2019). Transfer learning deals with the task of aligning a source domain D_s and a task T_s to a target domain D_t and task T_t. D consists of a feature space $X = [x_1, x_2, \dots x_n]$, a marginal probability distribution $P(X)$ over the feature space and a prior distribution $P(Y)$ called label space. Both domains have a conditional probability distribution $P(Y|X)$ that maps sequences of X to labels of Y. The main objective then becomes to learn the target domain D_t by learning $P_t(Y_t|X_t)$ given the information from D_s and T_t. Two cases are relevant for this thesis, namely sequential transfer learning and domain adaptation.

Sequential transfer learning occurs when the label spaces $Y_s \neq Y_t$ e.g. when the target task has different labels than the source task. For instance training a source Wikipedia language model with the next word prediction task, while the target tasks learn text summarization from the same corpus. Sequential transfer learning has a pre-training and adaptation phase, learning the source task and then adapt the target to it. Typically pre-training should lead to generalized/universal representations of language that is adaptable to specific corporas that are not in the trained domain. However there is no free lunch and obtaining an universal representation that is useful on all NLP tasks is not possible. When the source domain contains more data than the target domain, it is best to use sequential transfer learning. There is also active research to make one language model as adaptable as possible to as many tasks as possible.

Domain adaption is when the probability distributions $P_s(X_s) \neq P_t(X_t)$ which means that both corpora discuss different topics. For instance when pre-training a Wikipedia language model as the source and using a news corpus for fine-tuning on the same task. The words and the way the content is described will differ. For instance trainining a Wikipedia language model as the source task and learning a text summarization model as the target task on a news domain would make use of sequential transfer learning and domain adaptation at the same time. In the following we will discuss primarily pre-training and two adaptation methods, namely feature extraction and fine-tuning.

3.3 Word Embeddings

The goal with pre-training is to train universal models that can be used successfully in as many different NLP tasks as possible. There are three distinct categories, unsupervised, distantly supervised and supervised pre-training. Since most text datasets are not annotated, unsupervised pre-training learning directly from unstructured text is more viable. In theory word embeddings help to learn general features before they are plugged into a specific architecture. Thus word embeddings are learned prior distributions over words. First, this speeds up the training process leaving room for the actual task. Second, with more data in a source domain there is a chance of better word representations leading to an improved quality in downstream applications. The methodologies behind pre-training fall into two categories, e.g. language modelling and matrix factorization. Matrix factorization methods include

1. Latent Semantic Analysis (LSA) by Deerwester et al. (1990)
2. Brown Clustering by Brown et al. (1992)
3. Latent Dirichlet Allocation (LDA) by Blei et al. (2003)
4. Pretrained Hidden Markov Models (HMM) by Huang and Yates (2009)

As explained before the primary focus is on neural language modelling and embeddings. Since we already introduced language modelling above let us contrast the factorization based methods with word based contextual embedding models like Word2Vec, GloVe or FastText.

Word2Vec by Mikolov et al. (2013) was the first popular word embedding that was widely available and distributed and simply worked in most downstream applications, pushing accuracy scores by a few percents. Word embedding methods rely on the definition of a context and a loss function that is able to capture this context. The most prominent word embedding objectives are skip-gram and continuous bag-of-words (CBOW).

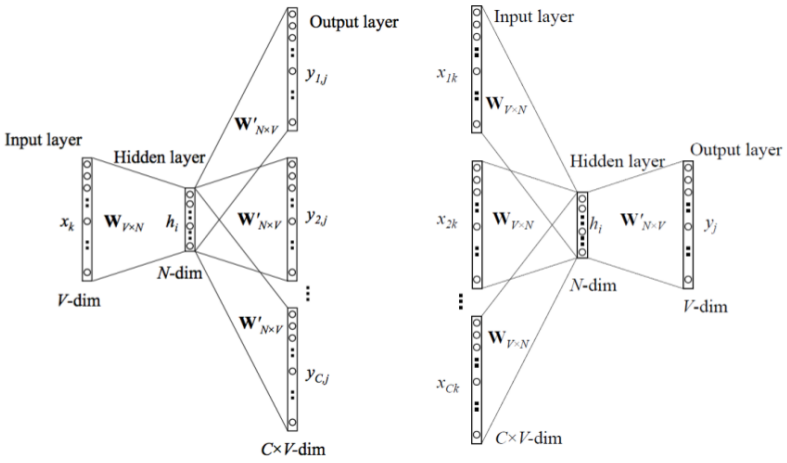


Figure 3.2: Skip-gram and CBOW Mikolov et al. (2013)

Skip-gram, as depicted in figure 3.2, predicts the context of a center word w_i over a window c and $[w_{i-c}, \ldots, w_i, \ldots w_{i+c}]$ are context labels

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} \log\, p(w_{t+j}|w_t) \tag{3.10}$$

Computing the probability p over all words is intractable, instead using skip-gram with negative sampling (SGNS)

$$\log\sigma(\mathrm{v'}_{w_O}^{T} v_{w_I}) + \sum_{t=1}^{k} \mathbb{E}_{w_i} \sim P_n(w)[\log\sigma(-\mathrm{v'}_{w_O}^{T} v_{w_I})] \tag{3.11}$$

This results in $2\dot{C}$ context words and a few more negative words constrained by k samples to discriminate. Here, σ is the sigmoid function, drawing k samples from the negative or noise distribution $P_n(w)$, to distinguish the negative draws $\mathbb{E}_{w_i}$ from the target word w_O drawn from the context of w_I. The idea behind this is, that the current word is

defined via its contextual window and not via all the words that are not. However the negative samples are needed so that the model learns what should not be considered as a contextual word. Often it is very useful to transform words into segments of symbols rather than the word itself. A good example would be stemming. A trend from the neural machine translation architecture is to use byte-pair encoding.

Byte-pair encoding (BPE) introduced by Sennrich et al. (2015), is a data compression algorithm and was quickly adopted for language tasks. The basic idea is to encode each word of the input sequence into word segmentation's such that rare and unknown words can be resolved. Often rare words provide useful meaning. Slight variations in the stem of a word and it will not be recognized. More dire is the situation on test dataset where there are potentially many out-of vocabulary words. Instead decomposing words into their optimal sub pieces, trying to find the parts of words with similar character segments. The use of BPE results in fewer words overall therefore greatly reduces oov words and therefore reducing the model size.

Global Vectors for Word Representation (GloVe) by Pennington et al. (2014) is an extension to the original skip-gram with negative sampling approach used in Word2Vec. GloVe incorporates global co-occurrence counts during training. Instead of using the raw probabilities like Word2Vec, it is possible to consider the ratios of vectors given the global co-occurrence of two words. The authors derived several enhancements including changing the loss function to a mean squared error (MSE) loss.

$$\mathcal{L}_\theta = \sum_{i,j=1}^{|V|} f(C_{ij})(x_i^T \tilde{x}_j + b_i + \tilde{b}_j - log\ C_{ij})^2 \tag{3.12}$$

x_i is a word and $\tilde{x_j}$ its context word with their corresponding biases. C_ij is the count where w_i occurs with the context $\tilde{x_j}$. f is a weighting function that assigns higher scores to words that occur in between rare and frequent words. Since Word2Vec and GloVe there are several new approaches to create embeddings: BPEmb Heinzerling and Strube (2017) that train BPE embeddings and provide them in 275 languages or FastText Joulin et al. (2016) a non neural approach using n-grams also providing a dozen multi lingual models.

3.4 Deep Language Representations

In recent years due to the success on downstream applications, the trend for ever larger and more complex neural networks replacing smaller shallow networks and bag-of-word models. The quality of results went high especially in areas where matrix factorization methods do a very poor job, such as language generation. This however comes at great costs such as computational complexity with training regimes of a few weeks and often unreproducible results. The size of a lot of models also means that only a few actors can actually train them from scratch. Since Howard and Ruder (2018) kicked off the idea of fine-tuning generalized language models with great success, multiple new ideas popped up in a very short amount of time. All with increasing model sizes and ever more complex ideas and architectures for pre-training. In chronological order

1. Universal language model fine-tuning (ULMFit) by Howard and Ruder (2018), used the AWD-LSTM introduced by Merity et al. (2017). They used new techniques like discriminative fine-tuning, slanted triangular learning rates and gradual unfreezing of layers successfully fine-tuning a generalized language model to a specific classification task.

2. Embeddings from language models (ELMo) by Peters et al. (2018) tackled some of the issues of ULMFit by making it simpler to fine-tune to new tasks using a biLSTM.

3. Transformer decoder (GPT-1 / GPT-2) by Radford and Salimans (2018); Radford et al. (2019) trained a state of the art language model that could be easily fine-tuned to multiple NLP tasks. GPT-1 is a smaller version of GPT-2. GPT-2s training dataset as well as the model was never published.

4. Bidirectional encoder representations from transformers (BERT) by Devlin et al. (2018) is a huge bidirectional transformer that was trained on a cloze task and is almost universal to some degree.

For this work we will take a closer look on ELMo and BERT. GPT-1 and GPT-2 are too big to retrain and were never published for multiple languages. A few notes on the above models. A recent study by Emma Strubell and McCallum (2019) argue that the recent trend in NLP tend to focus on ever larger models, costing a lot of electricity and money. Relevant for this thesis

1. ELMo was trained on 3 NVIDIA GTX 1080 GPUs for 2 weeks (336 hours) and costs $433 - $1472
2. NVIDIA reports that they can train a BERT model in 3.3 days (79.2 hours) using 4 DGX-2H servers, totaling 64 Tesla V100 GPUs and costing $3751 - $12,571
3. GPT-2 has 1542M parameters and is reported to require 1 week (168 hours) of training on 32 TPUv3 chips costing $12,902 - $43,008

For real world applications and different languages all of the above models would need to be specifically retrained on different data and languages. Replicating one of these is a time and power consuming task, requiring a lot of money. While not disturbingly high, these values should at least provide food for thought. Deep representations are different from word embeddings since it is not possible so simply extract features for a single word. A word is always seen in the context of its full sequence and consequently needs access to the sequence when encoding the representation. After the encoding it is possible to use single word embeddings. According to Ruder (2019) feature extraction and fine-tuning differ as follows: Feature extraction sets the learning rate η for all layers of the embedding to zero, meaning only the source task is trained

$$\eta_t^{(l)} = 0, \forall l \in [1, L_S] \forall t \tag{3.13}$$

By contrast fine-tuning sets the learning rate during training for all layers to some value.

$$\eta_t^{(l)} > 0, \exists l \in [1, L_S] \exists t \tag{3.14}$$

Here it is possible to use techniques such as gradual unfreezing of layers Howard and Ruder (2018). This opens up many possibilities to shift parameters trained on a domain with a lot of knowledge and transfer that knowledge while gradually moving the parameter space to the target domain. Often this can be done by a sharp increase of the learning rate, with a linear decay after some time and only training layers gradually, e.g. unfreezing them.

3.4.1 Embeddings from Language Models

Embeddings from Language Models (ELMo) by Peters et al. (2018) is a large biLSTM designed for solving syntactical language tasks. ELMo is a multi-layer bidirectional LSTM that can be used as a feature extractor in any downstream NLP task. The cost function ELMo maximizes is the negative log-likelihood of a forward and backward language model parameterized by an LSTM

$$\vec{\mathcal{L}} = p(x_i \mid x_1, \ldots, x_{i-1}; \theta_e, \vec{\theta}_{LSTM}, \theta_s) \tag{3.15}$$

$$\overleftarrow{\mathcal{L}} = p(x_i \mid x_{i+1}, \ldots, x_n; \theta_e, \overleftarrow{\theta}_{LSTM}, \theta_s) \tag{3.16}$$

$$\mathcal{L} = -\sum_{i=1}^{n} (log\ \vec{\mathcal{L}} + log\ \overleftarrow{\mathcal{L}}) \tag{3.17}$$

The biLSTM is a pretty common architecture, what differentiates ELMo from the rest is how it is used. ELMo stacks the hidden states of each layer l into $2L + 1$ vectors for each word x_i. Note each layer is present in forward and backward direction e.g. $h_{i,l} = [\vec{h_{i,l}}; \overleftarrow{h_{i,l}}]$. The vectors are then squashed into a single vector

$$h_{i,0:l} = [h_{i,0}\ ; h_{i,1}\ ; \ldots\ ; h_{i,l}] \tag{3.18}$$

The original ELMO architecture has 3 layers, 2 biLM layers and one embedding layer each with dimensionality 1024 resulting in a 3072 sized vector. In contrast to standard embeddings that assign a fixed length vector based on a word, ELMo needs the whole sequence. The usage of a forward and backward LSTM makes it possible to capture previous and next word information. Recall that an LSTM gradually builds up its hidden state based on the before going hidden state, which means that the feature extraction of a single token needs the whole sentence or document. By example with two sentences

$$sent_a = [sitting, on, a, bank]$$
$$sent_b = [keeping, money, in, a, bank]$$

and extract the features for the word *bank*

$$ELMo(sent_a, bank) = \vec{bank}_1 \in \mathbb{R}^{3072}$$
$$ELMo(sent_b, bank) = \vec{bank}_2 \in \mathbb{R}^{3072}$$
$$where\ \vec{bank}_1 \neq \vec{bank}_2$$

The word is the same, the result are two different real valued vectors based entirely on the passed context. The second important contribution is the way ELMo can be fine-tuned to a target task. By just extracting the features $E(R_i, \theta_e)$ from ELMo and inserting it in a downstream task with parameters θ_{task} there can be a considerable domain drift. Hence Peters et al. (2018) provided a way to fine-tune ELMo to a specific target task

$$ELMo_i^{task} = E(R_k, \theta^{task}) = \gamma^{task} \sum_{l=0}^{L} s_l^{task} h_{i,0:l} \tag{3.19}$$

Where γ is an optimization parameter and s_l^{task} a specific linear combination adjustable for the specific task. This makes ELMo suitable for fine-tuning as well. In the simplest case the linear combination can just be another language model on a new domain corpus. The downside is that $ELMo^{task}$ needs to be fine-tuned to each new task and to each new dataset. A part from this, what does ELMo actually learn? Given that it solves question answering, textual entailment, co-reference resolution or named entity extraction with scores yielding state-of-the-art-results.

3.4.2 Bidirectional Encoder Representations from Transformers

After Radford and Salimans (2018) published their transformer decoder language model, the open question was how to create a bidirectional transformer architecture.Devlin et al. (2018) took the opportunity and created a bidirectional encoder representations from transformers or short BERT. The problem with bidirectional transformers is that they are not time step dependant like LSTMs. Transformers consume the input at once. The problem is that the next word probability is already known from the backward language model. It follows that BERT cannot directly learn the next word probability task. Instead the authors coined the term masked-lm where randomly selecting a token and trying to predict it is possible. In the literature this is also known as a cloze task e.g.

Mary sat on a ? eating her ?

where some possible combinations would be *chair* and *lunch*. Consecutively BERT needs a way to mask its input representation. Additionally to learning the masking for single words, BERT also learns a next sentence prediction expecting two sequences divided by a delimiter. In figure 3.3, BERT expects sentence and sequence delimited input

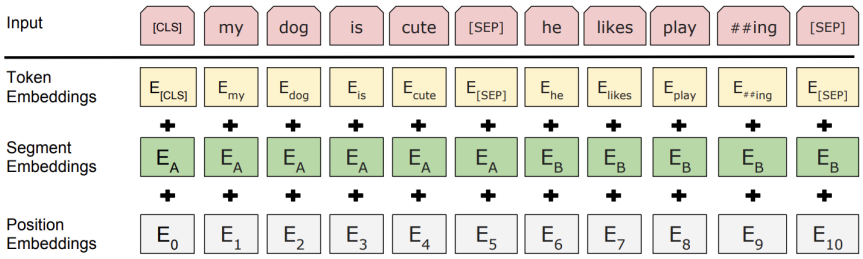


Figure 3.3: Bert input representation Devlin et al. (2018)

It consists of a token embedding for individual words, a segment embedding for sentences and a position embedding (since transformers have no sense of position). The input always starts with a $[CLS]$ token at the beginning and a $[SEP]$ token to delimit two sequences and mark the end of the pairs. The segment embedding marks the sentences such that the first and second sequence divided by the $[SEP]$ token have two marking. The masking works like this: with 15% chance randomly select a token in the sequence and replace it 80% of the time with the $[MASK]$ token, 10% at random from the vocabulary and 10% leave the token unchanged. The motivation by not choosing anything comes from the problem that the $[MASK]$ token will create a mismatch between pre-training and fine-tuning. As depicted in figure 3.4 fine-tuning with BERT becomes surprisingly simple

After pre-training a BERT model on a large corpus in a specific language, the input representation can be chosen given the task. One nice thing is, the authors published several versions of the model, including a multilingual version where BERT understands multiple languages to some degree. They used BPE to encode the tokens, reducing vocabulary and extracting meaningful word segmentation's. For BERT, fine-tuning is superior to feature extraction. However you can extract different representations from BERT and still have very good results in downstream applications. The concatenation of the last four hidden states, by ablation, was the most successful feature extraction method. In the end, what does BERT actually learn? Goldberg (2019) found out that

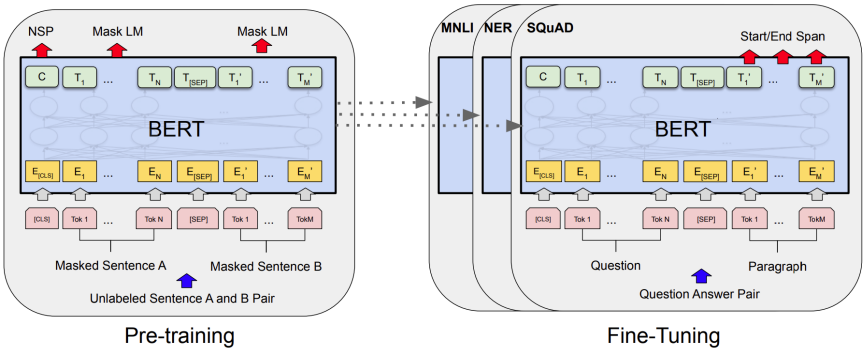


Figure 3.4: Bert for fine-tuning Devlin et al. (2018)

BERT actually learns hierarchical representations and long term dependencies. They also found that bigger is not necessarily better by comparing the large and base version finding that syntactical word level tasks are better in the base version. Jawahar et al. (2019) looked a bit deeper and found that each of the 12 BERT layers learned to perform different tasks on semantic and syntactic evaluation datasets.

3.5 Discussion

Since 2018 a new generation of language models and embeddings have appeared. For NLP this meant that the transfer learning moment finally arrived. Language modelling is a very general task with which it is possible to learn many morphological features of text. A single language model can capture hyponymy, synonymy, part-of-speech and structural dependence between words and sentences, e.g. context. Important models in this area include ELMo by Peters et al. (2018) and BERT by Devlin et al. (2018) pushing all NLP task to new state-of-the-art results while providing open source versions ready to use for researchers. While ELMo contextualizes words given the entire sentence, BERT uses the cloze prediction task.

With the rise of pre-training and fine-tuning, first proven to effectively work in NLP with ULMFit by Howard and Ruder (2018), a vital question arises: Do we extract the features or fine-tune language models? Peters et al. (2019) answered this question regarding the language models BERT and ELMo and found that BERT should be fine-tuned for a specific task and ELMo should be used with feature extraction.

The most prominent metric of language modelling is the perplexity metric which is connected to the cross-entropy. The overall task is to assign high next word probabilities to correct words, in case of word embeddings with a window k. While this makes sense in the domain of language modelling, it becomes clear later that the primary evaluation metric of summarization differs from the next word prediction task. In the context of sequence-to-sequence learning, sequential transfer learning by extracting features is most prominent. Recent ideas by Edunov et al. (2019); Zhang et al. (2019a) have shown that this can be learned jointly in and end to end fashion.

Later we compare how ELMo is used for extractive summarization via feature extraction by Gehrmann et al. (2018) and how to use BERT via fine-tuning to train a sentence extractor by Liu (2019). This makes ELMo and BERT directly comparable on extractive summarization. The probability masks for the next word or sentence prediction are directly applicable in hybrid extractive-abstractive summarization systems.

4 Neural Text Summarization

In this section we will lay out the basic building blocks of neural abstractive text summarization. The thesis works on English and German language, from two different datasets. The English dataset is the CNN/DM corpus with news articles from CNN and daily mail. Much more interesting is the transfer to German language, based on the DPA dataset. Before diving into specific topics and the presentation of the dataset let us develop a definition of summarization. Automatic summarization shortens texts while keeping the main points and ideas intact. Relevant points include:

1. Length - The summary should be short and concise
2. Coverage - All relevant information should be covered
3. Repetition - Words and information should not repeat
4. Factual correctness - Information should be correct
5. Syntax - Language should be grammatically correct
6. Fluency - Language should be fluent and in full sentences

Summarization is classified as extractive and abstractive summarization. The objective of summarization is to consume a source sequence X of words $(x_1, x_2, .., x_{|X|})$ and align it with a - usually shorter - target sequence Y of words $(y_1, y_2, .., y_{|Y|})$ sharing a vocabulary V. The objective goal is then to create a function f that finds a mapping $f : X \times Y \rightarrow T \in \mathbb{R}^m$ where T is essentially any permutation of possible sentences constrained to length m.

$$f(x, Y) = \underset{y \in Y}{\operatorname{argmax}} f(x, y) \tag{4.1}$$

For extractive summarization, there is an additional constraint, instead of generating words from the whole vocbulary, the objective is to rank sentences or words from the

source $X = X_{[1,..m]}$ that have the highest overlap with the target Y. The following section discusses the evaluation of summarization using the ROUGE score and cover extractive summarization.

Traditionally summarization methods are based on extractive summarization. Abstractive summarization is the primary goal, however sentence rewriting and language generation are long standing unsolved problems. Recently, neural approaches showed some promising results in that area. Extractive summarization commonly was tackled with sophisticated metrics such as cue words, title or location based methods where the most salient sentences are picked. Often extractive summarization is a preprocessing step to abstractive summarization, for it is easier to generate language from a few predefined sentences than from larger unstructured texts. The language representation has many forms such as graphs, cooccurence frequencies and latent vector space models. All of them have the goal to represent text in a way that it is possible to rank words and sentences by some metric, keeping repetition as low as possible. For a good overview of extractive summarization please refer to Allahyari et al. (2017); Dong (2018).

Let us quickly establish three extractive summarization baselines. The ROUGE score can be easily used to find the optimal extractive baseline that is able to maximize the score. This can only be applied on the training dataset and is not available during test time. This kind of solution is roughly speaking, the cheating solution to show what the maximum possible ROUGE could be. With extractive summarization our primary goal is to rank words, phrases or sentences in the source text given a target summary, copying the most salient information from the source.

Lead-n extracts the first n sentences, normally the first, second and third as extractive summarization. Since most summarization datasets deal with news items the assumption is that the first three sentences provide a good summary of the whole text. It is important when introducing new datasets to test this method since it provides a way to quantify how hard it will be to extract the content for a target summary. Moreover it gives a notion of how the information is spread across the whole source text. Low scores indicate that information is spread across the whole source text, very high scores indicate that most of the information is present in the first three sentences.

Oracle-n extracts n sentences for any source where n is strictly smaller than the maximum number of sentences. The ranking is as follows, from all the sentences pick the sentence with the highest ROUGE 1 to L score. Repeat until threshold was reached. On the way ideas like coverage, e.g. highly overlapping content to eliminate sentences that

are too close to an already added one, are used to minimize repetition. This procedure is very helpful since it gives you the maximum ROUGE score you could possibly achieve using extractive summarization. This in turn also sets the absolute value you can possibly achieve. Low scores indicate that the source and target are lexically disconnected from each other, while high scores indicate that both share a lot of words.

TextRank, Radim and Sojka (2010), is an unsupervised graph based algorithm where sentences or phrases are picked based on their relation. The weighting can have many different forms. Commonly sentences are mapped to a numeric representation making them comparable with distances like the cosine distance. The result is a dense matrix or low rank approximation of the a highly sparse matrix. The dense matrix is a ranking of sentences and phrases. Note it is also possible to initialize a sentence with a word embedding procedure and then create a cosine similarity matrix based on that.

The above procedures are not based on neural algorithms. The Oracle-n score is most informative in terms of performance. TextRank and Lead-n are simply heuristics resembling classically symbolic systems. Neural architectures on the other hand learn a compression, by either deleting words from the source until it is most similar to the target, or by picking the most probable words from the source that could possibly generate the target. Later in this chapter we will describe two state-of-the-art neural extractive summarization models by Gehrmann et al. (2018); Liu (2019) using ELMo and BERT.

4.1 DPA Dataset

The Deutsche Presse Agentur (DPA) provided us with roughly 1.2 million texts with various degrees of metadata. The datasets involved for summarization however are much smaller. There are roughly 100.000 texts that contain a description and 380.000 texts that have a headline long enough to work with. We received two DPA datasets, the news ml g2 dataset containing 214.504 articles as XML and the weblines dataset containing 1.001.719 as JSON articles. In total both contain 22 relevant metadata fields of which we use 7. Both contain the same information though with different value proportions.

The news ml dataset only contains 165.000 valid headlines and 35.697 descriptions, the latter amounting to 18.46% of the data. The categories and genres are important for preprocessing.

No.	Field	Unique	Not Nulls	Perc. not null
1	guid	213965	213965	100.00%
2	category_qcode	6	214725	100.00%
3	genre_qcode	30	214725	100.00%
4	text	186387	214504	99.90%
5	headlines	165107	214725	100.00%
6	description	35697	39635	18.46%

Table 4.1: News ML dataset

No.	Field	Unique	Not Nulls	Perc. not null
1	guid	1001719	1001719	100.00%
2	category_qcode	13	1001719	100.00%
3	genre_qcode	35	998318	99.66%
4	text	985551	1001719	100.00%
5	headlines	906274	1001719	100.00%
6	description	103786	110034	10.98%

Table 4.2: Weblines dataset

The weblines dataset on the other hand contains 906.274 valid headlines and 103.786 descriptions amounting in the latter case to 10.98% of the data. The dataset was concatenated into one large corpus.

4.2 Evaluation

Evaluating a summarization system is very delicate and ambiguous. On the one hand there are standardized scores like the ROUGE metric that measures for precision and recall of overlapping words from the hypothesis and the gold summary. On the other hand there is intrinsically testing a system by asking experts or doing blind comparisons. Both systems have problems, namely the standardized scores do not necessarily test for a good summarization and experts are mostly not available. Mechanical turks are biased and do not know what to look out for. Therefore it is necessary to briefly introduce the quantitative metrics, explaining the ROUGE score and BERT score.

4.2.1 ROUGE

The ROUGE metric, introduced by Lin (2004), is a recall based metric that measures the overlap of n-grams between the gold G (G' for unique n-grams) and hypothesis summary H (H' for unique n-grams). It is standard practice to compare n-grams of one, two and longest, namely, ROUGE-1, ROUGE-2 and ROUGE-L for longest overlapping sequence. The ROUGE-1 score, which can be generilized to any kind of n-gram overlap, can be defined as follows

$$\mathrm{P1}(H', G') = \frac{1}{|G'|} \sum_{i=0}^{|H'|} f(H'_i, G') \tag{4.2}$$

$$f(h, G') = \begin{cases} 1, & \text{if } h \in G' \\ 0, & \text{otherwise} \end{cases} \tag{4.3}$$

In other words, count the number of unique words that are in the hypothesis and in the gold summary at the same time and divide it by the total unique word count of G'. Recall is defined exactly the other way around. The F-score is defined as

$$2 \times \frac{P \times R}{P + R} \tag{4.4}$$

the harmonic mean of precision and recall. To consider a summarization system good it needs to have high scores on all three F-measures, R-1, R-2 and R-L, resulting in a high R-Avg. Simply put, the system with the most overlaps in both directions and with the longest common sub-sequences. Obviously this metric has several flaws in reality. Some of them are listed in this list.

1. It is questionable whether a good summary is measurable by counting overlapping n-grams. Fluency, correctness, coverage are all metrics that are not tested but very important in summaries.

2. The length of the hypothesis summary is really important. The longer the hypothesis is the more words can overlap with the reference. This can lead to very bad results if the summaries are very short. In reality the summaries are always chosen in a way that it needs to be fixed at a certain k or it is hard to compare. For this it would be good to have a length constrained penalty in the ROUGE score.

3. Different words can have the same meaning and the same words can have a different meaning. The ROUGE score does not account for contextual information nor for two different words with almost the same meaning.

4. ROUGE favours highly extractive datasets. That means if the overlap (single words, n-grams, entire sentences) between the original text and its reference is high, algorithms will simply learn to copy relevant passages. It becomes much easier to achieve good scores. What about very abstract references that describe the text on a different level?

To this end, there are a few considerations. The length constraint is easily solved since automatic summarizations have a fixed length of 100 words. This assumption obviously breaks down if the texts are much longer or much shorter. Secondly, making use of language models like BERT for dealing with word-sense disambiguation has proven to be fruitful, effectively dealing with different word combinations and extractive datasets. Extractive datasets are favoured since testing for n-grams, happens on a symbolic level. The English ROUGE score relies on WordNet to find overlapping lemmatization of a word, which eliminates some of the conceptual problems. However WordNet is an English database. Fluency and correctness can only be attested by experts, for there is no quantitative measure if a summarization is fluent.

4.2.2 BERT Score

The BERT score is a rather new invention. In a nutshell: embed the reference and candidate summary with BERT and do a cosine distance metric over the resulting word representations.

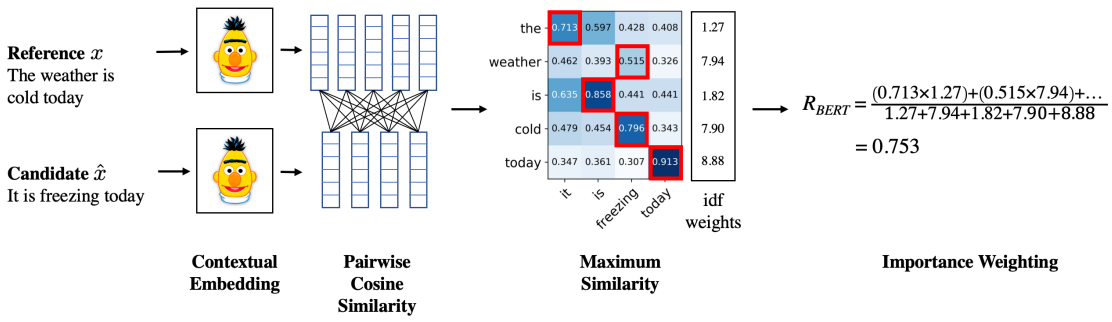


Figure 4.1: BERT score Zhang et al. (2019b)

This system deals with several things at once. First, long candidate sequences are penalized since the cosine similarity will be penalized for each additional word that is not part of the reference. Second, it deals with word-sense disambiguation: *freezing today* and *the weather is cold today* are pretty close in meaning, but would normally only have one overlapping word, resulting in a very bad ROUGE score. It could be problematic to use BERT in a downstream model since it is biased towards BERT. This idea is not restricted to BERT but to all kind of embeddings and language models to disambiguate words.

4.3 Abstractive Summarization

In abstractive summarization the question is on how to compress the original text and instead of copying the relevant parts generating new language. Language generation in and of itself is an almost impossible task to achieve. Above all, syntactical and factual correctness are very hard to generate. In addition to learning to generate coherent sentences, the system needs to learn the compression of a longer into a shorter sequence.

Neural text summarization is greatly inspired by neural machine translation systems and had its kick off with the first neural summarization system introduced by Rush et al. (2015). They managed to align two sentences where the source is longer than the target producing fluent summaries. Quick iterations followed by Chopra et al. (2016) extending the LSTM architecture by providing a new CNN attention mechanism. Nallapati et al. (2016) were able to generate longer sequences, via a pointer mechanism and different input representations that focus on morphological and co-occurrence statistics of language. Wiseman and Rush (2016) introduced beam search optimization during training and inference to bridge the disconnect between the actual task of sequence aligning and summarization. The introduction of beam search during training constraints the model to learn global sequence scores instead of many local ones. The pointer generator model by See et al. (2017) is nowadays considered the baseline model for abstractive summarization. They introduced an end to end trainable pointer mechanism with a coverage penalty, achieving a new state-of-the-art. In the same year Gehring et al. (2017) established the first sequence-to-sequence architecture fully trained with CNNs.

After this the progress was redirected to models with greatly increasing complexity based on reinforcement learning objectives. Paulus et al. (2018) created a model that enhances the coverage penalty by an intra-decoder attention mechanism allowing to penalizes words

already in the decoder sequence. Additionally, Paulus et al. (2018) used a mixed training objective function consisting of a non-differentiable reward function with a self-critical policy gradient algorithm maximizing the ROUGE metric directly. Zhou (2018) proposed an extractive model that jointly learns to score and infer the best sentences by building up a hierarchical representation of text. Celikyilmaz et al. (2018) proposed their deep communicating agents model, which is a multi-agent decoder-encoder where each agent learns to summarize paragraphs of the source.

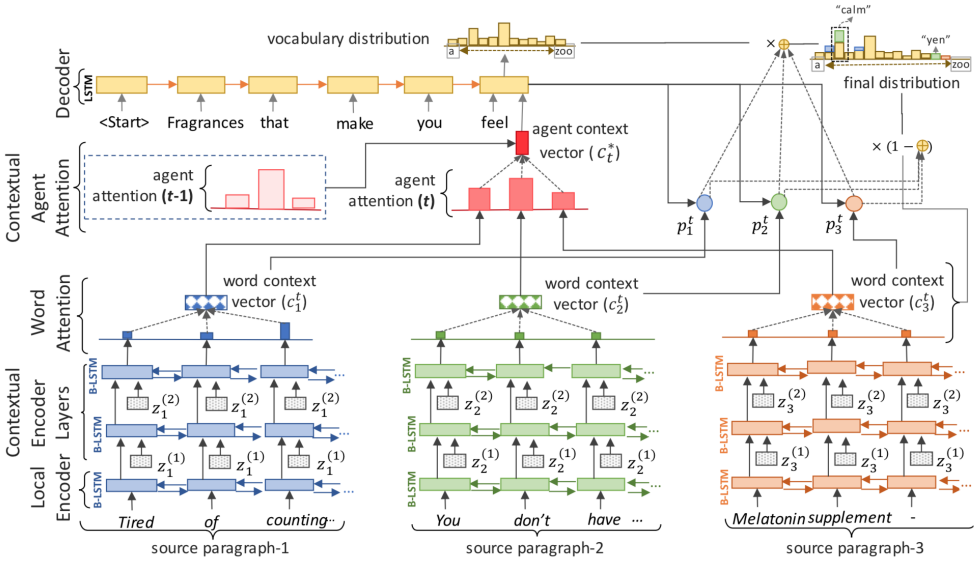


Figure 4.2: DCA by Celikyilmaz et al. (2018)

As depicted in 4.2 the word level attention probabilities of each paragraph are combined with an agent attention mechanism. The importance of locality is in strong focus, since every paragraph is important and treated locally. They add an additional semantic cohesion loss to reduce repetition and directly learn to learn the ROUGE metric with a reinforcement learning policy gradient method. This model is the state-of-the-art model.

Since reinforcement learning objectives are extremely hard to train correctly and the rise of ELMo and BERT created new possibilities, Gehrmann et al. (2018) introduced an abstractive summarization system based on a copy transformer and an ELMo extractor used during inference time, achieving a strong baseline comparable to Celikyilmaz et al. (2018). Since fine-tuning language models to specific tasks is now in reach, Liu (2019) used the BERT model to create a state-of-the-art extractive summarization system by

fine-tuning a linear classifier for sentence selection.

Our work will directly combine Bottom-Up summarization by Gehrmann et al. (2018) and the extractive summarization by Liu (2019) using BERT. As a side note, two new unpublished models came about in the last months. The focus is to extensively use ELMo or BERT in combination with pre-training and fine-tuning. Zhang et al. (2019a) successfully trained BERT with a reinforcement learning objective as a abstractive summarizer and Edunov et al. (2019) successfully fine-tuned ELMo, resulting in a strong new abstractive summarizer. While the models are not yet officially published they are included in the list.

4.4 Bottom-Up Summarization

Bottom-Up summarization by Gehrmann et al. (2018) is an ensemble algorithm with three different results.

As depicted in figure 4.3 the three results are, first, establishing an abstractive summarizer using a copy transformer, second, estbalishing an extractive summarizer using ELMo and third combining the extraction mask during test time in the copy transformer.

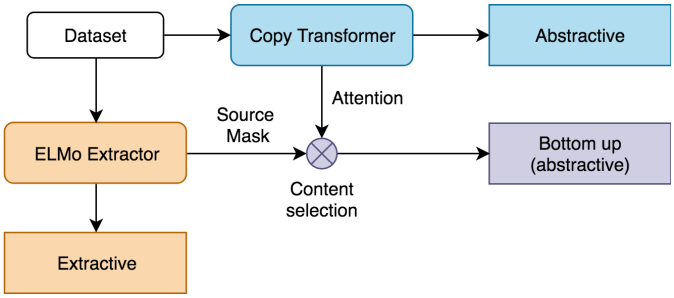


Figure 4.3: Bottom-Up summarization adapted from Gehrmann et al. (2018)

Two models are at work here, the abstractive copy transformer and the extractive ELMo model. The transformer architecture was introduced in chapter 2 the ELMo model in chapter 3. To make clear what components are involved and how they are applied for the summarization task there is a small overview of the relevant mechanisms like copying, coverage and length normalization.

The **ELMo Extractor** is a content selector, tagging parts of the source with words from the target. That is, each word in the source receives a 1 if it is in the target and a 0 if not. This is illustrated well in this figure:

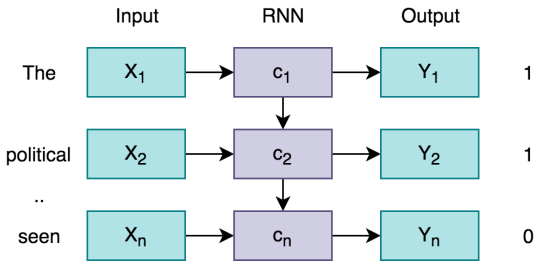


Figure 4.4: Many to many Sequence tagging

The ELMo extractor follows several ideas introduced by Collobert et al. (2011); Kim et al. (2015); Peters et al. (2017), namely a bidirectional character LSTM in a sequence-to-sequence learning setting with a conditional random field for label prediction using the viterbi algorithm for best sequence retrieval. Furthermore, the weights of the model are initialized using GloVe as a word embedder by Pennington et al. (2014) and ELMo as a character/word embedder. Both extractions are concatenated, resulting in a word dimensionality of $\mathbb{R}^{d_{glove}+d_{elmo}}$. Peters et al. (2017) found it beneficial to use a pre-trained language model in a named entity recognition task.

The **Copy Transformer** is a transformer encoder-decoder model based on Vaswani et al. (2017) with some enhancements like copying, length normalization, coverage and trigram blocking. Apart from it being a transformer model and thus not being a sequential model, it behaves exactly the same as the LSTM.

4.4.1 Copying and Coverage

Before See et al. (2017) established a benchmark with the pointer generator network for abstractive summarization, sequence-to-sequence models could not align long sources with target sequences. Additionally, summaries lacked coverage of the source and were prone to repetition. The source and target sequences could not be longer than a single sentence. See et al. (2017) changed this by introducing a one layer encoder-decoder LSTM network with two attention mechanisms and a coverage penalty. The first attention mechanism was the standard global attention by Bahdanau et al. (2014) aligning the encoder with the decoder. The second attention mechanism learns to distinguish between

when to copy a word directly from the source and when to generate one as depicted in figure 4.5.

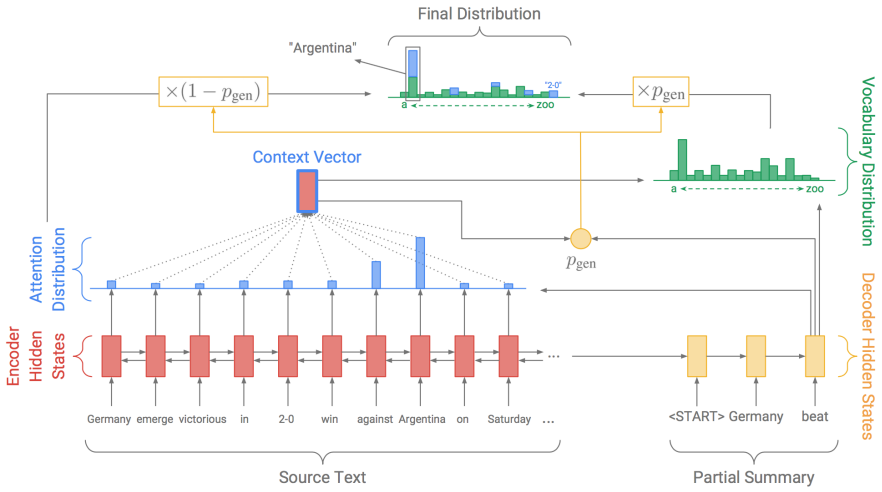


Figure 4.5: Pointer generator and coverage See et al. (2017)

Below there is a description of the core ideas presented in the pointer generator. The novelty about their approach was using two attention mechanisms for the standard sequence-to-sequence attention and an additional copy attention mechanism. It is possible to combine both attention mechanisms into one.

Copying is defined via a pointer attention mechanism that jointly learns when to copy and when to generate based on the source word attention distribution. The pointer attention is defined as

$$p_{gen}(x_t) = \sigma(w_h^T h_t + w_s^T s_t + w_x^T x_t + b_{ptr}) \tag{4.5}$$

$$p_{copy}(x_t, y_t) = p_{gen}(x_t)p(y_t) + (1 - p_{gen}(x_t)) \sum_{i:y_i=y_t} a_i^t \tag{4.6}$$

where h_t is the context vector and s_t the decoder state with the encoder input x_t and decoder input y_t at timestep t. All w are learnable parameters. This is essentially a latent switch $1 - p_{gen}$ which learns to either generate the next word according to the softmax probabilities $p(y_t)$ or copy a word given the weighted sum of the input word x_t attention probabilities a_i^t.

Coverage on the other hand is enforced with a additional coverage vector c_t

$$c_t = \sum_{i}^{t-1} a^i \tag{4.7}$$

that is the sum of all the attention distributions of the decoder until timestep $t-1$. It gives a magnitude for each source word. The higher the magnitude, the less likely the word will be selected. Besides, a step-wise penalty is needed to apply it every time a decoding step happens:

$$\text{covloss} = \sum_{i} \min(a_i^t, c_i^t) \tag{4.8}$$

The context vector c_t is used during the global attention from Bahdanau et al. (2014) at each timestep.

4.4.2 Beam Search

Beam search is a best-first search based on heuristics keeping the best n paths based on a parameter β and thus is a greedy algorithm. Beam search in sequence-to-sequence learning problems essentially uses the model decoder output log probabilities for the next word and adds it to a so called beam (candidate list). Sequence-to-sequence models involve decoding at each time-step t. Evaluating the next possible words, keeping only the best words given a threshold β. Every time the end-of-sequence symbol is selected, the beam is reduced by one until it is zero. When it reaches zero it picks the highest scoring sequence given the highest log probability of individual words.

Wiseman and Rush (2016) suggested that it is possible to enhance the standard encoder-decoder loss to incorporate beam search during training. It becomes then possible to apply length constraints and coverage during inference time. Let us derive some formalism's about beam search and then quickly connect its ideas to the sequence-to-sequence learning domain. The general normalization penalty can be formulated like this

$$s(Y, X) = \frac{\log P(Y|X)}{lp(Y)} + cp(X, Y) \tag{4.9}$$

where lp is the length penalty, cp the coverage penalty. The length penalty can be defined as

$$lp(Y) = \frac{(5 + |Y|)^\alpha}{(5 + 1)^\alpha} \tag{4.10}$$

where α is some threshold parameter. It penalizes the model to learn longer sequences the higher α goes. The coverage constraint can be seen as a minimization problem of the next word probabilities of every word in the source X to the target sequence Y

$$cp(X, Y) = \beta \sum_{i=1}^{|X|} \log(\min(\sum_{j=1}^{|Y|} p_{i,j}, 1.0)) \tag{4.11}$$

where β is a hyperparameter that when increased forces to cover more of the source X. Moreover, to this it is suggested in Paulus et al. (2018) that one of the core problems are repeated trigrams. That is three consecutive words are repeated endlessly. By simply constraining the beam search to refuse trigrams that are already present in the current path helps in reducing repetition. In cases where words are out of vocabulary, rendering an <unk> token, copying the best next word from the source X is a successful strategy.

4.5 BertSum for Extractive Summarization

Fine-tuning BERT for extractive summarization by Liu (2019) is a new extractive state-of-the-art algorithm using BERT to select the most salient sentences by learning the oracle selection. This is basically done in two steps. First, pre-train the masked BERT language model and second, fine-tune the domain dataset on BERT with a linear sentence classifier at the end. Liu (2019) enhanced the sentence tagging to be able to distinguish an entire document from a sentence. As explained before, BERT cannot traditionally predict next word probabilities, instead it can predict the next sentence given sentence pairs. Sentence pairs, however, can be entire documents. Recall from figure 4.6 that BERT is a combination of an input document delimited by special tokens embeded into a token, segment and positional embedding.

In order to distinguish between sentences within a document, Liu (2019) labels each sentence based on the sentence number with A when it is even and B when it is odd. The Oracle selection picks the most salient three sentences given the ROUGE score and

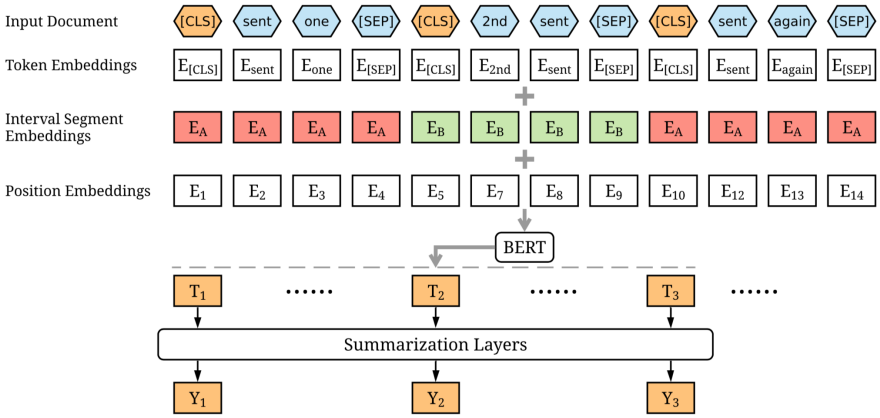


Figure 4.6: BertSum by Liu (2019)

labels each sentence with 0 or 1. Additionally, each sentence is separated by a $[CLS]$ token to mark a sentence having a definite ending. Recall that BERT cannot provide word probabilities, due to the cloze task. To go around this a linear classifier is stacked on top to get unnormalized sentence scores. Liu (2019) used two additional transformer layers for extracting sentence representations.

$$\hat{h}_l = LN(h_{l-1} + MHAtt(h_{l-1}) \tag{4.12}$$

$$h_l = LN(\hat{h}_l + W \cdot \hat{h}_l \cdot b)) \tag{4.13}$$

where LN is layer normalization and $MHAtt$ a multi head attention mechanism, followed by a simple feed forward network. On top of the intermediate representation and layer normalization a linear layer is stacked to get a per sentence score

$$\hat{Y}_i = \sigma(W_o h_i^L + b_o) \tag{4.14}$$

where L is the amount of additional transformer layers. Surprisingly, this beats all extractive summarization systems by up to 1.5 ROUGE points.

4.6 Bottom-Up BERT

Bottom-Up BERT is an esemble between Bottom-Up summarization Gehrmann et al. (2018) and a BERT extractor by Liu (2019). Recall that Gehrmann et al. (2018) used

ELMo as a feature extractor by providing word probabilities. The feature extractor is trained with sequence tagging, where the labels are the target text's words that are present in the source text. Via beam search the most probable path is selected through the sentence probabilities. Using BERT as a sentence ranker instead of ELMo, we push the performance of the Bottom-Up model. In the first phase the DPA data is fine-tuned to the BERT language model with a linear sentence predictor on top. The resulting selected sentence IDs, given by a single index are used to mask the source document with 1 when the word occurs in that sentence and 0 if not. The following formulation makes this clear

$$X \in \mathbb{N}^{d_{source}} \tag{4.15}$$

$$X_{sent} \in \mathbb{N}^{d_{sents}} \tag{4.16}$$

$$X_{selected} = BertSum(k, X, X_{sent}) \tag{4.17}$$

$$mask^{d_{source}} = \forall x \in X \begin{cases} 1 & \text{if } X_{sent}(x) \in X_{selected} \\ 0 & \text{else} \end{cases} \tag{4.18}$$

where X again is the source sequence and X_{sent} is a mapping or function that maps each word of X to its sentence index. Selecting the top k IDs with BertSum and creating the mask by setting all words to 1, when they are part of a selected sentence and 0 if not. This mask has the same dimensionality as the source text since during the beam search inference, it needs to align with the attention weights of the copy transformer. We also tried a soft variant, where instead we set each value to a probability relative to the score with a softmax and boosting factor, which did not improve the results.

4.7 Discussion

In this section we have learned that text summarization is split into extractive summarization - based on ranking words and sentences - and abstractive summarization - based on language generation methods with sequence-to-sequence models. Transfer learning greatly helps in improving extractive summarization methods and in turn can be used as a sentence selector for abstractive models. The ROUGE evaluation metric measures the quality of a summary given a gold summary. Recall from earlier chapters that this is a problem since the sequence-to-sequence framework models sequence alignment with a next word prediction task. With beam search during inference time there are possibilities

to use the copy weights, reduce repetition and constrain the summary to have a certain length. Using the next word prediction probabilities during beam search, helps in selecting additional salient information. However, the models are not end-to-end differentiable which doubles training time.

The ideas of BertSum by Liu (2019) and Bottom-Up summarization by Gehrmann et al. (2018) can be combined using the same masking scheme. This implicitly compares an extractive ELMo model with a fine-tuned BERT as suggested by Peters et al. (2019). Besides this, Edunov et al. (2019) have shown that it is possible to successfully use ELMo with fine-tuning for abstractive summarization. Zhang et al. (2019a) on the other hand have shown that BERT is too large to be fine-tuned effectively for abstractive summarization. Instead, they introduced reinforcement learning and also created an extractive ranking along the way. In the future we will see more models using language models of all kinds and achieving better results on the way.

We introduced the DPA dataset and the relevant statistics involved for summarization. The descriptions are longer with two to three sentences, while the headlines are relatively short with one to two sentences. Later The upcoming experiment section reveals that both datasets are not extractive since the overlaps of words from source text are low compared to the words appearing in the target text. It is therefore difficult to use copy mechanisms and extractive techniques on this dataset.

5 Experiments

In this section we will provide a detailed description of the summarization experiments, results and the required analysis to interpret them. In order, we will first compare Bottom-up BERT to other state-of-the-art models on the English CNN/DM corpus. Then we will proceed to describe all the results on the description and headlines dataset. Instead of a single model we use several different architectures like transformers, LSTMs, CNNs in an extractive as well as abstractive setting. Since there has been no prior work on the German DPA dataset, we added a section on the preprocessing. Since the length of the source text is important for summarization we tested shortening to 80 and 400 tokens. The description benefits from longer sources and the headlines benefit from shorter sources. As introduced in the last chapter we will use the ROUGE evaluation metric and the BertScore. Our model achieves a new state-of-the-art on the ROUGE-1 score, while the rest of the scores align with other top scores. Bottom-Up summarization and BertSum are both models building on two popular language models, ELMo and BERT. The comparison on various tasks between these too is especially interesting. They are however not directly comparable, since BERT is not able to do a next word prediction task. Instead both models are compared using proxy tasks like summarization. The experiments were conducted earlier this year and since then two new models by Edunov et al. (2019); Zhang et al. (2019a) appeared, using ELMo and BERT in a fine-tuning setting. Both models are strong contestants for new baselines. Kim et al. (2018) introduced a new reddit posts dataset along with a model that handles abstractive texts much better. Hoang et al. (2019) also studied fine-tuning on pretrained transformers in context of summarization and found that the length of the summary played an important role for the ROUGE metric and achieved better results on abstractive datasets.

5.1 Bottom-Up BERT on CNN/Daily Mail Corpus

The CNN/Daily Mail dataset contains roughly 290.000 documents with a test and validation set of 11.500 articles. The summaries are bullet point lists with sentences describing the salient information of the source text. In table 5.1 we see baselines and extractive metrics. The Oracle score with the best three selected sentences achieve extraordinary high results. On average 10 points higher than any learned baseline. This indicates that the CNN/DM dataset is highly extractive. The first three sentences alone without any learned ranking achieve very good results.

	Model	R-1	R-2	R-L	R-Avg
Baselines	Lead-3 (First three sentences)	40.24	17.70	36.45	31.46
	Oracle (Sentences chosen with ROUGE)	52.59	31.24	48.87	44.23
Extractive	TextRank (Radim and Sojka (2010))	40.20	17.56	36.44	31.40
	NeuSum (Zhou (2018))	41.59	19.01	37.98	32.86
	ELMo (Peters et al. (2018))	42.00	15.90	37.30	31.73
	BertSum (Liu (2019))	**43.23**	**20.22**	**39.60**	**34.35**
Abstractive	Pointer Generator (See et al. (2017))	39.53	17.28	36.38	31.06
	Copy Transformer (Gehrmann et al. (2018))	40.96	18.38	38.16	32.50
	RL, intra-attention (Paulus et al. (2018))	41.16	15.75	**39.08**	32.00
	DCA (Celikyilmaz et al. (2018))	41.69	19.47	37.92	33.03
	Bottom-Up (Gehrmann et al. (2018))	41.22	18.68	38.34	32.75
	Dynamic Convolutions (Wu et al. (2019))	39.84	16.25	36.73	30.94
	BERT + RL (Zhang et al. (2019a))	41.71	**19.49**	38.79	**33.33**
	Fine-tuned ELMo (Edunov et al. (2019))	41.56	18.94	38.47	32.99
	Bottom-Up BERT (Ours)	**41.75**	18.95	38.35	33.02

Table 5.1: Results of CNN/DM on Several Models.

Our Bottom-Up BERT ensemble model uses the abstractive copy transformer by Gehrmann et al. (2018), using the masking of BertSum Liu (2019), achieving a new state-of-the-art on 41.75 R1 ROUGE score. All three ROUGE scores improve by a margin compared to the Bottom-Up model. However, the overall winner in summarization on the R-Avg is still the DCA model by Celikyilmaz et al. (2018) using reinforcement learning. Our Bottom-Up BERT model however is ranked second compared to DCA by Celikyilmaz et al. (2018) and third when we also consider the BERT + RL model by Zhang et al. (2019a) which was introduced after the experimentation phase.

BertSum was trained with 2 inter transformer layers, 8 transformer heads, a feed forward size of 2048, with the Noam scheme and 10000 warm-up steps, dropout of 0.1 and total train steps of 50000. The maximum number of sentences are summary must have was three, with a maximum limit of 100 and a minimum of 5 tokens, clipped at 400 tokens.

Copy Transformer by Gehrmann et al. (2018) was replicated by following the training options for the original transformer by Vaswani et al. (2017). The embedding and model size has a dimension of 512, using the Adam optimizer with beta2 of 0.998 and label smoothing by 0.1. Warm-up steps of 8000, the Noam decay scheme, 8 transformer heads a learning rate of 2. Additionally copy attention is enabled. The original results as explained earlier also make use of extensive options during inference time with beam search. Subsequently a few examples comparing different summaries by different models from the table

Summary - gary locke has been interim manager since start of february. locke has won two and drawn four of his seven games in charge. the 37-year-old took over when allan johnston quit.
BertSum - the 39-year-old - who will speak at a press conference on friday morning - has lost just once in seven games since taking over at rugby park. kilmarnock interim manager gary locke has been given the role on a permanent basis after signing a three-year deal. the former hearts boss joined the club as assistant boss to allan johnston last summer but took control of the team when his ex-tynecastle team-mate quit at the start of february.
Copy Transformer - locke has been given the job at kilmarnock on a permanent basis. the former hearts boss joined the club as assistant boss last summer. locke has lost just once in seven games since taking over at rugby park.
Bottom-Up - gary locke has been given the role on a permanent basis. the former hearts boss joined the club as assistant boss to allan johnston last summer but took control of the team when his team-mate quit at the start of february. locke has lost just once in seven games at rugby park.
Bottom-Up BERT - gary locke has lost just once in seven games since taking over at rugby park. the former hearts boss took control of the team when his team-mate quit at the start of february. the 39-year-old will speak at a press conference on friday morning.

As we can see there is a lot of copying going on. Analysing Bottom-Up BERT more carefully per sentence

1. Directly from the copy transformer but adding the full name
2. First part is a direct copy in all summaries of the trig-ram "former hearts boss", second part of sentence is equal to Bottom-Up

3. Last sentence is a direct copy of the source (BertSum)

It is obvious that Bottom-Up BERT is a combination of the copy transformer and BertSum, mixing up the order of sentences. Bottom-Up BERT is constrained to three sentences, we could have shortened the summaries. It is questionable whether the last sentence is important or not. In this setting it makes sense that Bottom-Up BERT beats the ELMo version, since BertSum already achieves much higher results. The expectation therefore is that the average ROUGE score will increase compared to the ELMo masking. However only the ROUGE-1 score improved by a good margin, while the rest of the scores are only minimally better.

5.2 DPA Corpus Preprocessing

We did a variety of preprocessing steps for the DPA dataset. Since we dealt with unnormalized data but curated text it was vital to filter out bad examples. For this iteration of the dataset we did excessive filtering on categories, lengths of source and target texts and symbolic word filtering.

1. Keep the categories mixed (dpacat:vm), art (dpacat:ku), sports (dpacat:sp), politics (dpacat:pl)
2. Drop the dpatextgenre:32 (internal infos)
3. Sanitize text keeping words, numbers and sentence delimiters
4. Removing duplicated texts, headlines and descriptions
5. Mininum text length must be 60 words
6. Minimum description length is 12 words and minimum headline length is 6 words
7. Maximum source words limited to 80 and 400
8. Droping all nan values, lowercasing all words
9. Keeping the top 50000 words as vocabulary, replacing the rest with <unk>

It was vital to drop certain categories like economics since they only contain tables. This is an interesting additional task but does not really fit into a sequence to sequence learning objective. The category dpacat:rs only contains duplicates and inner communication of the DPA. Effectively non helpful and mostly without descriptions. The minimum length constraints are very important since we need to try generating real sequences. Too short and we do not really test the capabilities of the involved models. In this step we lost roughly 70% of the original data, since there are a lot of headlines with fewer than 6 words. It was also helpful to drop text containing repeating phrases that are not relevant to the task. Mostly these are texts in the dpacat:rs category. Other than this we constrained the source text to 80 tokens and 400 tokens which the default in most summarization models. Moreover the transformer models involved could not handle more than sequences of 512 words, since we could not train much larger versions. The vocab is capped at 50.000 tokens , since sequence to sequence models with attention have a large number of trainable weights. The zipf distribution of words immediately makes clear that more than 50.000 tokens results in words with a single occurence.

For the tokenization we used spaCy with its German state-of-the-art CNN parser, to chunk text into delimited sentences and white-space separated words. We wrapped $<t>$ and $</t>$ around each sentence to easily join and split sentences for efficient transformations later on.

5.3 Description Summarization

The DPA description dataset consists of a source and target text of a total of 104.942 pairs in the train and 11.661 in the validation dataset. The source is the original text and the target the description with a mean sentence length of 2.75.

There are some anomalies to be noted. The maximum sentence count for descriptions is 28, with a maximum word count of 75. This is likely an issue with the tokenization. On average however the mean description length of 32 and sentence count of 2.75 makes more sense. The maximum number of words in a single text also drastically vary between train and validation set, however only noise in comparison. The standard deviations on the other hand have expected magnitudes when compared to the mean. Also note that the minimum word length is exactly 12, as constrained by our preprocessing.

Split	Kind	Dataset	Mean	Median	Min	Max	Std
Train	Sents	Text	22.92	21.00	2.00	241.00	15.07
104.942		Description	2.75	3.00	1.00	28.00	1.30
	Words	Text	380.64	365.00	62.00	4012.00	233.27
		Description	31.98	32.00	12.00	75.00	7.78
Validation	Sents	Text	22.80	21.00	2.00	181.00	15.03
11.661		Description	2.75	3.00	1.00	26.00	1.37
	Words	Text	379.75	363.00	64.00	2400.00	233.49
		Description	31.95	32.00	13.00	65.00	7.83

Table 5.2: General statistics for DPA description.

5.3.1 Summarization Results

The results on the DPA description dataset are somewhat mixed. Compared to other academic datasets it seems much harder to summarize the descriptions. The descriptions benefit from longer source texts. When the source is capped at 80 tokens, as depicted in table 5.3, the results are worse than when we cap it on 400 source tokens, as depicted in table 5.4.

	Model	R-1	R-2	R-L	R-Avg
Baselines	Lead-1 (First sentence)	21.17	7.05	18.00	15.41
	Lead-2 (First and second sentence)	24.17	7.50	20.96	17.54
	Lead-3 (First three sentences)	24.19	7.38	21.18	17.58
	Oracle (Sentences chosen with ROUGE)	**29.85**	**10.75**	**25.86**	**22.15**
Extractive	TextRank (Radim and Sojka (2010))	16.73	4.59	14.81	12.04
	ELMo (Peters et al. (2018))	23.57	**8.21**	20.24	17.34
	BertSum (Liu (2019))	**24.36**	7.51	**21.16**	**17.67**
Abstractive	Pointer Generator (See et al. (2017))	20.74	5.02	17.78	14.51
	Copy Transformer (Gehrmann et al. (2018))	**22.16**	**7.50**	**20.31**	**16.65**
	Bottom-Up (Gehrmann et al. (2018))	21.96	6.93	19.98	16.29
	Bottom-Up BERT (Ours)	15.78	3.49	13.73	11.0

Table 5.3: Results on DPA Description source length of 80

The baseline section for 80 tokens indicates that extractive techniques cannot be that successful. The highest Oracle ROUGE average is 22.15. Again since there is no compar-

ison, we now have introduced an extractive baseline. In the extractive section, BertSum by Liu (2019) is the winner, followed by ELMo. In the abstractive section the copy transformer by Gehrmann et al. (2018) is the winner. However no model is really prominent here. Bottom-Up BERT cannot reproduce the numbers on the CNN/DM dataset. This is expected since the extractive baseline is also much lower. Moving on to the 400 token version, as depicted in table 5.4, there is a drastic increase in results.

	Model	R-1	R-2	R-L	R-Avg	BERT-F1
Baselines	Lead-1 (First sentence)	20.30	6.61	17.37	14.76	0.596
	Lead-2 (First and second sentence)	24.03	7.37	20.83	17.41	0.609
	Lead-3 (First three sentences)	24.02	7.17	21.06	17.42	0.608
	Oracle (Sentences chosen with ROUGE)	**35.68**	**14.19**	**31.25**	**27.04**	**0.645**
Extractive	TextRank (Radim and Sojka (2010))	20.80	5.07	18.00	14.62	-
	ELMo (Peters et al. (2018))	22.77	6.66	19.94	16.46	0.602
	BertSum (Liu (2019))	**24.34**	**7.39**	**21.38**	**17.70**	**0.610**
Abstractive	Pointer Generator (See et al. (2017))	21.35	6.26	19.15	15.59	0.597
	Copy Transformer (Gehrmann et al. (2018))	**27.34**	**12.27**	**25.35**	**21.65**	**0.624**
	Dynamic Convolutions (Wu et al. (2019))	23.43	7.75	17.32	16.20	0.611
	Bottom-Up (Gehrmann et al. (2018))	25.37	9.75	23.24	19.45	0.616
	Bottom-Up BERT (Ours)	22.56	7.53	20.46	16.85	0.605

Table 5.4: Results on DPA Description source length of 400

For the full versions we provide the BertScore as well. The BertScore follows roughly the ROUGE average score, however the magnitude seems to be different. The copy transformer again is the clear winner, the BertScore however is not that much higher. The relative magnitude between all the methods seem to be capped at roughly 0.6 to 0.65 while the ROUGE scores are much more scattered. Bottom-Up BERT greatly improves upon the first version and is on par with the other models. Still, it seems to be hard to generate abstractive sequences for the description dataset, since no score really took off. This is definitely an indication for further research into highly abstractive non copy-able summaries. Below we present a few examples. The description summaries contain very good examples and very bad examples. First we show a selected list of summaries where the language generation sounds fluent and the text makes sense.

BertSum - der kampf gegen rechts muss aus der mitte der gesellschaft geführt werden, sagte sachsens regierungschef michael kretschmer cdu auf dem friedensfest im ostsächsischen ostritz. zu dem neonazi-festival auf einem privatgelände werden bis zu 1000 rechtsextreme aus ganz deutschland und dem ausland erwartet. dort versammeln sich am wochenende neonazis zu dem festival schild schwert.
ELMo extractor - die eu verschärft zum 1. juli ihr handelsverbot für elfenbein , um den weltweiten kampf gegen wilderer und schmuggler zu stärken. das beschloss die eu - kommission am dienstag.
Copy Transformer - überschwemmungen , flugausfälle , haushalte ohne strom - der hurrikan maria hat erneut schäden angerichtet. naturschützer warnen vor den katastrophalen folgen des hurrikans.
Bottom-Up - us-präsident trump hat einen harten kurs in vietnam gemacht. doch chinas präsident lässt sich nicht beirren.
Dynamic convolutions - nach dem verlust ihrer absoluten mehrheit im bayerischen landtag will die csu koalitionsverhandlungen mit den freien wählern aufnehmen.

The summaries a precise, contain a lot of information, sound fluent and seem grammatically correct. In contrast to this there are a lot of bad examples. This is a comparison of one specific text from a selection of models

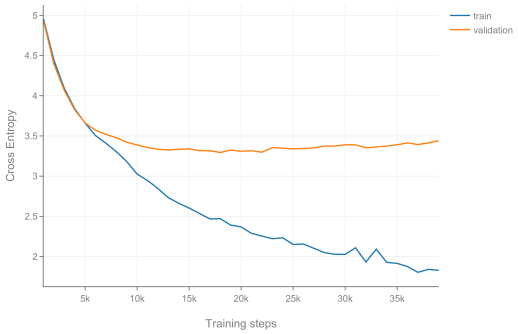
Original - die katalonien-krise spitzt sich weiter zu. im regionalparlament sorgt ein neuer antrag für aufregung.
BertSum - das katalanische regierungsbündnis jxsí und die linksradikale cup haben einen antrag auf konstituierung einer katalanischen republik als unabhängiger staat im parlament vorgelegt. das katalanische parlament soll nun zusammentreten und über den antrag abstimmen. der text spricht nicht von einer erklärung der unabhängigkeit , weil dies bereits der wähler bei dem umstrittenen referendum am 1. oktober getan habe.
Copy Transformer - die katalonien-krise spitzt sich zu. jetzt sitzen ehemalige katalanische ex-minister in untersuchungshaft. die spanische justiz nimmt eine neue phase.
Bottom-Up - die katalonien-krise spitzt sich zu. jetzt sitzen ehemalige katalanische ex-minister in katalonien vor gericht. die spanische justiz nimmt eine neue runde fest.
Bottom-Up BERT - die linksradikale cup wollte im parlament einen antrag auf konstituierung begrenzen. das katalanische regierungsbündnis wurde unabhängiger staat vorgelegt.

For a German reader it becomes immediately clear that there is repetition, imprecise wording, wrong facts through false structure.

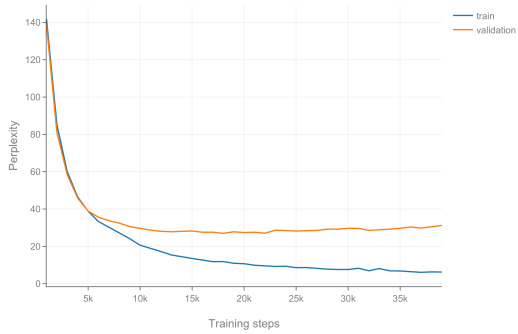
5.3.2 Model Overview and Training

In total we ran two complex extractive models with ELMo and BertSum and three distinct abstractive models with the pointer generator, copy transformer and dynamic convolutions model. Bottom-Up and Bottom-Up BERT are combinations of the extractors and abstractors. We will glance over the training schemes of the copy transformer and dynamic convolutions, as well as the BertSum model. It follows that Bottom-Up BERT is the combination of these two training schemes. Dynamic convolutions on the other hand is a new model entirely build on convolutions and self attention.

Copy Transformer trained with OpenNMT Klein et al. (2017), an open source platform for neural machine translations based on sequence to sequence models. The training regime followed a typical Noam learning rate decay scheme with warm-up steps around 8000. The training steps are in reality training batches. Translating 1 step to 50 batches, resulting in 35000 at step 70. The most important two metrics are the cross-entropy and perplexity score



Cross-entropy



Perplexity

The validation cross-entropy diverges quickly during training. While the training's cross-entropy gets lower with each step. The validation dataset quickly stales after the warm-up steps which implies overfitting to the training data. However recall that the ROUGE metric we are trying to optimize is not directly optimized during training since it is non-differentiable. During training we simply learn a next word prediction task. As the model trains further the ROUGE score actually improves. A model at step 10.000 is less capable than one at step 35.000. The language gets more fluent with each training step, although the validation set is not improving. After a while we can see effects of label

smoothing, e.g. the model gets more unsure about the choice which results in a little worse results. We could try to use a cosine scheduling scheme to increase the validation accuracy as well. The perplexity on the other hand seems to be much closer with the same effects as the cross-entropy. Since the scale is different and non-scaled the gap seems much closer when in fact it is not.

ELMo Extractor was trained with AllenNLP by Gardner et al. (2017). The ELMo model introduced by Peters et al. (2018) was in cooperation with the Allen Institute of AI. We used a German version of Wikipedia trained with ELMo. Since it costs too much time and resources to train it from scratch we had to rely on existing models. We use GloVe and ELMo as token embedders and thus do not fine-tune on it. We concatenate both vectors, resulting in 1024 dimensions from ELMo and 300 from GloVe with a total of 1324 features per word. We train on a simple sequence tagger where words of the summary are used as tags on the source dataset. For training we use the ELMo model with a two layer LSTM with 256 dimensions and highway layers. Gradient clipping in combination with the adagrad optimizer. The final training accuracy was 0.89 with a loss of 0.25 and a validation accuracy of 0.87 and a loss of 0.34.

BertSum was trained with the code provided by Liu (2019). We did not change much of the hyperparameters. For BERT we used the multi-language and German huggingface implementation since the training would take weeks on a vast amount of resources. The model has a vocab size of 30000 with a 12 attention heads and 12 hidden layers. Each hidden layer with a size of 768 and an intermediate layer size of 3072. The hidden activation function is the gelu. We train for 30000 steps, using the Noam decay with 10000 warm-up steps achieving a cross-entropy of 1.45 on the training set and 3.96 on the validation set.

Dynamic Convolutions was trained with Fairseq by Ott et al. (2019) a sequence-to-sequence learning framework. We first apply BPE with 40.000 subword units. Training with a cosine learning rate and the Adam optimizer. The loss of label smoothed cross-entropy. The warm-up steps are set to 10000. We use 8 attention heads with an embedding dimensionality of 512. We apply the dynamic self attentional convolutions. During generation we use a beam size of 8, with a length penalty of 0.9 with trig-ram blocking. The final training loss was 7.5 with a validation loss of 8.1.

5.4 Headlines Summarization

The DPA headlines dataset consists of a source and target text of a total of 187.884 pairs in the train and 11.444 in the validation dataset. The source is the original text and the target the headline with a mean sentence length of 2.34.

Split	Kind	Dataset	Mean	Median	Min	Max	Std
Train	Sents	Text	15.17	10.00	1.00	351.00	15.51
187.884		Headlines	2.34	2.00	1.00	19.00	1.07
	Words	Text	236.02	160.00	62.00	6012.00	224.81
		Headlines	13.13	12.00	7.00	59.00	3.87
Validation	Sents	Text	15.18	10.00	2.00	225.00	14.81
11.444		Headlines	2.27	2.00	1.00	12.00	1.08
	Words	Text	238.66	163.00	63.00	2982.00	217.67
		Headlines	12.98	12.00	7.00	36.00	3.83

Table 5.5: General statistics for DPA headlines.

There are some anomalies to be noted. The maximum sentence count for headlines is 19, with a maximum word count of 59. This is likely an issue with the tokenization again. On average however the mean headlines length of 13 and sentence count of 1 makes more sense. The standard deviations on the other hand have expected magnitudes when compared to the mean. Also note that the minimum word length is 7, as constrained to 6 by our preprocessing.

5.4.1 Summarization Results

The results on the DPA headlines dataset are much more promising compared to the description dataset. Compared to other academic datasets it seems to be easier to generate summary sentences with high ROUGE scores. Multi sentence summaries are probably much harder to generate. The headlines do not benefit from longer source texts. The opposite is the case, when the source text is constrained to 80, the ROUGE scores drastically improves. Compressing 400 tokens to a single sentence, seems to be a harder problem than compressing 80 tokens to a single sentence. In table 5.6 are the results of the headline summarization.

	Model	R-1	R-2	R-L	R-Avg
Baselines	Lead-1 (First sentence)	31.59	12.50	27.17	23.75
	Lead-2 (First and second sentence)	22.24	8.15	19.47	16.62
	Lead-3 (First three sentences)	18.67	6.66	16.52	13.95
	Oracle (Sentences chosen with ROUGE)	**34.43**	**14.10**	**29.81**	**26.11**
Extractive	TextRank (Radim and Sojka (2010))	12.49	3.85	10.80	9.05
	ELMo (Peters et al. (2018))	29.93	11.70	25.91	22.51
	BertSum (Liu (2019))	**32.19**	**13.2**	**27.89**	**24.43**
Abstractive	Pointer Generator (See et al. (2017))	35.02	17.49	32.93	28.48
	Copy Transformer (Gehrmann et al. (2018))	**51.34**	**35.22**	**49.40**	**45.32**
	Bottom-Up (Gehrmann et al. (2018))	39.49	22.71	37.48	33.23
	Bottom-Up BERT (Ours)	43.53	26.54	40.69	36.91

Table 5.6: Results on DPA Headlines source length of 80

The clear winner on the 80 dataset is the abstractive copy transformer model with an average ROUGE score of 45.32. BertSum on the other hand is the winner in the extractive model section with a ROUGE score of 24.43. ELMo is second with 22.51. This also shows in the abstractive category where Bottom-Up BERT achieves on average 3.68 higher ROUGE scores compared to the Bottom-Up model. It is quite interesting to see that BertSum could almost perfectly learn the Oracle procedure with 26.11 to 24.43.

	Model	R-1	R-2	R-L	R-Avg	BERT-F1
Baselines	Lead-1 (First sentence)	20.69	8.49	18.39	15.86	0.577
	Lead-2 (First and second sentence)	20.15	8.37	18.12	15.55	0.581
	Lead-3 (First three sentences)	17.62	7.02	15.97	13.54	0.572
	Oracle (Sentences chosen with ROUGE)	**32.69**	**16.01**	**29.47**	**26.06**	**0.620**
Extractive	TextRank (Radim and Sojka (2010))	13.63	4.06	12.01	9.90	-
	ELMo (Peters et al. (2018))	**18.01**	**6.98**	**16.04**	**13.68**	**0.574**
	BertSum (Liu (2019))	16.13	5.69	14.51	12.11	0.567
Abstractive	Pointer Generator (See et al. (2017))	40.37	24.87	39.02	34.75	0.673
	Copy Transformer (Gehrmann et al. (2018))	42.42	26.81	40.79	36.67	0.686
	Dynamic Convolutions (Wu et al. (2019))	**45.56**	**30.00**	**43.84**	**39.80**	**0.706**
	Bottom-Up (Gehrmann et al. (2018))	40.81	25.08	39.25	35.05	0.678
	Bottom-Up BERT (Ours)	39.36	23.36	37.79	33.50	0.667

Table 5.7: Results on DPA Headlines source length of 400

The clear winner on the 400 dataset is the abstractive Dynamic Convolutions model with

an average of 39.80 ROUGE and a BertScore of 0.706. In the extractive section the ELMo model outperforms with 13.68 ROUGE on average. Compared to the 80 dataset, the ROUGE score drastically decreases for extractive methods whereas the Oracle score stays about the same. This is a rather surprising result and needs some additional attention. It seems to be harder to map the 400 token source dataset to a shorter representation. Recall that the description increases in all ROUGE scores with the longer version, which makes sense, since the description has longer target sequences. Unsurprisingly, in this instance the Bottom-Up model, with 35.05 ROUGE, beats the Bottom-Up BERT model, with 33.50 ROUGE. Bottom-Up builds on the assumption that an extractive masking helps improving abstractive transformers. Since ELMo achieved a higher ROUGE score than BERT it makes perfectly sense that Bottom-Up beats Bottom-Up BERT. Note however this section is not directly comparable to the headlines 80 dataset, since we used a different BERT language model that was specifically trained on German. The results could potentially be better with the multi language version of BERT. Also note, while some results drastically decrease, e.g. the copy transformer from 45.32 to 36.67, models like the pointer generator improve from 28.48 to 34.75 ROUGE. A few examples of the same summary

Original - protest in katalonien - separatisten rufen zu sturm auf banken auf.
BertSum - im konflikt um die unabhängigkeitsbestrebungen in katalonien haben separatistische organisationen die bürger der spanischen region zu einem sturm auf die banken aufgerufen.
ELMo - im konflikt um die unabhängigkeitsbestrebungen in katalonien haben separatistische organisationen die bürger der spanischen region zu einem sturm auf die banken aufgerufen.
Copy Transformer - katalonien ruft bürger zum kampf um banken auf sturm.
Dynamic Convolutions - katalonien ruft zu sturm auf katalonien an katalonien ruft an x.
Bottom-Up BERT - konflikt um katalonien - bürger der region zu sturm auf banken.

Generating summaries using headlines is a good idea when using extractive systems. Extractive algorithms focus on the major points and sentences that highly correlate with the headline. Abstractive systems on the other hand are lacking fluency, correctness, repetitions and non-sense.

Some cherry-picked examples that do not lack fluency, contain repetition and correct in its content.

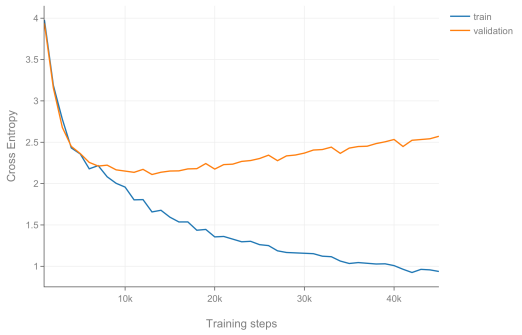
BertSum - die vereinten nationen haben angriffe der syrischen regierung scharf verurteilt, die zum abbruch einer dringend benötigten hilfslieferung im syrischen ostghuta führten. die gewalt offenbare einen mangel an respekt für die waffenruhe und die missachtung von sicherheitsgarantien für den konvoi, hieß es am dienstag in einer mitteilung des un-nothilfebüros ocha. der syrische verbündete russland versprach den rebellen unterdessen freies geleit.
ELMo - weltweit haben 815 millionen menschen im vergangenen jahr hunger gelitten. damit waren 38 millionen menschen mehr von hunger betroffen als 2015, wie die landwirtschaftsorganisation der vereinten nationen fao am freitag in rom mitteilte.
Copy Transformer (400) - zwei angriffe mit stichwaffen auf sicherheitskräfte in brüssel und london.
Copy Transformer (80) - 125.000 euro für integrationsprojekte im landkreis bautzen.
Dynamic Convolutions - un-sicherheitsrat über die lage in gaza. tote bei gaza-grenze.
Bottom-Up - mindestens zwölf tote bei explosion von autobombe in damaskus getötet.
Bottom-Up BERT - fritteuse verursacht brand mit drei toten in echzell.

5.4.2 Model Overview and Training

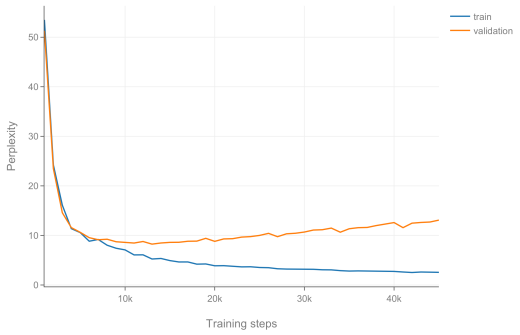
For an underpay description of the different training schemes, we refer to the description section. We mostly provide the same statistics and models with different values to the specific dataset.

Copy Transformer trained with OpenNMT Klein et al. (2017). The training regime followed a typical Noam learning rate decay scheme with warm-up steps around 8000. The training steps are in batches of 50. The most important two metrics are the cross-entropy and perplexity score

The validation cross-entropy diverges quickly during training. While the training's cross-entropy gets lower with each step. The validation dataset quickly stales after the warm-up steps which we also described happening on the descriptions. As long as the training accuracy improves the model receives improved ROUGE scores. A model at step 10.000 is less capable than one at step 50.000. The language gets more fluent with each training step, although the validation set is not improving. After a while we can see effects of label smoothing, e.g. the model gets more unsure about the choice which results in a little worse results.



Cross-entropy



Perplexity

ELMo Extractor was trained with AllenNLP by Gardner et al. (2017). We again, used a German version of Wikipedia trained with ELMo. We use GloVe and ELMo as token embedders and thus do not fine-tune on it. We concatenate both vectors, resulting in 1024 dimensions from ELMo and 300 from GloVe with a total of 1324 features per word. We train on a simple sequence tagger where words of the summary are used as tags on the source dataset. For training we use the ELMo model with a two layer LSTM with 256 dimensions and highway layers. Gradient clipping in combination with the adagrad optimizer. The final training accuracy was 0.94 with a loss of 0.15 and a validation accuracy of 0.93 and a loss of 0.19.

BertSum was trained with the code provided by Liu (2019). We did not change much of the hyperparameters. For BERT we used the multi-language and German huggingface implementation since the training would take weeks on a vast amount of resources. The model has a vocab size of 30.000 with a 12 attention heads and 12 hidden layers. Each hidden layer with a size of 768 and an intermediate layer size of 3.072. The hidden activation function is the gelu. We train for 30.000 steps, using the Noam decay with 10000 warm-up steps achieving a cross-entropy of 0.89 on the training set and 2.29 on the validation set.

Dynamic Convolutions was trained with Fairseq by Ott et al. (2019) a sequence-to-sequence learning framework. We first apply BPE with 40.000 subword units. Training with a cosine learning rate and the Adam optimizer. The loss of label smoothed cross-entropy. The warm-up steps are set to 10.000. We use 8 attention heads with an embedding dimensionality of 512. We apply the dynamic self attentional convolutions. During generation we use a beam size of 8, with a length penalty of 0.9 with trig-ram

blocking. The final training loss was 5.6 with a validation loss of 6.8.

5.5 Discussion

In this chapter we have presented three different results. We benchmarked our Bottom-Up BERT ensemble on the English CNN/DM dataset. We proceeded and established several state-of-the-art baselines on the description and headlines dataset. We also found out that Bottom-Up BERT did not perform as well on the German dataset as we hoped too. There are multiple reasons for this. Firstly the DPA dataset is not extractive dataset. We have seen with the CNN/DM dataset that BertSum greatly outperforms the abstractive baselines. There was reason to believe that the prior probabilities would aid in abstractive summarization inference. Secondly, In the case of headlines the extractive summaries were too long. Headlines are not really summaries and thus the headline generation is more of a proof of concept that generating longer and shorter sequences should be possible. Lastly, the hyperparameter tuning for state-of-the-art models like the copy transformer to a new dataset with different requirements from the academic datasets is cumbersome. We could possibly greatly improve our results here if we further explored this alley.

Technically there are some open questions, as of now the source code is spread across multiple projects and repositories. A lot of glue code holds all the models together. Adjustments are necessary for different datasets. We also did not follow up on all the different models and hyperparameter tuning. We compared summary generation with shorter source sequences (80) and longer source sequences (400). We tried to find better parameter sets for the transformer and pointer generator. For this automl would have been beneficiary. However we decided to use binary decision trees, paper recommendations and intuition vs automatic hyperparameter tuning. Hyperparameter optimization is very specialized and we decided to go into more breadth instead of depth. Also putting a lot of time into preprocessing the dataset than fiddling with over 30 parameters per model. Nonetheless, learning rates, warm-up steps, layer and hidden unit sizes should always be altered to see if differences occur. Therefore we used a test set of a few thousand examples to quickly iterate over different hyperparameters in a combinatoric way. The shortcomings are definitely that we did not optimize a single model for potentially even better results and instead trained multiple models. Keeping track of all the experiments was possible through very tidy book keeping. We did not implement any of the models

in full but rather relied heavily on frameworks like OpenNMT, Fairseq, AllenNLP and author specific implementations and results of these specific models. In table 5.8, a small overview of how many parameters were trained per model

Model	Encoder	Decoder	Total
ELMo (Peters et al. (2018))	-	-	137.781.526
BertSum (Liu (2019))	-	-	120.111.617
Pointer Generator (See et al. (2017))	7.322.624	35.467.093	42.789.717
Copy Transformer (Gehrmann et al. (2018))	22.930.048	24.265.253	47.195.301

Table 5.8: Number of model parameters

A few notes on language models. ELMo is not available in German so we used a recently trained ELMo model based on Wikipedia which deviates from the originals. Finding untrainable models, due to their computational costs, is also a real chore. BERT is available as a multi-language version and recently got a German version as well, which did not perform well. Sticking with the multi-language version seems to be the best approach. It is relatively easy to obtain German Word2Vec, GloVe or FastText embeddings, since they are officially available and trainable from scratch. Such models are however dated in comparison to the newer models. GPT-1 and GPT-2 could not be incorporated since their training time takes weeks on multiple TPU clusters with no pretrained German alternatives.

The training of the copy transformer reveals the problem space of testing for ROUGE evaluation vs testing for next word prediction. The gap between training and validation dataset was carefully tuned over weeks. We could not find a perfect match for the training data to converge on the validation data. This would also be an interesting topic for further research.

6 Conclusion

In this thesis we broadly gave an overview of neural text summarization and how to apply it to a German language corpus. We achieved a new high score on the ROUGE R-1 score on an English summarization dataset by combining two state-of-the-art models. We applied extensive feature extraction and fine-tuning models like GloVe, ELMo and BERT. By introducing a new German dataset from the DPA, we could test neural models on real world examples instead of highly optimized academic datasets. Subsequently we will glance at limits and generalization of our approach.

Generalization of neural models is a slippery slope. As far as generalization goes, the sequence-to-sequence framework makes it possible to transfer the models to any kind of dataset. However there are limits. How to tell whether a model generalizes? Evaluation would be possible based on performance on different tasks or out of domain datasets. Generalization in deep learning is a very unspecific topic, since it cannot be proven that something is able to deal with any kind of input. Instead, we provide intrinsic examples with idiomatic academic datasets that have some verifiable properties.

As far as the **transferability of neural models** is concerned, the answer is manifold. Language models like BERT and ELMo are dependent on the data source they were trained on. For instance, when building a summarization system in specialized domains such as medicine, law or engineering we will not be able to achieve the same results due to different contextualization of words and terminology. Both language models are based on news corpora and Wikipedia. The resulting model would struggle with a wide variety of out of vocabulary words. Specifically, the domain drift is too wide. To make models work with such different probability distributions requires careful fine-tuning and selection of applicable layers.

The **limits of neural models** are also well known. Deep learning is shorthand for creating a very complex differentiable, non-linear function and approximate its optimum by minimizing the error with respect to the steepest gradient change in high-dimensional

space. In language each dimension will contain some of the information provided by the dataset with respect to the statistical nature of text: Polysemy, hyponymy, synonymy, part-of-speech tagging, long range dependencies or named entities. Newer models like ELMo also partially encode the hierarchical nature of a word by contextualizing the sentence, paragraph or document it co-occurs with. As a result, all machine learning models that do not explicitly incorporate structural knowledge practically encode complex correlations on unstructured text. Simply put: it cannot tell causation from correlation. It does not handle prior knowledge. It is not transparent and explainable, since the solution space has potentially infinitely many parameter settings. Furthermore, our sequence-to-sequence models with attention suffer from very large parameter requirements. Our biggest model had 280 million parameters, with a vocabulary size of 50.000 or a BPE segmentation of 40.000. Small vocabulary size was partially tackled by copying out of vocabulary words directly from the source. It is questionable whether 50.000 tokens are enough to deal with highly specialized domains.

Sequence-to-sequence learning is a great successor to traditional frequency based approaches for generate fluent language. While most traditional algorithms perform extractive summarization, neural sequence models make it possible to encode the structural information required. In turn, fluency improves, repetition decreases and coverage of salient information increases. Moreover, they are end-to-end trainable, although not out of the box. Careful hyperparameter tuning for each dataset shifts the workload from tedious preprocessing towards searching a huge parameter space. With ever more complex models, new tools like automl have to emerge. It is possible to boost performance of all models with prior knowledge via fine-tuning and feature extraction. How much we could gain by training said models from scratch with specialized training data is not known. However, Merity et al. (2017); Radford and Salimans (2018) have shown that it is possible to learn and train language models from scratch achieving state-of-the-art results. This area of research will become incredibly relevant in the upcoming years. The reason being is that universal models can be distributed and thus must not be retrained. Another reason is increased performance by incorporating concepts that are not part of a specific dataset or only occur in small numbers.

7 Outlook

In this chapter we will provide an overview of some interesting ideas and topics that will be relevant in the upcoming years. We have decided to focus on transfer learning, reinforcement learning and multi-document summarization. Transfer learning has the most potential to achieve higher scores than any method yet. Reinforcement learning will gain relevance, since most models are not end-to-end differentiable. Thus, we deal with larger parameter spaces which only RL algorithms are known to optimize well. At last, we want to peek at a concrete example that is more relevant in industry context, namely multi-document summarization, where instead of one document at a time we process hundreds of documents at once.

7.1 Transfer Learning

One of the big upcoming topics in NLP will be transfer learning in all varieties. In this work we used ELMo via feature extraction and BERT via fine-tuning, showing that they achieve state-of-the-art results. There are, however, much more use-cases of fine-tuning. Edunov et al. (2019) showed that you can easily achieve new state-of-the-art results with pre-training and fine-tuning a language model based on ELMo. Hoang et al. (2019) made experiments adapting pretrained transformers, specifically GPT-1 by Radford and Salimans (2018) for abstractive summarization, achieving higher performance on abstractive datasets. This work was not yet published during the experimentation phase of this thesis and thus might be a good starting point for further research.

For pre-training and fine-tuning language models Merity et al. (2017); Howard and Ruder (2018); Radford and Salimans (2018) have found that adapting learning rates and applying regularization techniques increases performance. The primary goal is to generalize language models to as many datasets as possible by learning to solve problems in connection with polysemy, synonymy, part-of-speech, hierarchies and words within contexts.

We will see some groundbreaking results in the next years and ever more clever models and ideas on how to fine-tune to specific target tasks. The novelty will probably be not within the summarization literature but rather on general NLP problems like language modelling and sequence-to-sequence learning. Summarization is a nice practical task as a proxy for transfer learning performance.

7.2 Reinforcement Learning

Paulus et al. (2018); Celikyilmaz et al. (2018) and Zhang et al. (2019a) all have shown that summarization systems based on reinforcement learning lead to the best overall results. The reason being the inability to directly train the ROUGE evaluation metric with fully differentiable models. All RL approaches have in common that they globally optimize the ROUGE metric, directly outperforming current models. RL approaches are powerful since every model could be used within the agent framework introduced by Paulus et al. (2018). Future researchers should definitely think about ideas combining reinforcement learning with transfer learning to improve performance.

7.3 Multi-document Summarization and Dossiers

Multi-document summarization is one of the most interesting use cases. The benefits of single document summarization is limiting but useful. When the possibility arises to cluster documents around certain topics or keywords and summarize hundreds of documents by their most salient sentences, summarization makes more sense. Closely related is the concept of a dossier. Hälker (2015) defines a dossier as a collection of papers or other sources, containing detailed information about a particular subject. Key points are

1. Designed to fit a (topical) narrative / problem definition
2. Chronological / Historical / Hierarchical
3. Transparent and comprehensible
4. Presenting central (non-biased) arguments
5. Shallow at first glance, deep at second look

The requirements are of great variety and entirely out of scope for a single sequence-to-sequence architecture. A multi-document summarization system requires a multitude of different models solving different tasks. Such tasks include clustering, classification, topic modelling, language generation, discourse detection and sentence ranking.

Acknowledgements

First, I would like to thank Prof. Dr. Kai von Luck for the guidance and his relentless efforts to get things done. It would have been impossible to test out my ideas and train the models without the infrastructure provided through the Creative Space for Technical Innovations (CSTI).

Second, I would like to thank Prof. Dr. Marina Tropmann-Frick for the guidance and fruitful discussions on general topics relating to natural language processing.

Third, I would like to thank Tobias Eichler and Henrik Wortmann for their efforts in resolving all issues concerned with the infrastructure.

Last, I would like to thank the Machine Learning Group (ML-AG) of the HAW Hamburg for great discussions, especially Tasmin Herrmann and Stephan Halbritter.

Bibliography

Allahyari, M., Pouriyeh, S. A., Assefi, M., Safaei, S., Trippe, E. D., Gutierrez, J. B., and Kochut, K. (2017). Text summarization techniques: A brief survey. *CoRR*, abs/1707.02268.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. pages 1–15.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2001). A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). LDA-blei.pdf. 3:993–1022.

Brown, P. F., deSouza, P. V., Mercer, R. L., Pietra, V. J. D., and Lai, J. C. (1992). Class-based n-gram models of natural language. *Comput. Linguist.*, 18(4):467–479.

Celikyilmaz, A., Bosselut, A., He, X., and Choi, Y. (2018). Deep Communicating Agents for Abstractive Summarization.

Cheng, J., Dong, L., and Lapata, M. (2016). Long Short-Term Memory-Networks for Machine Reading.

Chopra, S., Auli, M., and Rush, A. M. (2015). Abstractive Sentence Summarization with Attentive Recurrent Neural Networks Sumit.

Chopra, S., Auli, M., and Rush, A. M. (2016). Abstractive Sentence Summarization with Attentive Recurrent Neural Networks. pages 93–98.

Chung, J., Gülçehre, Ç., Cho, K., and Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *CoRR*, abs/1412.3555.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. P. (2011). Natural language processing (almost) from scratch. *CoRR*, abs/1103.0398.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE*, 41(6):391–407.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

Dong, Y. (2018). A survey on neural network-based summarization methods. *CoRR*, abs/1804.04589.

Edunov, S., Baevski, A., and Auli, M. (2019). Pre-trained Language Model Representations for Language Generation.

Emma Strubell, A. G. and McCallum, A. (2019). Energy and policy considerations for deep learning in nlp. *In the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Gardner, M., Grus, J., Neumann, M., Tafjord, O., Dasigi, P., Liu, N. F., Peters, M., Schmitz, M., and Zettlemoyer, L. S. (2017). Allennlp: A deep semantic natural language processing platform.

Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. (2017). Convolutional Sequence to Sequence Learning.

Gehrmann, S., Deng, Y., and Rush, A. M. (2018). Bottom-Up Abstractive Summarization.

Goldberg, Y. (2019). Assessing bert's syntactic abilities. *CoRR*, abs/1901.05287.

Graves, A., Wayne, G., and Danihelka, I. (2014). Neural Turing Machines. pages 1–26.

Heinzerling, B. and Strube, M. (2017). Bpemb: Tokenization-free pre-trained subword embeddings in 275 languages. *CoRR*, abs/1710.02187.

Hoang, A., Bosselut, A., Celikyilmaz, A., and Choi, Y. (2019). Efficient Adaptation of Pretrained Transformers for Abstractive Summarization.

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

Howard, J. and Ruder, S. (2018). Universal Language Model Fine-tuning for Text Classification.

Huang, F. and Yates, A. (2009). Distributional representations for handling sparsity in supervised sequence-labeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 495–503, Suntec, Singapore. Association for Computational Linguistics.

Hälker, N. (2015). Teilautomatisierte Erstellung von Dossiers auf der Basis von Textmining-Verfahren.

Jawahar, G., Sagot, B., and Seddah, D. (2019). What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2016). Bag of Tricks for Efficient Text Classification.

Kim, B., Kim, H., and Kim, G. (2018). Abstractive Summarization of Reddit Posts with Multi-level Memory Networks.

Kim, Y., Jernite, Y., Sontag, D., and Rush, A. M. (2015). Character-aware neural language models. *CoRR*, abs/1508.06615.

Klein, G., Kim, Y., Deng, Y., Senellart, J., and Rush, A. M. (2017). OpenNMT: Open-Source Toolkit for Neural Machine Translation. *ArXiv e-prints*.

Levy, O. and Goldberg, Y. (2014). Neural word embedding as implicit matrix factorization. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 2177–2185. Curran Associates, Inc.

Li, J., Luong, M.-T., and Jurafsky, D. (2015). A Hierarchical Neural Autoencoder for Paragraphs and Documents.

Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries. In *Proc. ACL workshop on Text Summarization Branches Out*, page 10.

Liu, Y. (2019). Fine-tune BERT for Extractive Summarization.

Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective Approaches to Attention-based Neural Machine Translation.

Merity, S., Keskar, N. S., and Socher, R. (2017). Regularizing and Optimizing LSTM Language Models.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. pages 1–9.

Nallapati, R., Zhou, B., dos Santos, C. N., Gulcehre, C., and Xiang, B. (2016). Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond.

Olah, C. (2015). Understanding lstm networks. *GITHUB blog, posted on August*, 27:2015.

Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., and Auli, M. (2019). fairseq: A Fast, Extensible Toolkit for Sequence Modeling.

Paulus, R., Xiong, C., Socher, R., and Alto, P. (2018). a Deep Reinforced Model for Abstractive Summarization. pages 1–13.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *In EMNLP*.

Peters, M., Ruder, S., and Smith, N. A. (2019). To Tune or Not to Tune? Adapting Pretrained Representations to Diverse Tasks.

Peters, M. E., Ammar, W., Bhagavatula, C., and Power, R. (2017). Semi-supervised sequence tagging with bidirectional language models. *CoRR*, abs/1705.00108.

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

Popel, M. and Bojar, O. (2018). Training Tips for the Transformer Model. pages 1–28.

Radford, A. and Salimans, T. (2018). Improving Language Understanding by Generative Pre-Training (transformer in real world). pages 1–12.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners.

Radim, R. and Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA.

Ruder, S. (2019). Neural Transfer learning for NLP.

Rush, A. M., Chopra, S., and Weston, J. (2015). A Neural Attention Model for Sentence Summarization.

See, A., Liu, P. J., and Manning, C. D. (2017). Get To The Point: Summarization with Pointer-Generator Networks.

Sennrich, R., Haddow, B., and Birch, A. (2015). Neural Machine Translation of Rare Words with Subword Units.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. pages 1–9.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention Is All You Need. (Nips).

Wiseman, S. and Rush, A. M. (2016). Sequence-to-Sequence Learning as Beam-Search Optimization.

Wu, F., Fan, A., Baevski, A., Dauphin, Y. N., and Auli, M. (2019). Pay Less Attention with Lightweight and Dynamic Convolutions. pages 1–14.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., and Bengio, Y. (2015). Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2048–2057, Lille, France. PMLR.

Zhang, H., Xu, J., and Wang, J. (2019a). Pretraining-Based Natural Language Generation for Text Summarization.

Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., and Artzi, Y. (2019b). Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Zhou, Q. (2018). Neural Document Summarization by Jointly Learning to Score and Select Sentences. pages 654–663.

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „– bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] – ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: _____

Vorname: _____

dass ich die vorliegende Mastertarbeit – bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Ansätze zu Deutscher, Abstrahierender Textzusammenfassung mit Deep Learning

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

_____	_____	_____
Ort	Datum	Unterschrift im Original