UNIVERSITY OF BIELEFELD

BACHELOR THESIS

Deep recurrent neural networks for abstractive text summarization

Author:
Marie KLOENNE

Supervisor:
Prof. Dr. Barbara HAMMER
M.Sc. Johannes Brinkrolf

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science

in the

Technical Faculty

May 8, 2018

Declaration of Authorship

I, Marie KLOENNE, declare that this thesis titled, "Deep recurrent neural networks for abstractive text summarization" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"Torture the data, and it will confess to anything."

Ronald Coase

UNIVERSITY OF BIELEFELD

Abstract

Technical Faculty

Bachelor of Science

Deep recurrent neural networks for abstractive text summarization

by Marie KLOENNE

This thesis is dealing with the creation of a model for abstractive text summarization. For this purpose, recurrent neural networks are used to generate accurate summaries of given texts in the correct English language and context. We are appending a combination of recurrent neural network with hierarchical attention followed by Long Short Term Memory Networks (LSTM) building an auto-encoder structure.
This work shows a possible upgradeable variant for automatically summarizing texts and can now be expanded for further research. The abstract compilation of texts is still in its infancy, and there are still many different open possibilities waiting to be realized.
There are still some issues to fix like the amount of repetitions in the generated texts and getting the right context of the text, a possible point of approach for future works.

Acknowledgements

I wish to express my sicere thanks to the German Aerospace Center, for providing me with all the necessary facilities for the research.
I place on record, my sincere thank you to Prof. Dr. Barbara Hammer, for the continuos advices.
I am also grateful to Sivasurya Santhanam. I am extremly thankful and indebted to him for sharing expertise, and sincere and valuable guidance and encouragement extended to me.
Finally, I would like to thank all those who supported and motivated me during this bachelor thesis.

Contents

List of Figures

List of Tables

List of Abbreviations

RNN	**R**ecurrent **N**eural **N**etwork
LSTM	**L**ong **S**hort **T**erm **M**emory Network
A-RNN	**A**ttentional - **R**ecurrent **N**eural **N**etwork
NLP	**N**atural **L**anguage **P**rocessing
GPU	**G**raphics **P**rocessing **U**nit
RRA	**R**ecurrent **R**esidual **A**ttention
Seq2Seq	**Seq**uence to **Seq**uence

Hyperparameters

BS	Batchsize	= The size of the training inputs split to a batch for each iteration
IT	Iterations	= Number of Network parameters updates for each input instance
EP	Epochs	= Number of Network parameters updates for each input instance
AS	Attentionsize	= Linear size of the Attention weights
HS	Hiddensize	= Number of hidden layers
ES	Embeddingsize	= Dimension of embedding vector
AD	Alphadivider	= Divider of the number of alpha weights (output attention layer)
LR	Learningrate	= See Backpropagation (2.1)
KP	Keep Probability	= Control parameter for the dropout rate
D	Delta	= Linear coefficient threshold

Hyperparameters are those values that are set manually before the start of the learning process.

Chapter 1

Introduction

"The establishment of shared theoretical frameworks, combined with the availability of data and processing power, has yielded remarkable successes in various component tasks such as speech recognition, image classification, autonomous vehicles, machine translation, legged locomotion, and question-answering systems.
As capabilities in these areas and others cross the threshold from laboratory research to economically valuable technologies, a virtuous cycle takes hold whereby even small improvements in performance are worth large sums of money, prompting greater investments in research. There is now a broad consensus that AI research is progressing steadily, and that its impact on society is likely to increase. Because of the great potential of AI, it is important to research how to reap its benefits while avoiding potential pitfalls." - Steven Hawking

1.1 Motivation

The great benefit of the automatic merging of texts becomes more and more apparent with the increasing flow of data as well as the increased data availability. Deep learning is becoming an increasingly popular topic, due to the accretiving computing power. This also drives the NLP division, which is an essential component of future and current research. Speech and text will always accompany us and thus represents an essential field for neural networks. The possibilities offered by the automatic generation of texts, translation, etc. promise a high workload relief for future areas. This work deals with the handling of recurrent neural networks (RNN), for the abstract generation of text summaries. This enables to immerse in the field of Natural Language Processing (NLP) and to gain first insights and experience in dealing with RNNs in practice. The topic of abstractive text summarization is extraordinary, because it offers a possibility to generate sentences or whole texts with its own choice of words, while maintaining the grammar and context of the language and text. Research on Abstractive Text Summarization has been under the way for several years and is still relatively young in the field of NLP. The generation of important keywords is already very successful. However, the occurrence of repetitive word sequences and finding the right context is still a problem which needs to be solved.

The benefit of a abstractive summarization model becomes apparent when we look at the amount of texts that circulate worldwide. With an automatic generation of variable-length summaries,

1. the work of manually compiling texts is significantly reduced, in the best case taken completely, and
2. summaries can be delivered so that readers do not have to read the texts completely, in order to determine that the content of the text may not be useful or relevant.

The presented summarization model is interesting for a wide range of users, who could save the creation of many summaries or even the reading of complete texts. In this way, the benefits are relevant for multiple fields, such as economics, research and for private use. Neural networks have been around for almost 60 years, but it is precisely now that the computing power is powerful enough to implement such a system. With training times of less than one week one can combine several 10000 texts and the tendency is rising. As the number of textual data grows, so does the use of short summaries, to get an overview before reading.

1.2 Problem definition

Up to now, previous works deal mostly with extractive text summarization, where words or passages that are important get extracted from texts in order to obtain an accurate summary of the text. A weakness of extractive text summarization is, that it only extracts the important sentences from the text without doing an independent summary by itself. Now, more and more people are looking for possibilities of freer translations or summaries to solve the problem of just getting sentences from the original text as a summary towards generating a summary in own words like human do. [The field of natural language processing is working towards humanlike translations and summaries. It is in the interest of getting high quality summaries, which make sense.] Neural networks provide a good foundation for NLP. Building on this foundation, the system should be able to understand the context of the paragraphs and assemble them into a grammatically correct sentence. The current state of research for abstract text summaries still shows flaws in the area of word repetition and context sensitivity. For this reason the task of this thesis is to find an approach that implements the two properties of context-based and grammatical sentence generation. To achieve this, hierarchical attention is combined with memory networks, the so-called RNN. In short, this work explores a new combination of common methods for automatically generating text summaries.

Chapter 2

Fundamentals

The following chapter deals with the basics for a general understanding of the treated work.

2.1 Neural Networks

This section is based on [2]. Artificial neural networks were originally treated as an analogy to the neural networks of the brain, first introduced in 1943 by Warren McCulloch and Walter Pitts, to solve learning problems (in order to be able to learn). Approximately 40 years later, interest in the neural networks increased and continues to this day, even though the analogy to the brain is often no longer in focus.

Neurons

Each neuron (also called units or nodes) contributes to the learning process by receiving and processing stimuli in the form of weighted inputs and an offset, the bias. In detail, the weighted inputs are summed up and supplemented by the offset. The calculated value is then transferred to the activation function.

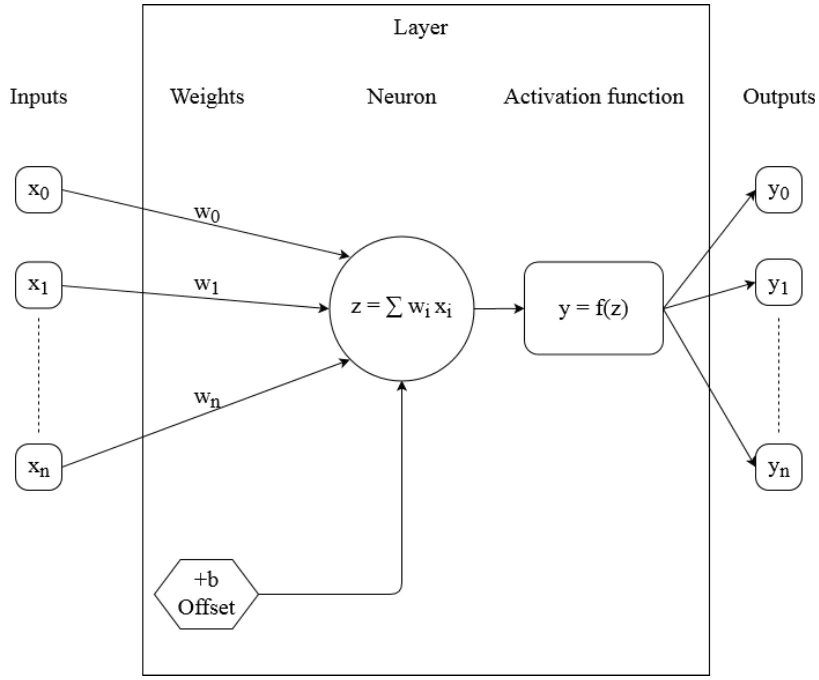


FIGURE 2.1: Neuron by [2].

Activation functions

There are different functions for the activation of a neuron. The 3 top used ones are the sigmoid function, the tanh function and the ReLu(Rectifier Linear Unit) function. In more recent approaches, ReLu (or related piecewise linear activation functions) are among the more popular methods as they avoid the vanishing gradient problem [6]. Every function have the same intention, to generate an output of different intensity, depending on the input. In doing so the functional relationship for any of the functions is not linear. In summary, each neuron generates one output for each of its specific weighted inputs. The interlinking of these neurons then makes it possible to tackle more complex learning problems.

Example of a feed forward neural network

A neural network is build by many neurons, which could end up in many different architectures. Therefore the neurons can be put into three categories. The input layer, the hidden layers and the output layer. There is always one input- and one output layer for each neural network. To say it is a deep neural network, the number of the layers must be high enough, e.g. 30 layers or even 2 layers (For the counting of the layers, the input layers do not count). If every neuron of the layer is connected to every neuron on the next layer, so that the layers are fully connected, we call it a Dense-Layer.

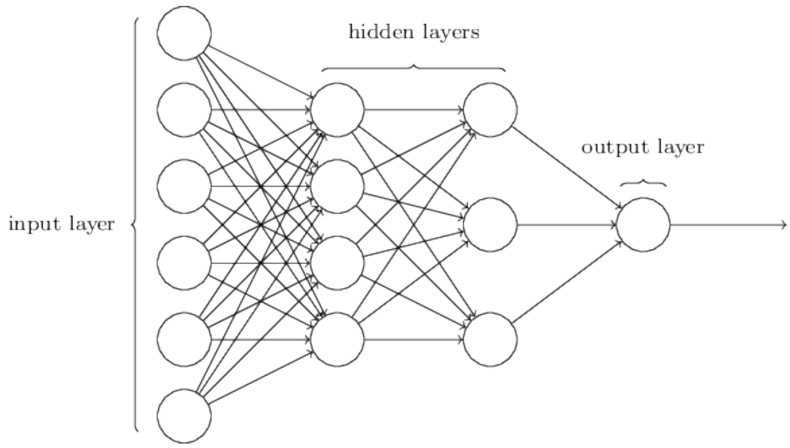


FIGURE 2.2: A feed forward neural network by [11].

Objective function - loss

Similar to humans understanding of learning the neuron also need to know how good it performance was to improve themselfs over time. One way of describing the performance of the network is to recognize how much the learned result of the network deviates from the desired result. That's what the loss is for. The loss describes the error made by the neural network. There are two different kinds of loss functions, the supervised, i.e. functions which take into account desired outputs, and the unsupervised loss functions, which judge the rationality of the network based on the given input signals only. In my work, I will only consider supervised loss, which shows the error compared to the true values of our targets.

Backpropagation

In the backpropagation learning rule, short Backpropagation [15] the neurons which were responsible for a to high loss are getting the error back, to check wether the error is increasing or decreasing when the activation is getting increased. After that there will ideally be a direction for either increasing or decreasing the activation. How much of the error has to be corrected depends on the proportion of the neuron in the result and the previously set learning rate. For example, if the learning rate is 0.001, only (the calculated error)x(0.001) is corrected. Formally, the backpropagation learning rule has been derived as a particularly efficient way to compute the gradients of a given loss function for a feed forward network by a set of update rules which propagate derivations backwards through the networks.

The learning part

When we move on to the actual process of learning, it is important to start by randomly weighting the individual connections. Then the presented processes become active and the neurons calculate their output to be processed further by the activation function. After calculating the loss, backpropagation is used. The preceding processes can be repeated as often as required and form an epoch/iteration. The number of epochs to choose is a hyperparameter, which must be adapted for every learning purpose individually, to get the golden mean between overfitting and underfitting. Mathematically speaking, such learning rules implement a form of gradient descent which often processes error signals in mini-batches.

2.1.1 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are Neural Networks with special forms of feedback from neurons in one layer to neurons in the same or previous layers. These special forms are RNNs with:

- **direct feedback**

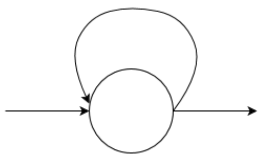
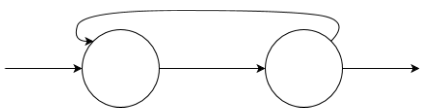


FIGURE 2.3: direct feedback by Beck

  In this case, the output/activation of the neuron is directly given as an additional input to itself.

- **indirect feedback**



  In this case, the output of the neuron is given as an additional input to the previous layer neurons.

FIGURE 2.4: indirect feedback by Beck

- **lateral feedback**

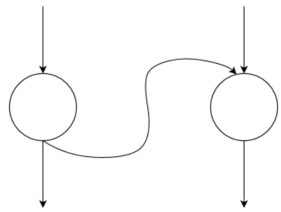


FIGURE 2.5: lateral feedback by Beck

In this case, the output of the neuron is given as an additional input to neurons of the same layer.

- **complete feedback**

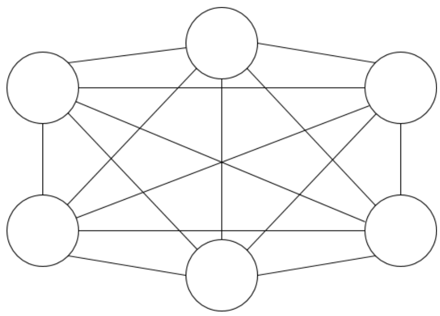


FIGURE 2.6: complete feedback by Beck

In this case. the output of the neuron is given as an additional input to neurons of all layers in the network. (cf. Beck)

Recurrent connections extend the realm of networks towards dynamic, temporal systems, used e.g. for time series processing (for partial feedback) or the realization of associative memory architectures (for complete feedback). In the thesis we will have a look at time series processing with **direct feedback** RNNs.

Advantage and Disadvantage of RNNs

RNNs were build to deal with sequences and time series. Especially for those purposes it could be needful to have information about previous states, e.g. for generating the next word of a sentence. Basical RNNs in practice can only hold a 'memory' for a few timesteps, due to the so-called problem of long term dependencies, Hochreiter and Schmidhuber invented the LSTMs - Long Short Term Memory Networks for remembering more (previous) steps in a sequence[6].

Long Short Term Memoy Networks

Long Short Term Memory Networks are a special form of Recurrent Neural Networks.The special thing about them is, that they avoid the Vanishing Gradient Problem that occurs with RNNs by having a strong memory. This means that the network can also remember dependencies over long sequences (cf. [4]).

2.2 Summarization-Basics

2.2.1 Seq2Seq

Seq2Seq is an architecture based on Deep Neural Networks, invented by the Google Researchers Ilya Sutskever, Oriol Vinyals and Quoc V. Le in 2014 [16]. They used the Seq2Seq technique to fill the gap of handling the mapping of sequences to sequences within neural networks. To realize that, they used an end-to-end strategy with two LSTMs. One LSTM is applied for acting as an Encoder and the other LSTM for acting as a Decoder. Therefor the Encoder is mapping the input sequence into a fixed size vector, which is used from the Decoder to generate a new sequence (target sequence) out of it. Their paper thereby used this approach to produce a translation of an English sequence to a French sequence.(cf. [16])

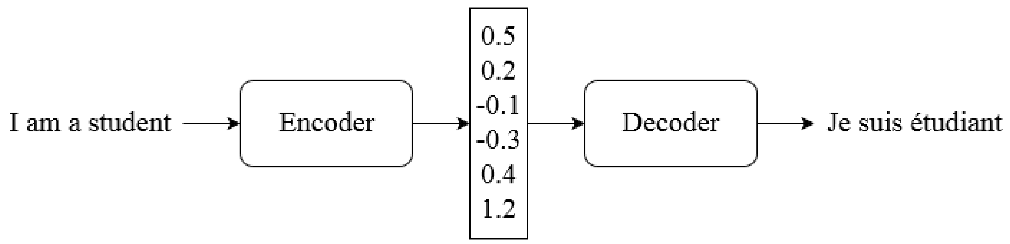


FIGURE 2.7: Sequence to Sequence architecture by [18].

2.2.2 Attention

Hierarchical Attention

Yang et al. offers a hierarchical attention network for classifying documents. The model has two characteristic features:

1. It has a hierarchical structure for reflecting the hierarchical structure of documents;

2. It provides two tiers of attention mechanisms on word and sentence level, which allow it to focus on more and less important content in the document.

These two attention mechanism both are based on the attention mechanism introduced by D. Bahdanau, explained in the next section. Experiments show that the proposed architecture far exceeds previous methods. (cf. [20])

Other Attention mechanisms

Bahdanau used in Hierarchical attention

For the attention mechanism a vectorrepresentation has to bear the load of encoding the "meaning" of the whole input sequence. Despite the huge variance in the

language, this appends very difficult. For this purpose, encoders and decoders are used, which must be able to recognize important but also small differences.
Bahdanau et al.[1] introduced an attention mechanism that addresses the problem of paying attention to parts of the input by the used decoder. Therefor the decoder, step by step, chooses a different part of the input sequence to 'pay attention' to.
To calculate this attention, Bahdanau is using another feedforward layer in the decoder. The feed forward layer then uses the current input and the hidden state to generate a new vector of the same fixed size as the input sequence. This vector is processed by softmax to calculate attention weights multiplied by the outputs of the encoders. By doing so he is creating a new context vector that is used for the prediction of the next output[1].

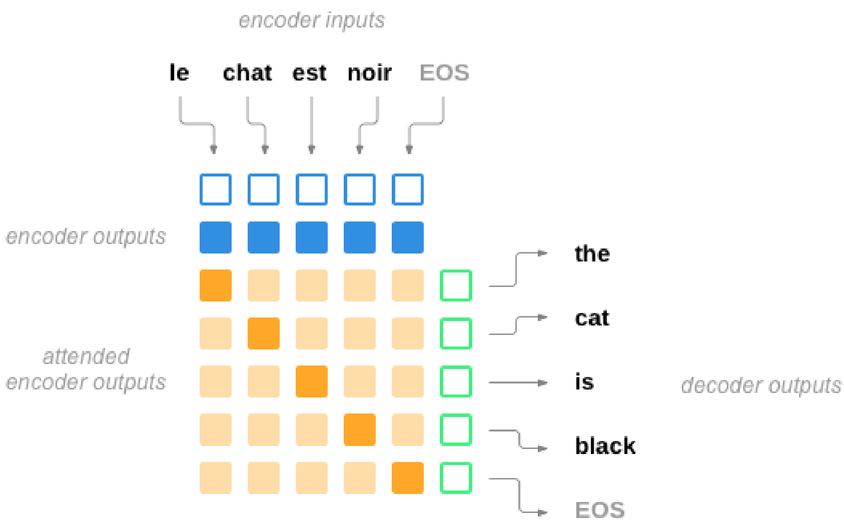


FIGURE 2.8: encoder-decoder

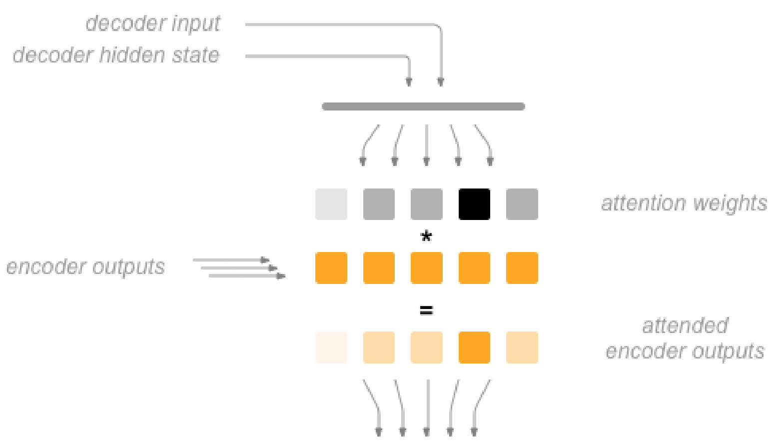


FIGURE 2.9: feedforward layer in the decoder

FIGURE 2.10: images has been taken from [14]

[1] 14.

Luong Attention used in LSTM part

In Luong attention alignment at the current time step t is computed by using the hidden state at time step t and all source hidden states. It creates an independent RNN-like structure to take the concatenatation of the context vector of the current time step and the hidden state of the current time step as input and output the new hidden state which serves to predict the output of the current time step and add additional input features.(cf.[9])

2.3 Word-Embedding Methods

2.3.1 General

The neural networks could only handle numbers for their computations. Thats why the input words first have to be changed into a representation based on numbers. There are different ways to represent a word with numbers for feeding it into a neural network. Previously, One-hot encoding often was used. It represents words by vectors. Each vector got the length of the number of words and represents the current word by 1 and all other words by 0. The disadvantage of one-hot encoding is, that it do not preserve any information, e.g. of the words context, and the length of the vectors is getting super long when dealing with many words. We will go through a short explanation of the two techniques Word2Vec and GloVe. This work is going to deal with the GloVe Embedding.

2.3.2 Word2Vec

Word2Vec is a context based prediction method, to represent words with context vectors. This means, that words with a similar context are getting arranged nearby each other in the vector space. (cf. [10])

2.3.3 GloVe

GloVe constittues a popular alternative to Word2Vec which is based on extensive statistics of given corpora. Pennington, Socher and Manning characterize its design in the following way: 'GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.' [12]
Later on, we will rely on GloVe as a vectorial embedding tool for text, and deep recurrent and lstm techniques for learning our tasks. It turned out that the introduced attention mechanism is crucial to obtain performant models.

Chapter 3

Materials and methods

3.1 Concept

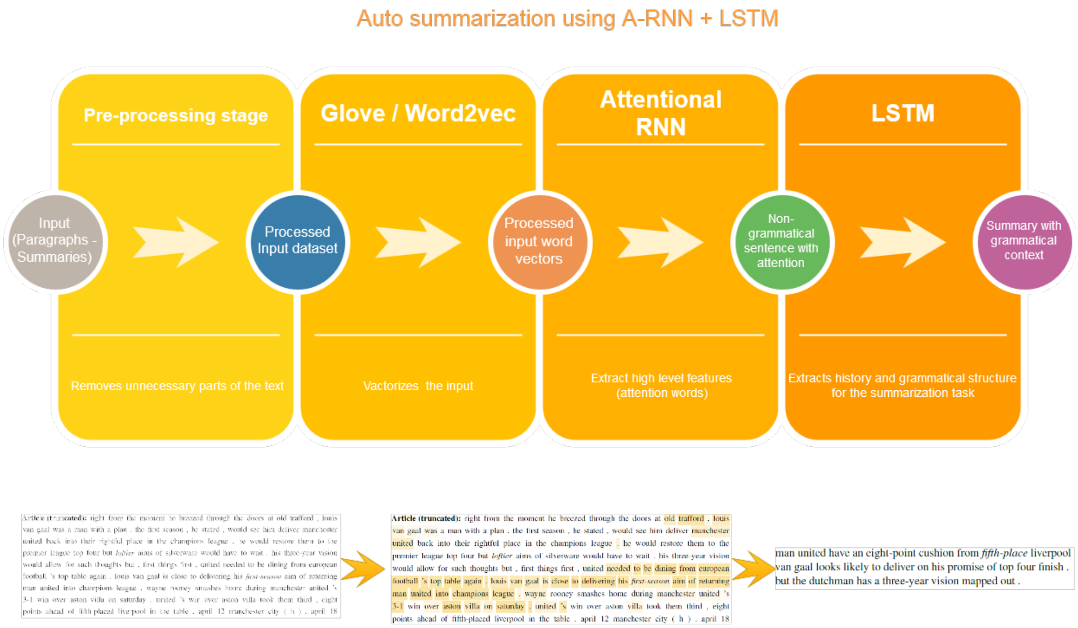


FIGURE 3.1: Architecture auto summarization.

In this section we are going to explain the basic structure of the model. We want to give text as an input and then use a series of adapted RNNs to get an automatically generated summary. We have developed a special architecture for this. The architecture of the model can be split into 4 stages. The preprocessing stage, the Word Embedding stage, the Attentional Recurrent Neural Network stage and the Long Short Term Memory Network stage. First of all the dataset has to be preprocessed. That means, that unnecessary parts of the texts has to be removed in order to avoid the neural networks learning from unnecessary words, e.g. punctuations, stop words or numbers. On top of that, the texts need to be divided in an article and in a summary part, which is important for the setup of our later inputs and targets/labels used in the Neural Networks. After the raw data got preprocessed, the words of each paragraph and each corresponding summary are getting embedded, so that the neural network can handle its given input data. The words are changed into an integer based representation to feed into the attentional RNN. The attentional RNN summary, weighted values for each word via an hierarchical attention mechanism so that the output of the attentional RNN part is going to be a non-grammatical sequence of highlighted words for each the input articles. Finally the LSTM generates a summary out of the highlighted words, outputted by the attentional RNN, and by the original summary, which should capture the grammatic and context.

3.2 Model

In the Model section we will look at the more precise implementation of the architecture and what exactly was used for its realization. The presented architecture is realized by an implementation based on the programming/script language python 3. For the input texts of our model, we use articles from CNN Dailynews.

Preprocessing

Before the raw text can be fed into the neural networks, it has to be preprocessed. Therefor, the texts are getting split into an article and a summary part. Later on the preprocessed article is going to be our input X and the preprocessed summary is going to act as our target Y. The preprocessing contains the following steps:

1. the data is getting split to the article part and the summary part
2. both articles and summaries are getting cleaned now. This cleaning contains:
   - converting every word to lowIn der Model Sektion werden wir die genauere Umsetzung der Architektur betrachten und was zur Realisierung genau genutzt wurde. ercase
   - removing punctuations
   - removing multiple arranged white spaces
   - removing start and end phrase of each article, e.g. location, author, copyright
   - removing '@highlight' for the summary part
   - replacing contractions like 'aren't' with 'are not'
3. remove stopwords from the summary input of the attentional RNN
4. create word vectors for each paragraph and its summary
5. for the autoencoder we are using a pretrained GloVe embedded vocabulary out of 400K uncased words with a vector dimension size of 50 for each word representation[1] which is added by every word of the articles and its GloVe vector representation if it is not already contained in the vocabulary.
6. for the attentional RNN every word has to be represented by an integer. Therefor the inputs for the A-RNN are getting converted into integer representations, realized by the Keras preprocessing tokenizer and after that finally been put in Numpy arrays.

[1] 7.

LONDON, England (Reuters) -- Harry Potter star Daniel Radcliffe gains access to a reported £20 million ($41.1 million) fortune as he turns 18 on Monday, but he insists the money won't cast a spell on him.
Daniel Radcliffe as Harry Potter in "Harry Potter and the Order of the Phoenix"
To the disappointment of gossip columnists around the world, the young actor says he has no plans to fritter his cash away on fast cars, drink and celebrity parties. [...] Meanwhile, he is braced for even closer media scrutiny now that he's legally an adult: "I just think I'm going to be more sort of fair game," he told Reuters. E-mail to a friend
Copyright 2007 Reuters. All rights reserved.This material may not be published, broadcast, rewritten, or redistributed.

@highlight
Harry Potter star Daniel Radcliffe gets £20M fortune as he turns 18 Monday
@highlight
Young actor says he has no plans to fritter his cash away
@highlight
Radcliffe's earnings from first five Potter films have been held in trust fund

FIGURE 3.2: Example raw article

['harry', 'potter', 'star', 'daniel', 'gains','access',...,'to', 'a', 'friend']

FIGURE 3.3: Extract preprocessed article text

['harry', 'potter', 'star', 'daniel', 'radcliffe', 'gets', 'also', 'fortune', 'as', 'he', 'turns', '18', 'monday', 'young', 'actor', 'says', 'he', 'has', 'no', 'plans', 'to', 'fritter', 'his', 'cash', 'away', 'radcliffes', 'earnings', 'from', 'first', 'five', 'potter', 'films', 'have', 'been', 'held', 'in', 'trust', 'fund']

FIGURE 3.4: Preprocessed summary

Realization of the Attentional RNN

The first learning part of our model, which is the attentional RNN, handles its input only represented as integers. To reach this representation, the words of the created word vectors needs to be converted to integers, laying in Numpy arrays. After finishing this convertion, the vector representation of the articles are put into multiple batches for the minibatch training of the following implementation:
For the attentional RNN we are using a Bi-directional Basis RNN-Structure combined with a dropout and an attention layer. The attention layer implements the attention mechanism of Yangs Hierarchical Attention, which includes the Bahdanau

Attention for its two attention lookups. We are using a Dropout for the training process which is followed by the prediction of Y-hat. To calculate the loss via sigmoid cross entropy with logits [17] we use the predicted Y-hat for the logits input and the integer represented summary for the labels, continued by an Optimization, the Adam algorithm [8].

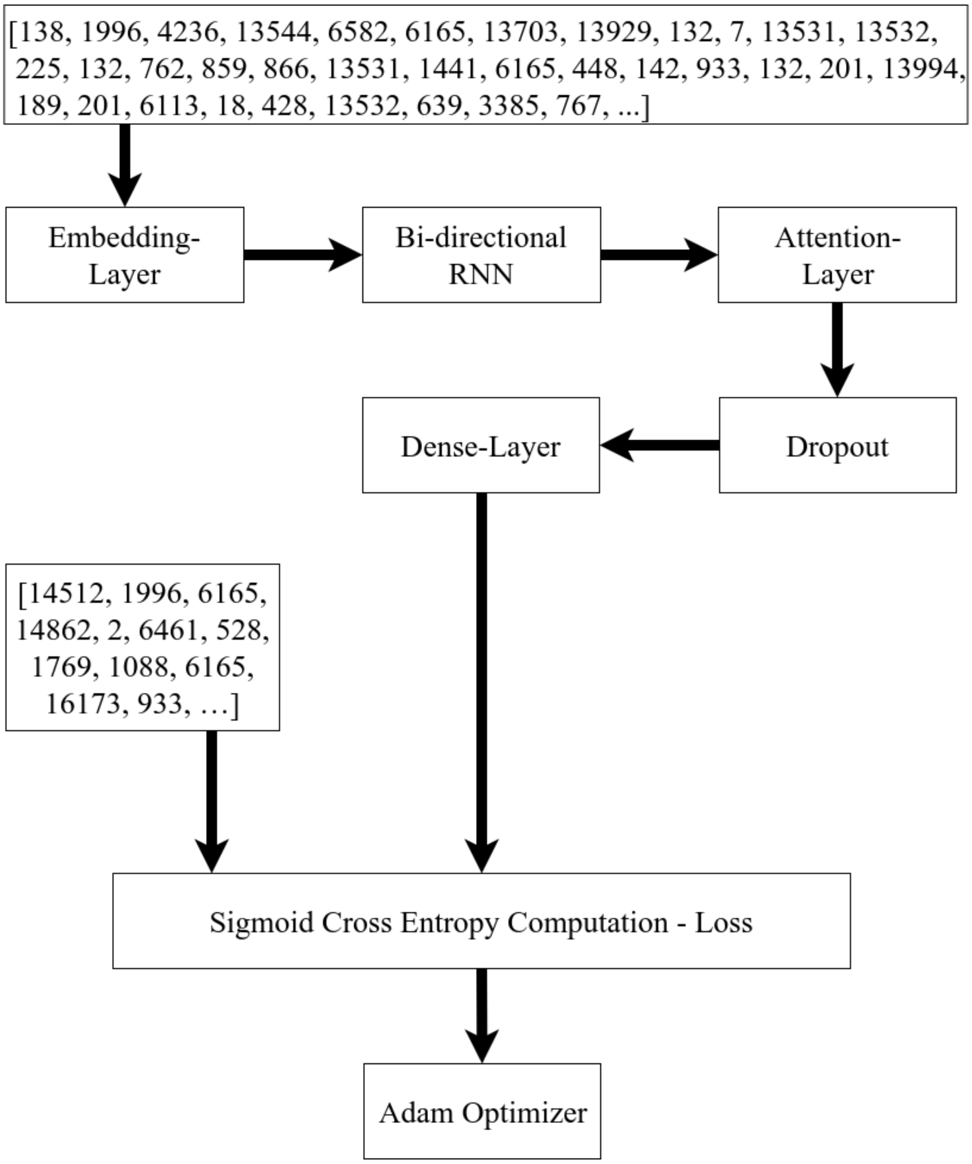


FIGURE 3.5: Structure A-RNN

Realization of the Auto-Encoder with LSTMs

The last part of the model is based on the implementation of Jishnu Ray Chowdhurys Model for Abstractive Textsummarization[2].
The LSTM based architecture follows the Seq2Seq architecture, invented by [16] with local attention. Doing so the input words are converted to be represented by GloVe word embeddings. If we look at the Seq2Seq architecture, it is roughly divided into an encoder and a decoder part. For the encoder we use a bidirectional LSTM with

[2] 5.

RRA [19]. This means that each input is processed both forwards, starting with the first word, and backwards, starting with the last word.
For the RRA, K is the number of hidden states that must be taken into account for remaining connections. The model calculates the weighted sum of the previous K hidden states, which ends up as the RRA.
Ensuing, the attention mechanism by Luong is being appended to create a context vector (established by the weighted hidden states in the D depending attention window) and calculate the probability distribution for the first output token from the context vector and decoder hidden state.
Therefor we need the hyperparameter D, which creates an attention window as follows: The attention window contains only the hidden states in the range pt-D to pt+D. Pt is the current attention center. [9].
"The word vector represention of the output token - the word with maximum the predicted probability in the recently calculated probability distribution, is used as the decoder input token. The output decoder hidden state from that current decoder input token, and the hidden state, is used again in the next loop to calculate the probability distribution of the next output token and so on. The loop continues for 'output_len' no. of iterations" [8].
After that we calculate the loss via sigmoid cross entropy with logits [17]. Therefor we use the outputs of the autoencoder for the logits input and the GloVe embedded summary for the labels, continued by an Optimization, the Adam algorithm [8].

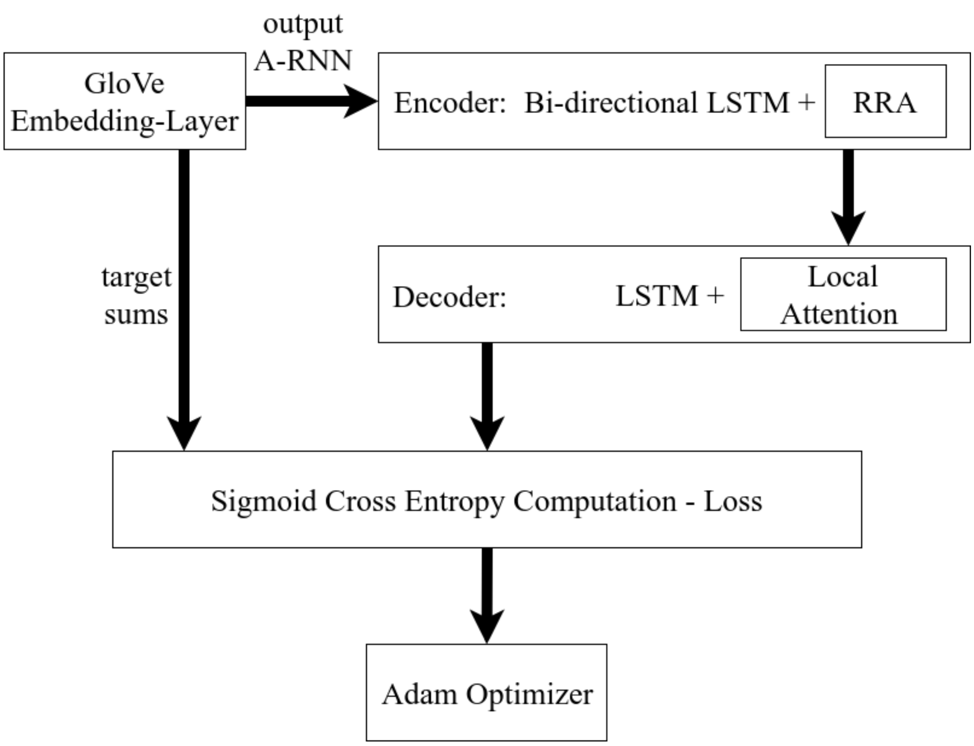


FIGURE 3.6: Structure Auto-Encoder LSTM

Chapter 4

Results and Discussion

The succeeding chapter introduces the individual experiments and their evaluation, as well as the dataset and the used hyperparameters.

4.1 Dataset

In this work we are using a Dataset extracted from CNN Dailymail Newsarticles. The advantage of the CNN Newsarticles, as a dataset for abstractive text summarization, is their structure. Each article has the typical article part and some additional sentences, which should summarize the highlights of the text in other words. Based on this structure, we used the article text for our input data, where we want to extract abstractive summaries from. The highlight sentences are used for the target labels in our model, as a representation of the hand written summaries.

4.2 Experiments & Evaluation

The Training ran on a Quadro P5000, as well as on a Tesla K40 GPU. For the evaluation of our model performance, we splittet the model evaluation up, to evaluate each learning part seperately. Therefor we applied the pyrouge implementation of the ROUGE score ones for the attentional RNN and the LSTM[1]. The ROUGE score is an evaluation technique containing the calculation of the precision, the recall and the resulting f-measure.
The F-measure can be used to measure the test's accuracy. Its score is calculate by the precision p and the recall r. Therefor p is the number of correct positive results divided by the number of all positive results and r is the number of correct positive results divided by the number of all positive samples. The F-Measure, also called F1 score builds the harmonic average of the precision and the recall and is best at value 1, worst at 0.
In our evaluation we are using the ROUGE score(Recall-Oriented Understudy for Gisting Evaluation score), which is used for evluation purposes in automatic summarization and machine translation. This metrics comparison could be used for summary or translation measurings, by comparing them against a set of references (human-produced) summary or translation (cf. [13]).
To look at the results of the experiments, an example text is taken and evaluated.

Experiment 1

Number of training data: 1000 articles
Runtime A-RNN on Quadro P5000 GPU: 7408.1844 sec

[1] 21.

Runtime LSTMs on Quadro P5000 GPU: 38503.15521 sec
(Training duration round about 12,5 hours)
Original Summary Example:
['sullivan', 'regains', 'the', '50meter', 'world', 'freestyle', 'record', 'in', 'sydney', 'sullivan', 'sets', 'the', 'new', 'mark', 'of', 'top', 'seconds', 'in', 'the', 'australian', 'olympic', 'trials', 'frenchman', 'alain', 'bernard', 'recorded', 'a', 'time', 'of', '2150', 'seconds', 'four', 'days', 'earlier']

Attentional RNN

Hyperparamter	Epochs	ED	HS	AS	KP	BS	Delta	LR	AD
RNN	3	100	150	50	0.8	50	0.5	0.003	50

TABLE 4.1: Hyperparameters of the Attentional RNN

Y: Example Input Target Summary:
['eamon', 'sullivan', 'regains', '50meter', 'world', 'freestyle', 'record', 'sydney', 'sullivan', 'sets', 'new', 'mark', 'top', 'seconds', 'australian', 'olympic', 'trials', 'frenchman', 'alain', 'bernard', 'recorded', 'time', '2150', 'seconds', 'four', 'days', 'earlier']
Example Output Predicted Highlights:
['sydney', 'man', 'record', 'at', 'in', 'a', 'off', 'by', 'duel', 'year', 'tell', 'just', 'events', 'girls', 'through', 'friend']
ROUGE Score results of the training example:

Scorings	ROUGE-Score	Precision	Recall
A-RNN	0.04651258118	0.0625	0.03703703704

TABLE 4.2: ROUGE-Score Example Attentional RNN

LSTM Auto-Encoder

Hyperparamter	Epochs	HS	LR	D	K
LSTM	15	350	0.003	5	5

TABLE 4.3: Hyperparameters of the LSTM Auto-Encoder

As the Target Input Summary we are using the original summary as shown above.
Example Output Predicted Summary:
['sullivan', 'regains', 'freestyle', 'landing', 'the', 'freestyle', 'landing', 'the', 'trials', 'trials', 'the', 'trials', 'trials', 'the', 'trials', 'trials', 'the', 'trials', 'trials', 'the', 'trials', 'award', 'trials', 'the', 'trials', 'award', 'trials', 'the', 'trials', 'award', 'trials', 'the', 'trials']
ROUGE Score results of the training:

Scorings	ROUGE-Score	Precision	Recall
LSTM	0.2089561739	0.2121212121	0.2058823529

TABLE 4.4: ROUGE-Score Example LSTM Auto-Encoder

Results

Average ROUGE-Score for the Train and Test Sets:
Average Loss for the Test Sets:

AVG Scorings	ROUGE-Score	Precision	Recall
A-RNN Train	0.05632	0.06834	0.04910
A-RNN Test	0.05506	0.06552	0.04875
LSTM Train	0.10872	0.11031	0.10718
LSTM Test	0.11413	0.11575	0.11255

TABLE 4.5: Average ROUGE-Scores

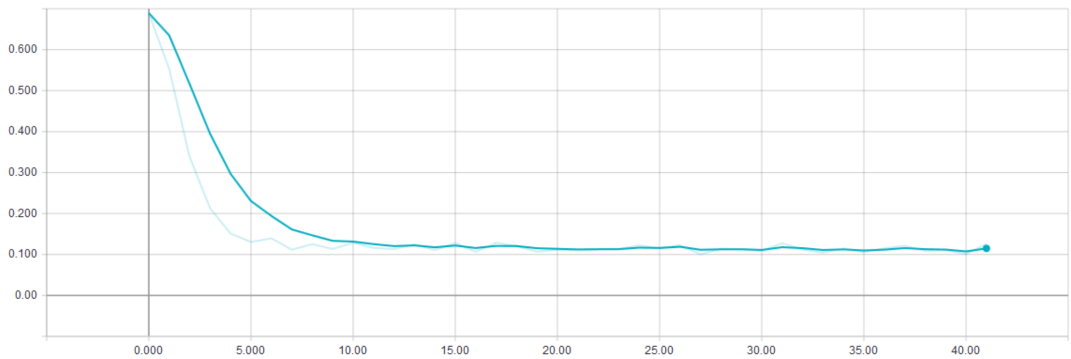


FIGURE 4.1: A-RNN training loss

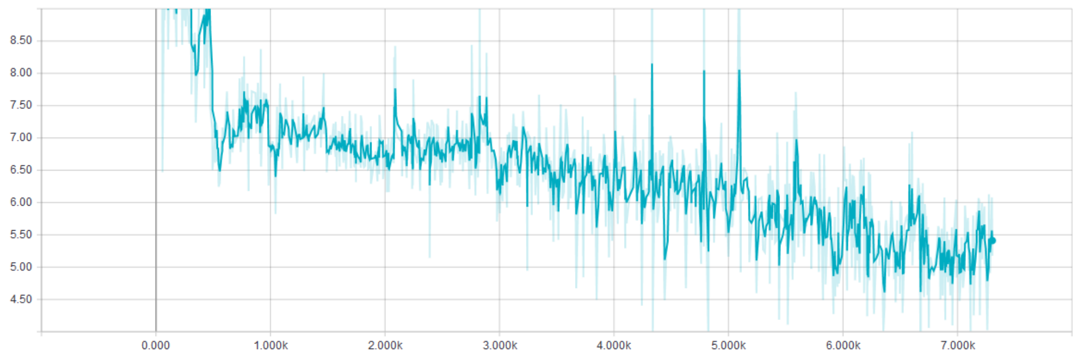


FIGURE 4.2: LSTMs training loss

	AVG Test Loss	STD Test Loss
A-RNN	0.112833333	0.003666667
LSTM	6.43125537	1,56875019

TABLE 4.6: Average & Standard deviation Test-Losses

Evaluation

The training performed achieves poor values in relation to the ROUGE score of the attentional RNN. The subsequent training of the auto-encoder also achieves only moderate values in relation to the ROUGE score.

Experiment 2

Number of training data: 1000 articles
Runtime A-RNN on Quadro P5000 GPU: 7369.768194 sec
Runtime LSTMs on Quadro P5000 GPU: 36846.83647 sec
(Training duration round about 12,25 hours)
Original Summary Example:
['sullivan', 'regains', 'the', '50meter', 'world', 'freestyle', 'record', 'in', 'sydney', 'sullivan', 'sets', 'the', 'new', 'mark', 'of', 'top', 'seconds', 'in', 'the', 'australian', 'olympic', 'trials', 'frenchman', 'alain', 'bernard', 'recorded', 'a', 'time', 'of', '2150', 'seconds', 'four', 'days', 'earlier']

Attentional RNN

Hyperparamter	Epochs	ED	HS	AS	KP	BS	Delta	LR	AD
RNN	3	100	250	50	0.8	50	0.5	0.003	50

TABLE 4.7: Hyperparameters of the Attentional RNN

Y: Example Input Target Summary:
['eamon', 'sullivan', 'regains', '50meter', 'world', 'freestyle', 'record', 'sydney', 'sullivan', 'sets', 'new', 'mark', 'top', 'seconds', 'australian', 'olympic', 'trials', 'frenchman', 'alain', 'bernard', 'recorded', 'time', '2150', 'seconds', 'four', 'days', 'earlier']
Example Output Predicted Highlights:
['sydney', 'world', 'trials', 'from', '2150', '2156', 'february', '100m', 'late', 'into', 'get', 'have', 'always', 'think', 'couple', 'competing', 'time', 'previous', 'swimming', 'you', 'heart', 'am', 'this', 'coming', 'friend']
ROUGE Score results of the training example:

Scorings	ROUGE-Score	Precision	Recall
A-RNN	0.1153855655	0.12	0.1111111111

TABLE 4.8: ROUGE-Score Example Attentional RNN

LSTM Auto-Encoder

Hyperparamter	Epochs	HS	LR	D	K
LSTM	15	350	0.003	5	5

TABLE 4.9: Hyperparameters of the LSTM Auto-Encoder

As the Target Input Summary we are using the original summary as shown above.
Example Output Predicted Summary:
['beck', 'of', 'to', 'beck', 'to', 'to', 'greater', 'beck', 'to', 'to', 'depression', 'the', 'to', 'seconds', 'depression', 'the', 'the', 'staged', 'staged', 'to', 'seconds', 'staged', 'to', 'seconds', 'staged', 'the', 'the', 'staged', 'staged', 'to', 'seconds', 'staged', 'the']
ROUGE Score results of the training:

Scorings	ROUGE-Score	Precision	Recall
LSTM	0.1492546814	0.1515151515	0.1470588235

TABLE 4.10: ROUGE-Score Example LSTM Auto-Encoder

Results

Average ROUGE-Score for the Train and Test Sets:
Average Loss for the Test Sets:

AVG Scorings	ROUGE-Score	Precision	Recall
A-RNN Train	0.06989	0.07668	0.06611
A-RNN Test	0.07622	0.08201	0.07324
LSTM Train	0.11594	0.11761	0.11432
LSTM Test	0.12367	0.12543	0.12197

TABLE 4.11: Average ROUGE-Scores

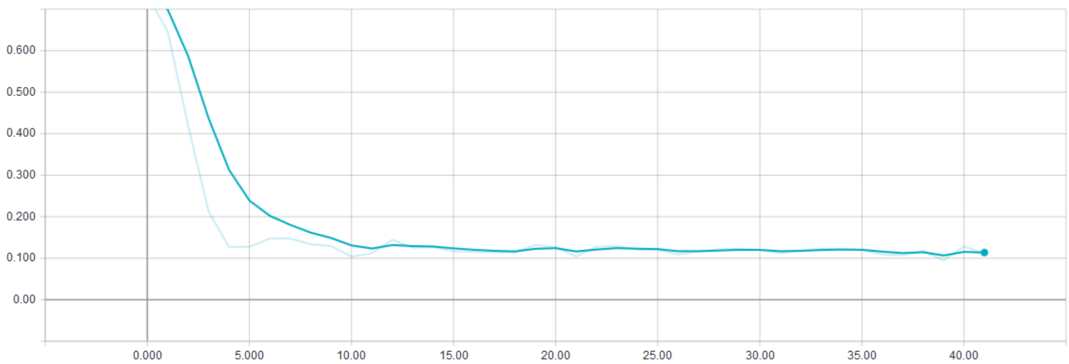


FIGURE 4.3: A-RNN training loss

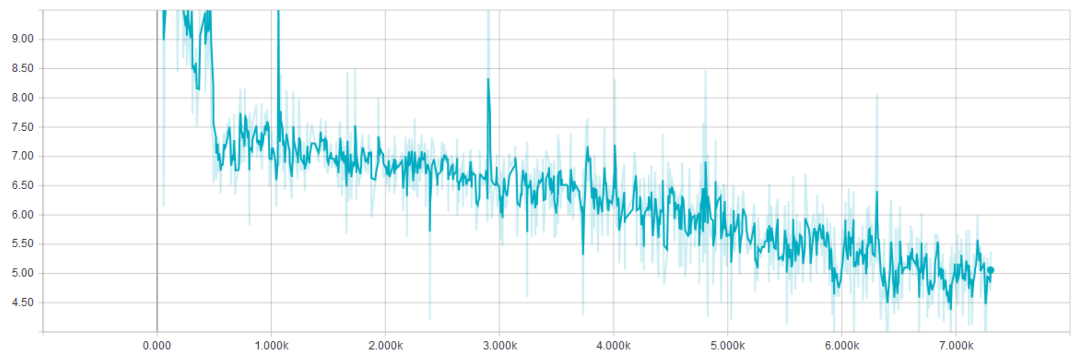


FIGURE 4.4: LSTMs training loss

	AVG Test Loss	STD Test Loss
A-RNN	0,119166667	0,003138889
LSTM	6.420761162	1,61250428

TABLE 4.12: Average & Standard deviation Test-Losses

Evaluation

Increasing the hidden size of the A-RNN from 150 to 250, the training performed achieves poor values in relation to the ROUGE score of the attentional RNN. However, there is a minimal difference to the previous experiment, which does not have to be significant. The subsequent training of the auto-encoder also achieves only moderate values in relation to the ROUGE score, similar to our first experiment.

Experiment 3

Number of training data: 1000 articles
Runtime A-RNN on a Tesla K40 GPU: 10900.46187 sec
Runtime LSTMs on a Tesla K40 GPU: 112584.8058 sec
(Training duration round about 34 hours)
Original Summary Example:
[*'united', 'states', 'believes', 'dating', 'rebels', 'shot', 'down', 'the', 'malaysia', 'air', 'jetliner', 'american', 'officials', 'say', 'russia', 'bears', 'the', 'blame', 'for', 'supplying', 'rebels', 'with', 'missiles', 'a', 'us', 'official', 'said', 'russia', 'will', 'have', 'a', 'hard', 'time', 'explaining', 'this', 'away', 'but', 'its', 'unclear', 'how', 'russias', 'vladimir', 'putin', 'will', 'respond', 'will', 'he', 'back', 'down', 'in', 'ukraine'*]

Attentional RNN

Hyperparamter	Epochs	ED	HS	AS	KP	BS	Delta	LR	AD
RNN	3	100	150	50	0.8	50	0.5	0.003	20

TABLE 4.13: Hyperparameters of the Attentional RNN

Y: Example Input Target Summary:
[*'united', 'states', 'believes', 'dating', 'rebels', 'shot', 'malaysia', 'air', 'jetliner', 'american', 'officials', 'say', 'russia', 'bears', 'blame', 'supplying', 'rebels', 'missiles', 'official', 'said', 'russia', 'will', 'hard', 'time', 'explaining', 'away', 'unclear', 'russias', 'vladimir', 'putin', 'will', 'respond', 'will', 'back', 'ukraine'*]
Example Output Predicted Highlights:
[*'washington', 'states', 'shift', 'has', 'man', 'ambassador', 'samantha', 'a', 'surfacetoair', 'fired', 'in', 'us', 'helped', 'to', 'used', 'effectively', 'from', 'means', 'operating', 'there', 'regardless', 'system', 'bears', 'its', 'equipment', 'firmly', 'establish', 'innocents', 'started', 'if', 'whether', 'on', 'agencies', 'russia', 'nationalists', 'told', 'of', 'rabbit', 'open', 'rebels', 'decision', 'peace', 'message', 'what', 'will', 'ultimately', 'harden', 'problem', 'issue', 'cnn', 'coverage'*]
ROUGE Score results of the example:

Scorings	ROUGE-Score	Precision	Recall
A-RNN	0.09302420754	0.07843137255	0.1142857143

TABLE 4.14: ROUGE-Score Example Attentional RNN

LSTM Auto-Encoder

As the Target Input Summary we are using the original summary as shown above.
Example Output Predicted Summary:

Hyperparamter	Epochs	HS	LR	D	K
LSTM	15	350	0.003	5	5

TABLE 4.15: Hyperparameters of the LSTM Auto-Encoder

$\left[\text{'hoffman', 'to', 'to', 'book', 'book', 'ignore', 'ignore', 'ignore', 'ignore', 'ignore', 'the'}\right]$

ROUGE Score results of the example:

Scorings	ROUGE-Score	Precision	Recall
LSTM	0.0396049104	0.04	0.03921568627

TABLE 4.16: ROUGE-Score Example LSTM Auto-Encoder

Results

Average ROUGE-Score for the Train and Test Sets:

Average Loss for the Test Sets:

AVG Scorings	ROUGE-Score	Precision	Recall
A-RNN Train	0.07515	0.05891	0.10931
A-RNN Test	0.06236	0.04837	0.09497
LSTM Train	0.02420	0.02450	0.02390
LSTM Test	0.03145	0.03189	0.03104

TABLE 4.17: Average ROUGE-Scores

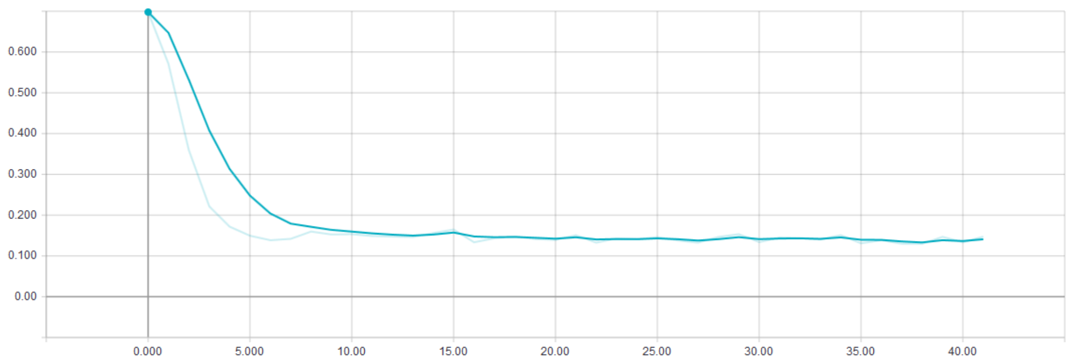


FIGURE 4.5: A-RNN training loss

Evaluation

Due the first hidden size of 150, the training performed achieves similar values in relation to the ROUGE score of the attentional RNN compared to the second experiment. Changing the Alpha-Divider to a size of 20, the subsequent training of the auto-encoder achieves worse values in relation to the ROUGE score.

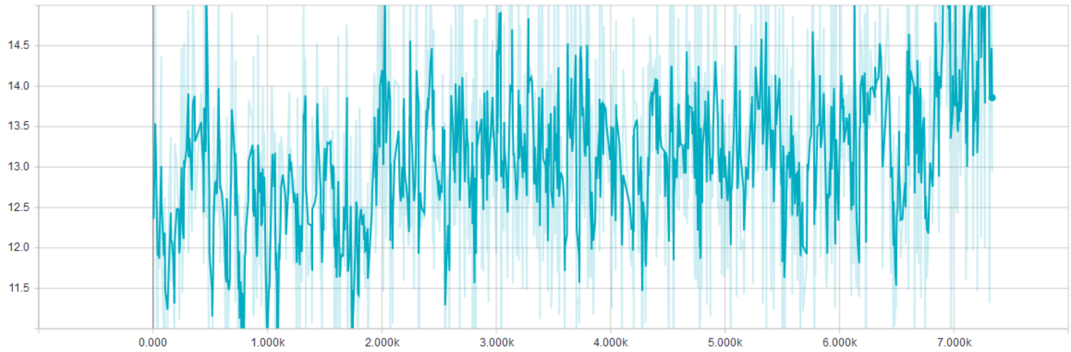


FIGURE 4.6: LSTMs training loss

	AVG Test Loss	STD Test Loss
A-RNN	0.127761111	0.002975
LSTM	13.2251684	2.81065529

TABLE 4.18: Average & Standard deviation Test-Losses

Experiment 4

Number of training data: 1000 articles
Runtime A-RNN on Quadro P5000 GPU: 7205.767463 sec
Runtime LSTMs on Quadro P5000 GPU: 39562.51708 sec
(Training duration round about 13 hours)
Original Summary Example:
['sullivan', 'regains', 'the', '50meter', 'world', 'freestyle', 'record', 'in', 'sydney', 'sullivan', 'sets', 'the', 'new', 'mark', 'of', 'top', 'seconds', 'in', 'the', 'australian', 'olympic', 'trials', 'frenchman', 'alain', 'bernard', 'recorded', 'a', 'time', 'of', '2150', 'seconds', 'four', 'days', 'earlier']

Attentional RNN

Hyperparamter	Epochs	ED	HS	AS	KP	BS	Delta	LR	AD
RNN	3	100	250	50	0.8	50	0.5	0.003	50

TABLE 4.19: Hyperparameters of the Attentional RNN

Y: Example Input Target Summary:
['eamon', 'sullivan', 'regains', '50meter', 'world', 'freestyle', 'record', 'sydney', 'sullivan', 'sets', 'new', 'mark', 'top', 'seconds', 'australian', 'olympic', 'trials', 'frenchman', 'alain', 'bernard', 'recorded', 'time', '2150', 'seconds', 'four', 'days', 'earlier']
Example Output Predicted Highlights:
['sydney', 'to', 'relaxed', 'last', 'for', 'get', 'chance', 'hope', 'well', '600', 'libby', 'up', 'come', 'an', 'journey', 'but']
ROUGE Score results of the training example:

Scorings	ROUGE-Score	Precision	Recall
A-RNN	0.04651258118	0.0625	0.03703703704

TABLE 4.20: ROUGE-Score Example Attentional RNN

LSTM Auto-Encoder

Hyperparamter	Epochs	HS	LR	D	K
LSTM	30	350	0.003	5	5

TABLE 4.21: Hyperparameters of the LSTM Auto-Encoder

As the Target Input Summary we are using the original summary as shown above.
Example Output Predicted Summary:
['sullivan', 'regains', 'the', 'unk', 'world', 'freestyle', 'record', 'world', 'sullivan', 'sullivan', 'mark', 'mark', 'mark', 'mark', 'mark', 'mark', 'mark', 'mark', 'mark', 'trials', 'trials', 'trials', 'trials', 'trials', 'trials', 'trials', 'trials', 'world', 'trials', 'trials', 'trials', 'world', 'trials']
ROUGE Score results of the training:

Scorings	ROUGE-Score	Precision	Recall
LSTM	0.2985084127	0.303030303	0.2941176471

TABLE 4.22: ROUGE-Score Example LSTM Auto-Encoder

Results

Average ROUGE-Score for the Train and Test Sets:
Average Loss for the Test Sets:

AVG Scorings	ROUGE-Score	Precision	Recall
A-RNN Train	0.05251	0.08170	0.04229
A-RNN Test	0.06018	0.08587	0.05021
LSTM Train	0.25276	0.25655	0.24909
LSTM Test	0.25850	0.26222	0.25489

TABLE 4.23: Average ROUGE-Scores

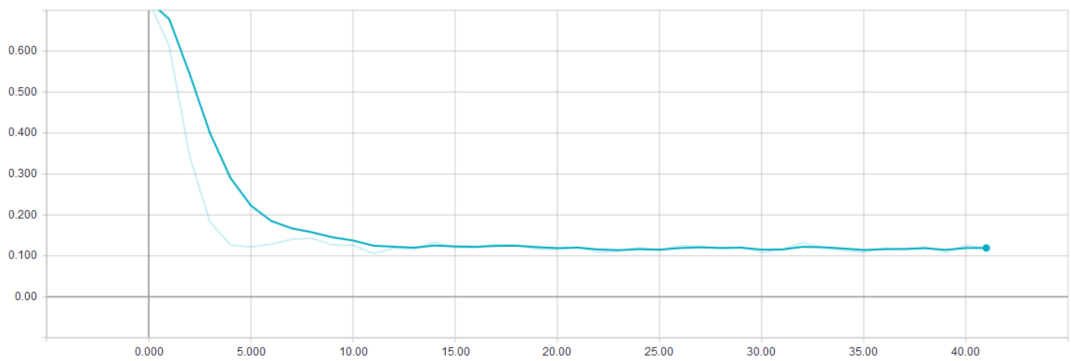


FIGURE 4.7: A-RNN training loss

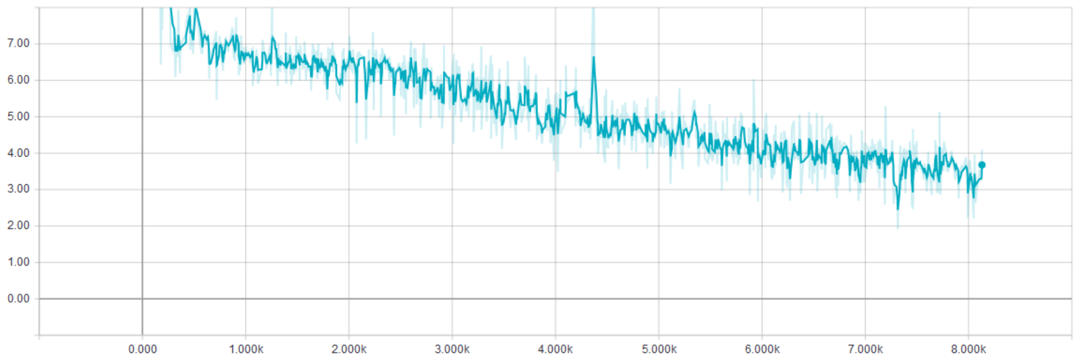


FIGURE 4.8: LSTMs training loss

	AVG Test Loss	STD Test Loss
A-RNN	0.11922697	0.001001138
LSTM	5.25055838	1.45382965

TABLE 4.24: Average & Standard deviation Test-Losses

Evaluation

The training performed achieves poor values in relation to the ROUGE score of the attentional RNN like the experiments before. The subsequent training of the auto-encoder achieves twice as good results than previous experiments in relation to the ROUGE score, by redoubling the number of epochs to 30.

Experiment 5

Number of training data: 1000 articles
Runtime A-RNN on Quadro P5000 GPU: 7233.047348 sec
Runtime LSTMs on Quadro P5000 GPU: 69985.15334 sec
(Training duration round about 21 hours)
Original Summary Example:
['sullivan', 'regains', 'the', '50meter', 'world', 'freestyle', 'record', 'in', 'sydney', 'sullivan', 'sets', 'the', 'new', 'mark', 'of', 'top', 'seconds', 'in', 'the', 'australian', 'olympic', 'trials', 'frenchman', 'alain', 'bernard', 'recorded', 'a', 'time', 'of', '2150', 'seconds', 'four', 'days', 'earlier']

Attentional RNN

Hyperparamter	Epochs	ED	HS	AS	KP	BS	Delta	LR	AD
RNN	3	100	400	50	0.8	50	0.5	0.003	50

TABLE 4.25: Hyperparameters of the Attentional RNN

Y: Example Input Target Summary:
['eamon', 'sullivan', 'regains', '50meter', 'world', 'freestyle', 'record', 'sydney', 'sullivan', 'sets', 'new', 'mark', 'top', 'seconds', 'australian', 'olympic', 'trials', 'frenchman', 'alain', 'bernard', 'recorded', 'time', '2150', 'seconds', 'four', 'days', 'earlier']
Example Output Predicted Highlights:
['regained', 'australian', 'had', 'back', 'recorded', 'european', 'the', 'late', 'last', 'hundredths',

'just', 'olympics', '5288', 'break', 'are', 'a', 'be', 'team', 'email']
ROUGE Score results of the training example:

Scorings	ROUGE-Score	Precision	Recall
A-RNN	0.08695747325	0.1052631579	0.07407407407

TABLE 4.26: ROUGE-Score Example Attentional RNN

LSTM Auto-Encoder

Hyperparamter	Epochs	HS	LR	D	K
LSTM	30	350	0.003	5	5

TABLE 4.27: Hyperparameters of the LSTM Auto-Encoder

As the Target Input Summary we are using the original summary as shown above.
Example Output Predicted Summary:
[*'sullivan', 'regains', 'the', 'unk', 'brief', 'freestyle', 'unk', 'a', 'a', 'sullivan', 'a', 'sets', 'a', 'sets', 'a', 'sets', 'a', 'sets', 'a', 'a', 'recorded', 'recorded', 'a', 'recorded', 'recorded', 'a', 'recorded', 'recorded', 'a', 'recorded', 'recorded', 'a', 'recorded'*]
ROUGE Score results of the training:

Scorings	ROUGE-Score	Precision	Recall
LSTM	0.2686576664	0.2727272727	0.2647058824

TABLE 4.28: ROUGE-Score Example LSTM Auto-Encoder

Results

Average ROUGE-Score for the Train and Test Sets:
Average Loss for the Test Sets:

AVG Scorings	ROUGE-Score	Precision	Recall
A-RNN Train	0.04877	0.06641	0.03963
A-RNN Test	0.05319	0.06890	0.04468
LSTM Train	0.23199	0.23540	0.22869
LSTM Test	0.23004	0.23332	0.22686

TABLE 4.29: Average ROUGE-Scores

	AVG Test Loss	STD Test Loss
A-RNN	0.1263089834	0.002821918996
LSTM	5.247702599	1.449566126

TABLE 4.30: Average & Standard deviation Test-Losses

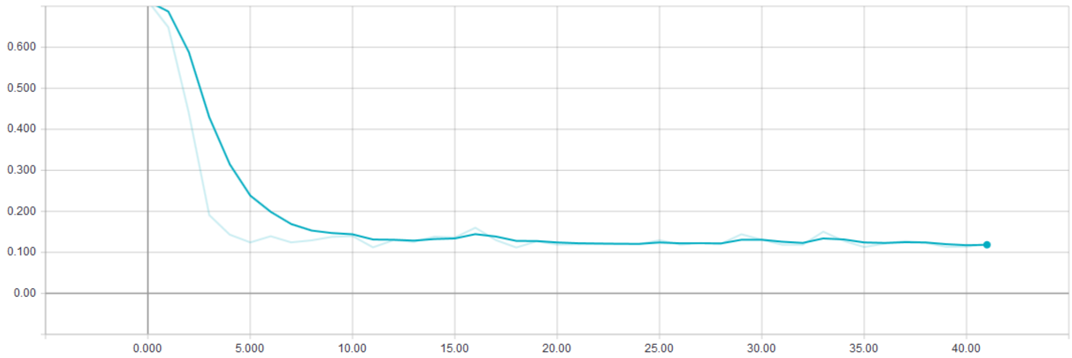


FIGURE 4.9: A-RNN training loss

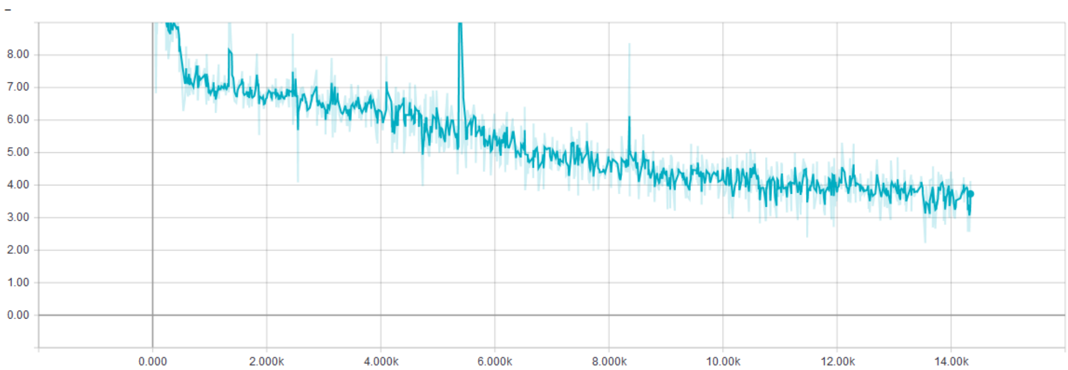


FIGURE 4.10: LSTMs training loss

Evaluation

Increasing the hidden size of the A-RNN from 400, the training performed achieves poor values in relation to the ROUGE score of the attentional RNN. The subsequent training of the auto-encoder also achieves better values in relation to the ROUGE score, similar to our previous experiment. However, there is a minimal difference to the previous experiment, which does not have to be significant.

Experiment 6

Number of training data: 1000 articles
Runtime A-RNN on a Tesla K40 GPU: 9766.658195 sec
Runtime LSTMs on a Tesla K40 GPU: 112584.8058 sec
(Training duration round about 34 hours)
Original Summary Example:
['union', 'unite', 'says', 'it', 'is', 'taking', 'british', 'airways', 'to', 'court', 'over', 'working', 'changes', 'ba', 'wants', 'to', 'impose', 'is', 'a', 'reduction', 'in', 'the', 'number', 'of', 'crew', 'members', 'on', 'flights', 'ba', 'plans', 'to', 'impose', 'the', 'changes', 'starting', 'november', '16', 'according', 'to', 'unite']

Attentional RNN

Y: Example Input Target Summary:
['union', 'unite', 'says', 'taking', 'british', 'airways', 'court', 'working', 'changes', 'ba', 'wants', 'impose', 'reduction', 'number', 'crew', 'members', 'flights', 'ba', 'plans', 'impose', 'changes', 'starting', 'november', '16', 'according', 'unite']

Hyperparamter	Epochs	ED	HS	AS	KP	BS	Delta	LR	AD
RNN	6	100	250	50	0.8	50	0.5	0.003	50

TABLE 4.31: Hyperparameters of the Attentional RNN

Example Output Predicted Highlights:
['london', 'england', 'try', 'changes', 'the', 'court', 'british', 'was', 'of', 'ballot', 'holiday', 'spokeswoman']
ROUGE Score results of the example:

Scorings	ROUGE-Score	Precision	Recall
A-RNN	0.05263253573	0.08333333333	0.03846153846

TABLE 4.32: ROUGE-Score Example Attentional RNN

LSTM Auto-Encoder

Hyperparamter	Epochs	HS	LR	D	K
LSTM	15	350	0.003	5	5

TABLE 4.33: Hyperparameters of the LSTM Auto-Encoder

As the Target Input Summary we are using the original summary as shown above.
Example Output Predicted Summary:
['unite', 'says', 'it', 'to', 'to', 'to', 'to', 'to', 'taking', 'to', 'impose', 'impose', 'to', 'to', 'to', 'impose', 'to']
ROUGE Score results of the example:

Scorings	ROUGE-Score	Precision	Recall
LSTM	0.2337671838	0.2368421053	0.2307692308

TABLE 4.34: ROUGE-Score Example LSTM Auto-Encoder

Results

Average ROUGE-Score for the Train and Test Sets:
Average Loss for the Test Sets:

AVG Scorings	ROUGE-Score	Precision	Recall
A-RNN Train	0.03842	0.07984	0.02721
A-RNN Test	0.03094	0.06735	0.02254
LSTM Train	0.20611	0.20912	0.20318
LSTM Test	0.23127	0.23550	0.22721

TABLE 4.35: Average ROUGE-Scores

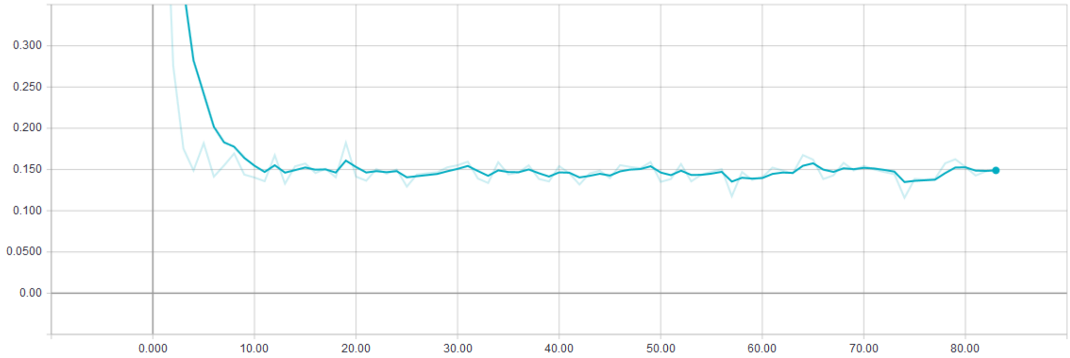


FIGURE 4.11: A-RNN training loss

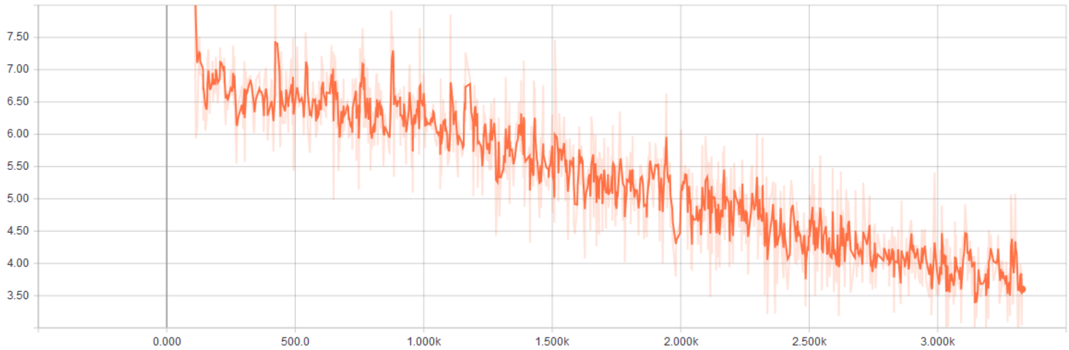


FIGURE 4.12: LSTMs training loss

	AVG Test Loss	STD Test Loss
A-RNN	0.130819	0.0013102
LSTM	5.515734	1.466071

TABLE 4.36: Average & Standard deviation Test-Losses

Evaluation

Based on the previous experiments, the hidden size of the attentional RNN seems to be good enough for a size of 250. Increasing the amount of epochs of the A-RNN to 6, leads to less accurate results in relation to the ROUGE score of the attentional RNN. (May be because of the Mini-batch training) The subsequent training of the auto-encoder also achieves better values in relation to the ROUGE score, similar to our previous experiment. However, there is a minimal difference to the previous experiment, which does not have to be significant.

Experiment 7

Number of training data: 1000 articles
Runtime A-RNN on a Tesla K40 GPU: 9587.092386 sec
Runtime LSTMs on a Tesla K40 GPU: 211915.5298 sec
(Training duration round about 61 hours)
Original Summary Example:
['new','panetta', 'says', 'service', 'members', 'must', 'meet', 'the', 'highest', 'standards', 'of', 'conduct', 'new', 'grassley', 'questions', 'white', 'house', 'counsels', 'review', 'the', 'prostitution', 'scandal', 'has', 'financial', 'the', 'secret', 'service', 'and', 'pentagon', 'the', 'white',

'house', 'says', 'there', 'is', 'no', 'indication', 'any', 'of', 'its', 'staff', 'members', 'were', 'involved']

Attentional RNN

Hyperparamter	Epochs	ED	HS	AS	KP	BS	Delta	LR	AD
RNN	3	100	250	100	0.8	50	0.5	0.003	50

TABLE 4.37: Hyperparameters of the Attentional RNN

Y: Example Input Target Summary:
[*'new', 'panetta', 'says', 'service', 'members', 'meet', 'highest', 'standards', 'conduct', 'new', 'grassley', 'questions', 'white', 'house', 'counsels', 'review', 'prostitution', 'scandal', 'financial', 'secret', 'service', 'pentagon', 'white', 'house', 'says', 'indication', 'staff', 'members', 'involved'*]
Example Output Predicted Highlights:
[*'washington', 'his', 'while', 'who', 'have', 'country', 'during', 'thats', 'sullivan', 'were', 'king', 'york', 'helped', 'transport', 'sparked', 'took', 'engaging', 'conduct', 'personnel', 'services', 'report'*]
ROUGE Score results of the example:

Scorings	ROUGE-Score	Precision	Recall
A-RNN	0.04761904762	0.03448275862	0.04000095128

TABLE 4.38: ROUGE-Score Example Attentional RNN

LSTM Auto-Encoder

Hyperparamter	Epochs	HS	LR	D	K
LSTM	15	350	0.003	5	5

TABLE 4.39: Hyperparameters of the LSTM Auto-Encoder

As the Target Input Summary we are using the original summary as shown above.
Example Output Predicted Summary:
[*'panetta', 'the', 'service', 'meet', 'meet', 'meet', 'meet', 'meet', 'meet', 'the', 'the', 'service', 'the', 'the', 'grassley', 'the', 'the', 'the', 'the', 'the', 'the', 'the', 'the', 'the', 'the', 'indication', 'the', 'the', 'the', 'the', 'the', 'the', 'the', 'the', 'the', 'the', 'the', 'the', 'the', 'the', 'the', 'the'*]
ROUGE Score results of the example:

Scorings	ROUGE-Score	Precision	Recall
LSTM	0.2117656559	0.2142857143	0.2093023256

TABLE 4.40: ROUGE-Score Example LSTM Auto-Encoder

Results

Average ROUGE-Score for the Train and Test Sets:
Average Loss for the Test Sets:

AVG Scorings	ROUGE-Score	Precision	Recall
A-RNN Train	0.06530	0.07509	0.06096
A-RNN Test	0.05963	0.06776	0.05695
LSTM Train	0.20929	0.21213	0.20652
LSTM Test	0.21933	0.22265	0.21612

TABLE 4.41: Average ROUGE-Scores

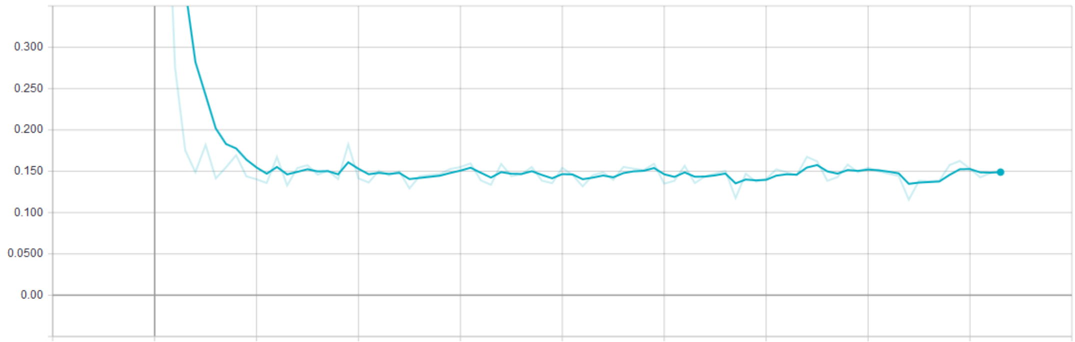


FIGURE 4.13: A-RNN training loss

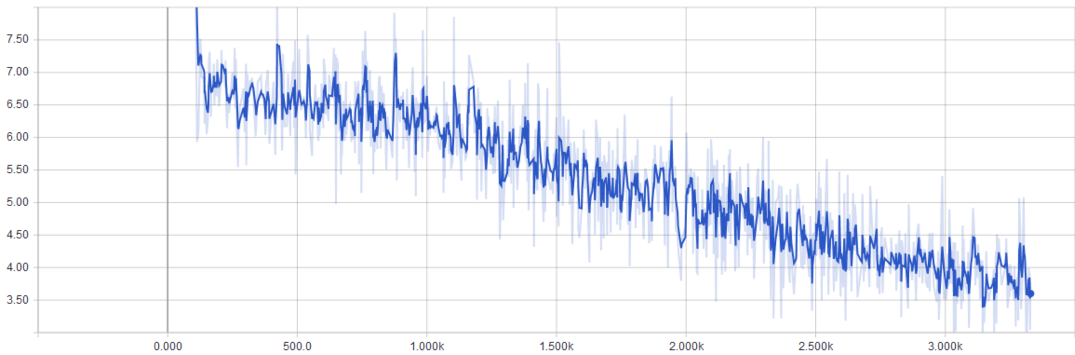


FIGURE 4.14: LSTMs training loss

	AVG Test Loss	STD Test Loss
A-RNN	0.1315018	0.0014638
LSTM	5.3854823	1.3980658

TABLE 4.42: Average & Standard deviation Test-Losses

Evaluation

Increasing the size of attention of the A-RNN to 100, leads to no significant changed results in relation to the ROUGE score of the attentional RNN. The subsequent training of the auto-encoder also achieves same better values in relation to the ROUGE score, similar to our previous experiment.

Experiment 8

Number of training data: 1000 articles
Runtime LSTMs on Quadro P5000 GPU: 161712.2759 sec
(Training duration round about 44 hours)

To prove the performance of our model, we need to check, wether the LSTM auto-encoder, which is responsible for creating the summaries, is getting better results by generating summaries without the previous training of the attentional RNN. If so, our extended model architecture is no improvement compared to a single auto-encoder architecture. We re-trained the last experiment of the model with the auto encoder separately.

Original Summary Example:
['communist', 'party', 'of', 'nepal', 'chairman', 'won', '464', 'out', 'of', '577', 'votes', 'a', 'simple', 'majority', 'was', 'enough', 'to', 'be', 'elected', 'the', 'prime', 'minister', 'assembly', 'declared', 'nepal', 'a', 'republic', 'in', 'may', 'and', 'july', 'elected', 'first', 'president', 'the', 'post', 'of', 'president', 'is', 'largely', 'ceremonial', 'pm', 'has', 'executive', 'powers']

LSTM Auto-Encoder

Hyperparamter	Epochs	HS	LR	D	K
LSTM	30	350	0.003	5	5

TABLE 4.43: Hyperparameters of the LSTM Auto-Encoder

As the Target Input Summary we are using the original summary as shown above.
The input of the LSTM auto-encoder is the whole article exactly as usual with the attentional RNN. **Example Output Predicted Summary:**
['zealand', 'radio', 'radio', 'radio', 'in']
ROUGE Score results of the training:

Scorings	ROUGE-Score	Precision	Recall
LSTM	0.02247286012	0.02272727273	0.02222222222

TABLE 4.44: ROUGE-Score Example LSTM Auto-Encoder

Results

Average ROUGE-Score for the Train and Test Sets:
Average Loss for the Test Sets:

AVG Scorings	ROUGE-Score	Precision	Recall
LSTM Train	0.01495	0.01516	0.01475
LSTM Test	0.01318	0.01337	0.012995

TABLE 4.45: Average ROUGE-Scores

	AVG Test Loss	STD Test Loss
LSTM	14.1278286	1.894893765

TABLE 4.46: Average & Standard deviation Test-Losses

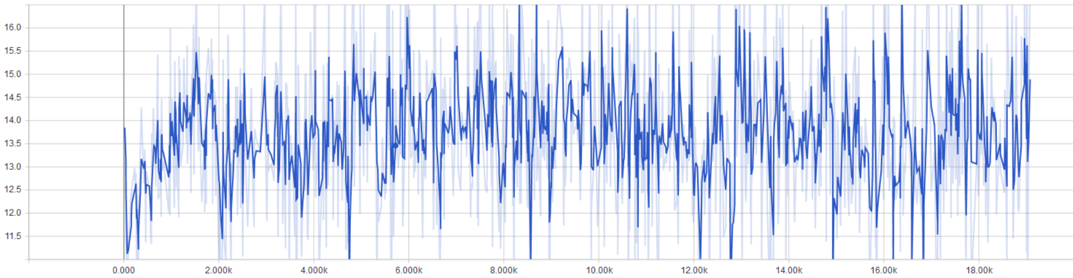


FIGURE 4.15: LSTMs training loss

Evaluation

As we can see, the loss of the LSTMs is much higher in the separated training of the auto-encoder than for the whole architecture training. Also, the average ROUGE-Score is getting much worse results. The results therefore prove that the structure of the presented architecture of this work already achieves better results on a few data compared to a simple auto-encoder architecture(Exp.8).

Chapter 5

Summary

5.1 Summary

The work gave a good insight into the application of neural networks, especially in the field of word processing. With a minimal lead, the 7th experiment promises the best results and could thus provide a direction for further experiments. From the previous experiments it became apparent that e.g. the hidden size of the A-RNN has no strong effect on the final result and a size of 250 is sufficient. Comparing the 7th experiment with the first one, it seems like the attention size also do not affect the results when increasing. It is striking, that increasing the epochs in the A-RNN reduce performance. It can also be assumed that increasing the epochs in the auto-encoder leads to a better performance on its part. The same applies to the hidden-size of the auto-encoder. In summary, it can be said that the model of this work has unfortunately not yet achieved perfect results, but it should not be completely excluded for future experiments. It also shows certain tendencies for a few input data and thus has a certain potential to be improved by adjustments.

5.2 Future work

For future work it would probably be an improvement to pre-train the auto-encoder. This would require the input of grammatically correct sentences so that the model can generate better sentences. The idea behind it is that it can already learn common sentence structures through pre-training. On the basis of this work, the experimental work can be continued with larger computing resources, so that all hyperparameters can be fully investigated. The course of the results would be interesting for the training of several data or data sets in order to obtain a strong development trend. It would be interesting to investigate how the extended adaptation of experiments and the increase in the data used can change and in the best case even improve the performance of the architecture and what maximum performance the model can achieve.

Bibliography

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *CoRR* abs/1409.0473 (2014).

[2] Paul Balzer. *Motorblog » [Tutorial] Neuronale Netze einfach erklärt*. http://www.cbcity.de/tutorial-neuronale-netze-einfach-erklaert. (Accessed on 03/17/2018). Mar. 2016.

[3] Fabian Beck. *Neuronale Netze - Eine Einführung - Rekurrente Netze*. http://www.neuronalesnetz.de/rekurrente.html. (Accessed on 03/17/2018).

[4] Denny Britz. *Deep Learning Glossary – WildML*. http://www.wildml.com/deep-learning-glossary/#LSTM. (Accessed on 03/17/2018).

[5] Jishnu Ray Chowdhury. *GitHub - JRC1995/Abstractive-Summarization: Implementation of abstractive summarization using LSTM with Residual Recurrent Attention in the encoder-decoder architecture with local attention*. https://github.com/JRC1995/Abstractive-Summarization. (Accessed on 03/18/2018).

[6] S. Hochreiter et al. "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies". In: *A Field Guide to Dynamical Recurrent Neural Networks*. Ed. by Kremer and Kolen. IEEE Press, 2001.

[7] Christopher D. Manning Jeffrey Pennington Richard Socher. *GloVe: Global Vectors for Word Representation*. https://nlp.stanford.edu/projects/glove/. (Accessed on 03/24/2018).

[8] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980 (2014). arXiv: 1412.6980. URL: http://arxiv.org/abs/1412.6980.

[9] Thang Luong, Hieu Pham, and Christopher D. Manning. "Effective Approaches to Attention-based Neural Machine Translation". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*. 2015, pp. 1412–1421. URL: http://aclweb.org/anthology/D/D15/D15-1166.pdf.

[10] Tomas Mikolov et al. "Efficient Estimation of Word Representations in Vector Space". In: *CoRR* abs/1301.3781 (2013). arXiv: 1301.3781. URL: http://arxiv.org/abs/1301.3781.

[11] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.

[12] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "Glove: Global Vectors for Word Representation". In: *EMNLP*. 2014.

[13] David M. W. Powers. "Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation". In: 2008.

[14] Sean Robertson. *practical-pytorch/seq2seq-translation.ipynb at master · spro/practical-pytorch · GitHub*. https://github.com/spro/practical-pytorch/blob/master/seq2seq-translation/seq2seq-translation.ipynb. (Accessed on 03/27/2018).

[15] Jürgen Schmidhuber. "Deep Learning in Neural Networks: An Overview". In: *CoRR* abs/1404.7828 (2014).

[16] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. "Sequence to Sequence Learning with Neural Networks". In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani et al. Curran Associates, Inc., 2014, pp. 3104–3112. URL: http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf.

[17] Tensorflow. *tf.nn.sigmoid_cross_entropy_with_logits | TensorFlow*. https://www.tensorflow.org/api_docs/python/tf/nn/sigmoid_cross_entropy_with_logits. (Accessed on 03/27/2018).

[18] Rui Zhao Thang Luong Eugene Brevdo. *Neural Machine Translation (seq2seq) Tutorial | TensorFlow*. https://www.tensorflow.org/tutorials/seq2seq. (Accessed on 03/17/2018). Jan. 2018.

[19] Cheng Wang. "RRA: Recurrent Residual Attention for Sequence Learning". In: *CoRR* abs/1709.03714 (2017). arXiv: 1709.03714. URL: http://arxiv.org/abs/1709.03714.

[20] Zichao Yang et al. "Hierarchical Attention Networks for Document Classification". In: *HLT-NAACL*. 2016.

[21] Pengcheng YIN. *GitHub - pcyin/PyRouge: A python library to compute rouge score for summarization*. https://github.com/pcyin/PyRouge. (Accessed on 03/25/2018).