TECHNISCHE HOCHSCHULE NÜRNBERG
GEORG SIMON OHM

Faculty of Computer Science

IT-based text generation using NLP methods

State of the art and design of a prototype

Bachelor Thesis in
Business Information Systems and Management

by

Tim Löhr

Student ID 3060802

First advisor:	Prof. Dr. Alfred Holl
Second advisor:	Prof. Dr. Florian Gallwitz

Hinweis: Diese Erklärung ist in alle Exemplare der Abschlussarbeit fest einzubinden. (Keine Spiralbindung)

TECHNISCHE HOCHSCHULE NÜRNBERG
GEORG SIMON OHM

Prüfungsrechtliche Erklärung der/des Studierenden

Angaben des bzw. der Studierenden:

Name: Vorname: Matrikel-Nr.:

Fakultät: Studiengang:

Semester:

Titel der Abschlussarbeit:

Ich versichere, dass ich die Arbeit selbständig verfasst, nicht anderweitig für Prüfungszwecke vorgelegt, alle benutzten Quellen und Hilfsmittel angegeben sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Ort, Datum, Unterschrift Studierende/Studierender

Erklärung zur Veröffentlichung der vorstehend bezeichneten Abschlussarbeit

Die Entscheidung über die vollständige oder auszugsweise Veröffentlichung der Abschlussarbeit liegt grundsätzlich erst einmal allein in der Zuständigkeit der/des studentischen Verfasserin/Verfassers. Nach dem Urheberrechtsgesetz (UrhG) erwirbt die Verfasserin/der Verfasser einer Abschlussarbeit mit Anfertigung ihrer/seiner Arbeit das alleinige Urheberrecht und grundsätzlich auch die hieraus resultierenden Nutzungsrechte wie z.B. Erstveröffentlichung (§ 12 UrhG), Verbreitung (§ 17 UrhG), Vervielfältigung (§ 16 UrhG), Online-Nutzung usw., also alle Rechte, die die nicht-kommerzielle oder kommerzielle Verwertung betreffen.

Die Hochschule und deren Beschäftigte werden Abschlussarbeiten oder Teile davon nicht ohne Zustimmung der/des studentischen Verfasserin/Verfassers veröffentlichen, insbesondere nicht öffentlich zugänglich in die Bibliothek der Hochschule einstellen.

Hiermit ☐ genehmige ich, wenn und soweit keine entgegenstehenden Vereinbarungen mit Dritten getroffen worden sind,

☐ genehmige ich nicht,

dass die oben genannte Abschlussarbeit durch die Technische Hochschule Nürnberg Georg Simon Ohm, ggf. nach Ablauf einer mittels eines auf der Abschlussarbeit aufgebrachten Sperrvermerks kenntlich gemachten Sperrfrist

von Jahren (0 - 5 Jahren ab Datum der Abgabe der Arbeit),

der Öffentlichkeit zugänglich gemacht wird. Im Falle der Genehmigung erfolgt diese unwiderruflich; hierzu wird der Abschlussarbeit ein Exemplar im digitalisierten PDF-Format auf einem Datenträger beigefügt. Bestimmungen der jeweils geltenden Studien- und Prüfungsordnung über Art und Umfang der im Rahmen der Arbeit abzugebenden Exemplare und Materialien werden hierdurch nicht berührt.

Ort, Datum, Unterschrift Studierende/Studierender

Preface I

The following thesis was created during the seventh and last semester at the Georg Simon Ohm University of Applied Science. Within the last three semesters, I realized that my major interest among all IT related topics is artificial intelligence.

My personal interest started basically with a group IT-project, in which my team and I programmed an autonomously driving remote control car with a deep neural network together with a Raspberry Pi 3B+. From this first project on, I selected all my further elective courses to be related to machine learning or data science in any possible way. I wanted to increase my knowledge further, so I searched for a website that provides courses related to AI. I found *www.udacity.com*, which offers courses in cooperation with top IT companies, such as Google, Airbnb, or Microsoft. Out of curiosity, I bought the course *Natural Language Processing*. After successfully finishing it, I was encouraged to write my bachelor thesis in a *Natural Language Processing* related topic. Together with my professor *Prof. Dr. Alfred Holl*, I worked out a structured methodological table for the entire structure of this paper. Even though Natural Language Processing is just a subfield of machine learning, the current state-of-the-art research is far beyond what I can research within a bachelor thesis. I decided to write my thesis about the subfield *textgeneration* within NLP. My state-of-the-art research includes all *hot topics* within NLP, and my prototype focuses only on the text generation part, to dive deeper into what NLP and especially text generation can accomplish in the year 2020.

Preface II

For my research, I encountered a lot of old and recently published papers, mostly from *https://arxiv.org/*. To read through the papers requires a lot of prior knowledge, especially in mathematics, which I learned during my semester in Hong Kong at the City University of Hong Kong. To fully understand the mathematics given in this thesis, enhanced knowledge of calculus and linear algebra is required. Even if this is not the case, I will describe the process in such a way that it can be comprehended without looking at the maths.
Machine Learning and, more specifically, NLP is not an intuitive study. I provided for the matrix notations the common terminologies originated from top researchers and tried to make the entry into this field as smooth as possible if the reader has no prior knowledge about this topic. During the five-month development process of the bachelor thesis, I gained much knowledge. I recognized that NLP is a huge topic, constantly under research. To keep up to date with the latest publications requires much effort.
To give a full state-of-the-art review about *all* NLP related disciplines is not possible within this thesis. For this reason, I focus entirely on the development of the *Neural Text Generation* (NTP), which includes more fields than the reader might imagine.

	Titel / Kapitel		Untertitel / Unterkapitel	——Woher?—— Wissensinput	——Woher?—— Frageinput	——Wie?—— Methode	——Was?—— Zielbeschreibung
0	**IT-basierte Textgenerierung mit Hilfe von NLP-Methoden** **State of the Art & Entwurf eines Prototypen**			Allgemeingültig: • Fachbücher, Bücher • HongKong, TH-OHM • Online Kurse	1. Was ist der State of Art von NLP - Systemen. 2. In welcher Qualität kann ich den Textgenerierungs- Prototypen selbst programmieren und welche Güte hat dieser?	1. Darstellung des State of the Art der NLP-Systeme. 2. Studium der relevanten Aspekte des NLP und Programmierung eines IT-basierten Textgenerierungs-Prototypen.	1. State of the Art fachlich herausarbeiten. 2. Einen Prototypischen Algorithmus programmieren, der zu einem gegeben Input z.B. ein Buch immer wieder neue kreative Fortsetzungen generiert.
1	**Einleitung**	**1.1**	**Fallbeispiel eines aktuellen NLP-Systems**	• [0.1] • Wissenschaftliches Schreiben und	1. Was sind aktuelle, nützliche Einsatzgebiete von NLP-Textverarbeitungs-Systemen? 2. Was ist der Nutzen meines NLP-Prototypen im Bereich der Textverarbeitung?	1. Recherche über die aktuellen und geplanten NLP-Systeme, im Bereich der Textverarbeitung. 2. Vorstellung meines Beitrags zu NLP-Systemen mithilfe meines Prototyps.	1. Antwort auf die Frage, warum meine Bachelorarbeit sinnvoll ist und welche Motivation ich habe zur Bearbeitung 2. Erläuterung durch einen interessanten leichten Einstieg.
2	**State of the Art**	**2.1**	**Relevante Aspekte der Mathematik**	[1.1]	Welches mathematische „know-how“ ist notwendig, um NLP-Systeme für Textverarbeitung und meinen Prototypen technisch verstehen zu können?	Recherche nach den relevanten Aspekten der Mathematik für dieses Thema.	Beschreibung der anwendungsbezogenen mathematischen Modelle für diesen Themenkomplex anhand von Formeln und Erklärungen.
		2.2	**Geschichte des NLP**	• [0] • [0.1]	1. Seit wann wird an NLP-Systemen geforscht? 2. Ab welchem Punkt konnte man effektiven Nutzen aus diesen Systemen ziehen?	1. Literaturrecherche über die Geschichte des NLP (40 Jahre). 2. Literaturrecherche über die ersten Einsätze der NLP-Systeme.	1. Darstellung der Geschichte des NLP in Form einer zeitlichen Abfolge. 2. Nutzen der ersten NLP-Prototypen oder Technologien die im Einsatz waren.
		2.3	**Aktuelle Trends der Technologie**	• [0] • [0.1] • [2.2] • Fallbeispiele	1. Was sind aktuelle NLP-Systeme imstande zu leisten? 2. Wo sind die Einsatzgebiete?	1. Literaturrecherche über aktuelle Trends (+ - 5 Jahre). 2. Recherche von aktuelle Papern und Veröffentlichungen.	1. Darstellung der aktuellen Technologien. 2. Blick in die kurzfristige Zukunft anhand von aktuellen Fallbeispielen und Forschungsergebnissen.
3	**Prototyp**	**3.1**	**Zielsetzung / Anforderungen**	• [0] • [1] • [2]	1. Was soll mein Prototyp mit gegebenen Mitteln leisten können? 2. Welcher Output ist im besten Fall zu erwarten?	1. Requirements Engineering. 2. Klassifizierung und Analyse möglicher Ergebnisse, z.B. ob der Output grammatikalisch korrekt ist.	1. Erläuterung des Umfangs meines Prototyps. 2. Sammlung und Klassifizierung der Anforderungen an den Algorithmus und dessen Output.
		3.2	**Fachkonzept**	[3]	1. Wie ist mein Prototyp strukturiert? 2. Welche Algorithmen verwende ich? 3. Welche Prozesse durchlaufen die zu verarbeitenden Daten? 4. Wie werden die Daten verarbeitet?	1. Erstellen eines Fachkonzepts 2. Algorithmus modellieren 3. Prozessmodellierung 4. Datenflussmodellierung und, oder Datenmodellierung	1. Fachkonzept fertig erstellt. 2. Der Prototyp wird ohne IT Bezug anhand von verschiedenen Teilmodellen modelliert. 3. Die einzelnen Prozesse werden ohne konkreten Implememtierungs-Vorschlag modelliert. 4. Datenverarbeitung visualisiert
		3.3	**Implementierung**	[3.3]	1. Welche Technologien verwende ich für meinen Prototypen: - „Welche Python Bibliotheken und IDE?“ - „Welche HW & SW-Anforderungen gibt es?“ 2. Welche Probleme traten bei der Programmierung auf?	1. Software-Abhängigkeits-Portfolio erstellen - Vergleich geeigneter Programmiersprachen - Recherche der erforderlichen Bibliotheken - Recherche der erforderlichen Hardware, Software und Auswahl 2. Software entwickeln - Fehler reporten an Hersteller, Bib, etc.	1. Erstellung eines IT-Konzepts in Form einer Beschreibung der notwendigen technischen Mittel anhand von Teilmodellen 2. Problemstellungen erklären und das Auftreten eines Problems „reverse Engineeren“
		3.4	**Evaluation**	[3.4]	1. Wie ist der Output des Prototyps zu bewerten? 2. Wie bewertet man die Qualität des Outputs? 3. Was kann verbessert werden?	1. Soll-Ist-Vergleich der Anforderungen mit dem Output des Prototypen. 2. Vergleich mit verwandten Arbeiten. 3. Recherche über potentielle Verbesserungen des Algorithmus.	1. Evaluation und Analyse des Ergebnisses anhand von grammatikalischer Richtigkeit und Sinn. 2. Bessere Ergebnisse mit meinen vergleichen. 3. Optimierungsmöglichkeiten für meinen Prototypen evaluieren.
4	**Generierung von übertragbarem Wissen**			[0] bis [3]	Um welche Elemente könnte mein Projekt modular Erweitert werden um ein Anderes oder Besseres Ergebnis zu erzeugen und welchen Einfluss könnte es auf die Forschung haben?	Verallgemeinerung aus den bisher erarbeiteten Ergebnissen.	Einordnung der Evaluationsergebnisse in einen gesellschaftlichen Kontext.

Abstract

– At the end , finally finished :) –

Contents

Chapter 1.

Introduction

1.1. Structure of the thesis

The aim of my thesis is to survey the current state of the art in text generation, especially on the focus of text summarization. For readers who are not familiar with machine learning in general, I will provide a zoom-in introduction into text summarization. My approach is feed-forward from the definition of machine learning, into the natural language processing field, further into the text generation field and within that, I focus on the text summarization part in chapter 1 - Introduction. When research and state of the art results in some natural language processing fields are achieved, those results can often be used across other disciplines as well. For this reason, I provide the most crucial text generation historical achievements in combination with the latest text summarization results, because both topics intersect in many aspects. The crucial concept of historical and modern approaches to summarize and generate text are introduced in chapter 2 - State of the Art. To illustrate the basic workflow of a text summarizing system, I programmed a prototype. The concept, development and evaluation of this summarizer are located in chapter 4 - Prototype, but it requires prior knowledge to fully understand the mechanism from the input to the output. Finally in the last chapter I will discuss further improvements for my prototype and a brief discussing into the future of text generation.

1.2. Machine Learning

In the last decade, Machine Learning (ML) is increasingly finding its way into businesses and society. Many websites and businesses use Machine Learning techniques to improve the user and costumer experience. The phrase *Machine Learning* was originally introduced in 1952 by Arthur Samuel. He developed a computer program for playing the game checkers in the 1950s. Samuel's model was based on a model of brain cell interaction by Donald Hebb from his book called *The Organization of Behavior* published in 1949. Hebb's book introduces theories on neuron excitement and the neural communication. Figure 1.1 illustrates the

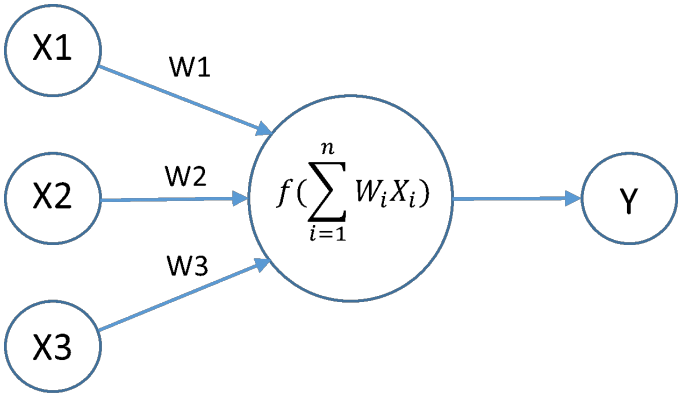

Figure 1.1.: A simple Neuron with 3 inputs and 1 output [Sing 17]

model of the brain cell. Nowadays, this brain neuron based model is mostly declared to be not realistic enough [Andrew Ng, deeplearning.ai], because the structure of a neuron in the brain is far more complex than the illustration in figure 1.1 suggests. Nevertheless, it provides a really good entry point for this research field.

The roots of Neural Networks (NN) lie down almost 80 years ago in 1943 when **McCulloch-Pitts** [McCu 43] compared for the first time neuronal networks with the structure of the human brain. The range in which Neural Networks (in the year 2020) apply to modern technologies is wide. Some disciplines have only been created due to the invention of Neural Networks, because they solve existing and new problems very effectively and efficiently. Many frequently held conferences around the globe proof continuous evidence of the successes of Neural Networks. Among those various disciplines counts for example *Pattern recognition* with Convolutional Neural Networks (CNN) [Yann 98] for example to predict the classes of images with the famous *CIFAR-10* dataset [Kriz]. Many amateurs [Löh 19] and experts annually attempt to show their latest results in beating the former best accuracy.

Convolutional Neural Networks is just one of many other Neural Network building blocks, because a modern network consists of many different layers. Natural Language Processing is one of the various sub fields of Machine Learning. Strictly speaking, it is actually a multidisciplinary field consisting of Artificial Intelligence (AI) and computational linguistics. Natural Language Processing is dedicated to understand and process the interactions between human (natural) language and computers. Natural Language Processing is a very broad term and can apply many different tasks, such as:

- Sentiment Analysis
- Machine Translation
- Speech Recognition

- Text Generation (Neural Text Generation *NTG*)
- Chat Bots

All of this tasks require many steps to function properly. In the broadest sense, there is always an Input and an Output, which are shown in Table 1.1.

Components of NLP methods			
	Speech	Text	Images
Input Analysis	Speech Recognition	Natural Language Processing methods	Image Recognition
Output Synthesis	Generation of Speech	Generation of Text	Generation of Images

Table 1.1.: A closer look into the NLP disciplines

It shows that the text generation is often the output part of a Natural Language Processing model. Data is collected through various different sources, e.g. images, videos or speech, then it is further processed and generates the desired output. Useful examples are shown in Table 1.2.

Examples of NLP methods			
	Speech	Text	Images
Input Analysis	Siri listens	Read in document	Image of a face
Output Synthesis	Siri answers	Generate Summary	Face detected

Table 1.2.: Examples for three different NLP tasks

For this Bachelor thesis, the focus is on the output part of a Natural Language Processing system, which inputs text as shown in Table 1.1 and 1.2. Text generation is therefore in general the output part of an input-output NLP system.

Another term for text generation is *Language Modelling*, because text generators use the words of a language and grammar as input for the model. In the past five years, primarly two approaches were used for modelling a Natural Language Processing system, namely the **rule-based** system and the **template-based** system (Figure 1.2) [Xie 17]. Today neural end-to-end systems are *state-of-the-art* [Jeka 17]. These systems offer more flexibility and scale with proportionately better results, and less data is required because of the increased

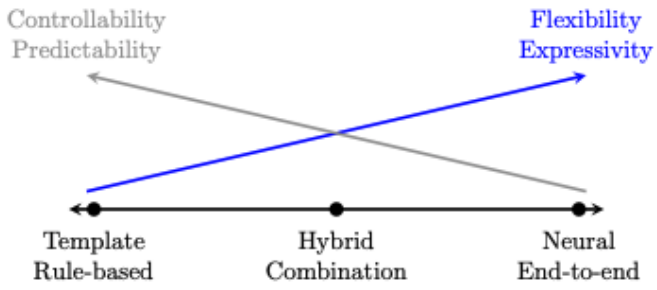

Figure 1.2.: Rule-Based vs. Neural-Text-Generations System [Xie 17], Page 4

complexity. A major disadvantage is that the necessary computing power has increased exponentially. However, this leads to a complex problem because it becomes more and more challenging to understand the decisions of the neural network. The neural network is still, to a large extent, a *black box*. Especially in NLP it gives surprisingly good results. The neural network models for text processing are difficult to understand, so nowadays, compromises between rule-based systems still have to be made, and hybrid systems are most commonly in use.

When Neural end-to-end systems are used, text generation is often referred as Neural Text Generation (NTG). More examples for Neural Text Generators as output synthetical component are:

- Speech recording and conversion to text
- Conversation systems e.g. chatbots
- Text summary
- Caption generation

In order to train language models properly, Deep Learning (DL) algorithms teach the model the probabilities of occurring words with respect to the preceding words. There are several approaches to achieve this goal. Language models can be trained on the level of words, whole sentences, or even whole paragraphs. The granularity in which the training takes place is called *n-grams*, where n represents the number of preceding words. Further explanation in Section 2.1.2.1 of Chapter 2. Deep Learning will be explained in necessary depth in Section 2.2.

1.3. Case study of a Text Summarization System

Chapter 2.

State of the Art

The goal of this chapter is to survey the development of the text generation from the old days until 2020. I can not go into detail for every single state of the art technology at each time step, because the focus is on the current state of the art. Briefly introducing the Recurrent Neural Network and Long Short Term Memory is necessary to understand the difference of the bad performing old technologies and the state of the art technologies nowadays. Since those two networks can fill up a thesis by themselves, I will provide only the basic information to understand the concept. Most of the text generation research results directly benefit the text summarization itself too, for that reason I cover up the development of text generation and its general architecture, such as text summarization, which approaches my prototype as well.

2.1. Background and Theory

2.1.1. Recurrent Neural Networks

Even though I introduced the neuron in a neural network as a kind of brain cell imitation, the neuron of a basic neural network will forget everything when it is shut down, unlike the brain. Making information persistent is a crucial step towards better performing models. Recurrent neural networks, or RNN, address this issue. They are networks with integrated loops, which allow the information to persist [Olah 15]. The network architecture of the RNN is important, because it denotes the first step into neural text generation and neural text summarization.

Figure 2.1 shows an unrolled Recurrent Neural Network. The input x_t on time step t, is passed to the neural network A. The network looks at the input on this time step and outputs the hidden state h_t at the same time step t. This loop allows the network to pass information from one time step to another. The picture 2.1 shows, that the learned parameter from input [x] on time step [t] will be passed as additional information to the next time step [t + 1] and so on. For example if a RNN wants to predict the next word in the sentence "Since I am

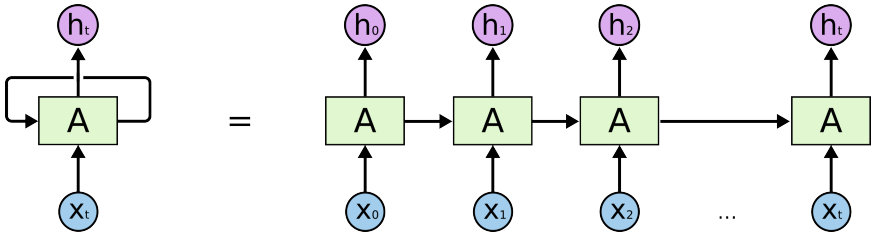

Figure 2.1.: Recurrent Neural Network with integrated loops [Olah 15]

living in Hong Kong .. by now I speak fluent *Cantonese*". The network needs to remember that the target country is Hong Kong to predict the language Cantonese. At each time step t, the hidden state h_t of the Recurrent Neural Network is updated by:

$$\boldsymbol{h_{(t)}} = f(\boldsymbol{h_{(t-1)}}, x_t)$$

where f is a non-linear activation function and x is the input in form of a word. The function f can be in the simplest case a sigmoid function which has either 0 or 1 as output, or the more complex and effective Long Short Term Memory cell, explained in the next Section 2.1.2 [Hoch 97]. The Recurrent Neural Network is trained to predict for example the next word in a sentence or sequence. This prediction is possible due of the learned probability distribution over a sequence. The output at each time step t is a conditional distribution $p(x_t|x_{t-1}, ..., x_1)$.

Theoretically with this approach it is possible to retain information from many time steps ago, but unfortunately, as the time span back grows, RNN's become unable to learn the information from too long ago cells. This phenomenon was explained by Sepp Hochreiter in 1991 [Hoch 91] under the name *vanishing gradient problem*. The solution to this problem is the Long Short Term Memory, short LSTM.

2.1.2. Long Short Term Memory

Long Short Term Memory cells were first proposed by Sepp Hochreiter and Jürgen Schmidhuber in 1997 [Hoch 97]. The LSTM is a special kind of Recurrent Neural Network, because it is able to remember long-term dependencies and informatiom. The goal of the cell is to solve the vanishing gradient problem of the Recurrent Neural Network. Inputs into this cell can be stored for a long period of time, without forgetting them, as in Recurrent Neural Networks. The LSTM is designed to avoid the loss of information (vanishing gradient problem), by intentionally ledging on to certain information over plenty of time steps.

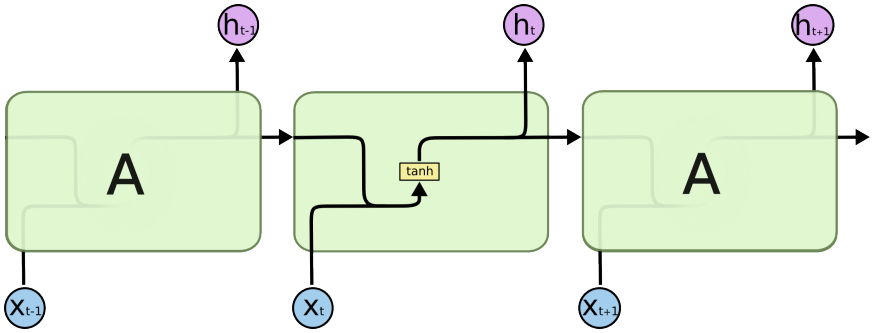

Figure 2.2.: The repeating module in an Recurrent Neural Network contains one single layer [Olah 15]

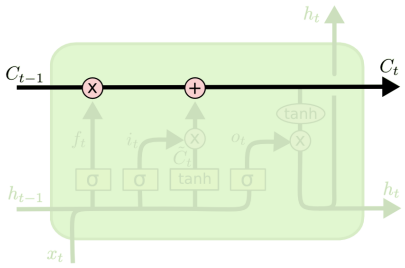

Figure 2.3.: Cell State of the Long Short Term Memory which acts as data highway [Olah 15]

LSTM's can be enrolled the same way like RNN's, but there is a core difference between the Recurrent Neural Network in Figure 2.2 and the Long Short Term Memory in Figure 2.4. The LSTM has four gates instead of one like the RNN. The four gates are:

- Forget Gate
- Input Gate
- Cell State
- Output Gate

The **Forget Gate** decides what information should be thrown away or kept. Information from the previous hidden state and information from the current input is passed through a sigmoid function. A sigmoid function takes an input and returns high values closer to 1 and smaller values closer to 0. The closer to 0 means to forget the state, and the closer to 1 means to keep the state.

The **Input Gate** updates the cell state. That decides which values will be updated by computing the values to be between 0 and 1 like the Forget Gate. Important information is closer to 1 and 0 means less important.

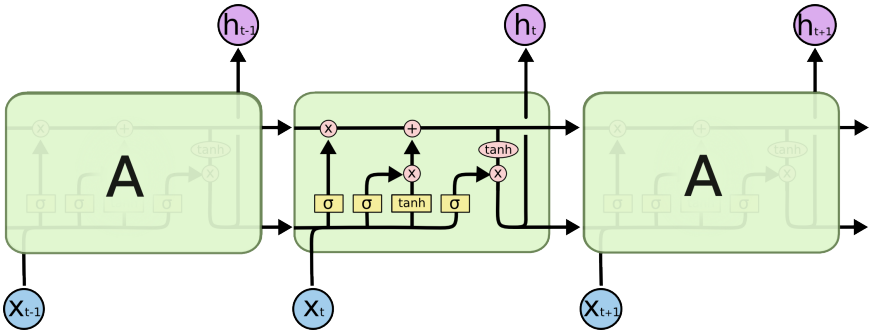

Figure 2.4.: The repeating module in an LSTM contains four interacting layers [Olah 15]

The **Cell State** is the core of the LSTM. It is the horizontal line shown in Figure 2.3. The cell state acts like the information highway in the cell. With only some minor linear computation, it runs through the entire cell. This way information can pass very easily through the cell.

The **Output Gate** decides what the hidden state of the next LSTM cell should be. The hidden state contains information on previous inputs and it is also used for predictions. The hidden state denotes the state which is passed from the output gate on time step t to the input gate for the LSTM cell on time step $t+1$.

The main idea of the LSTM is, that it can decide which information to remove, to forget, which to store and when to use it. It can also decide when to move the previous state information to the next, like the RNN shown in Figure 2.1. Even though many variations of the LSTM occupy the state of the art performance, the LSTM is used in many real business cases in production, like the Google translater or weather forecasting. The Long Short Term Memory paved the way for the sequence to sequence models.

2.1.2.1. N-Grams

2.1.3. Sequence to Sequence

In the year 2014, Google invented a new way to translate language by learning a statistical model with a neural machine translation approach [Suts 14]. Google called it Sequence to Sequence model [Suts 14], often shortened down to seq2seq, which consists of an encoder and a decoder.

Before that, language translation was originally processed by rule-based systems [Chen 96]. The systems computed their work by breaking down sentences into plenty of chunks and translating them phrase-by-phrase, but this approach created not easily understandable language.

After rule-based systems, statistical models have taken over the. Given a source text in e.g. German (f), what is the most suitable translation into e.g. English (e)? The statistical model p($g|e$) is trained on multiple texts (corpus) and finally outputs p(e), which is calculated only on the target corpus in English.

$$\hat{e} = argmax_e(e|g) = argmax_e p(g|e)p(e)$$

The formula means, among all Baysian probabilities p($g|e$)p(e), select the pair of words (translation), select the most likely to be the best translation (argmax). Even though this approach produces good results, it looses the wider semantical view, and so it is especially not effective for a good summarization technique.

For the first time, neural networks in form of feed-forward fully-connected neural networks produced such good results, that they replaced all non-network techniques. Affine matrix transformations are stacked together and are followed by non-linearities to the input and each following hidden layer [Beng 03] Page 1141-1142. However, these models require a fixed content length for their calculations, which makes them again not flexible enough to produce human-like translations.

Even if a LSTM (Section 2.1.2) was used to map sequences of words from one language to another, it will most likely produce errors or bad results. A single LSTM cell needs the same input length and output length, which is unrealistic for translating multilingual. For example the English "He is running" translated into German is "Er rennt". The LSTM itself can not translate that, because of the different word length. The Long Short Term Memory cell from Section **??** was invented independently from the sequence to sequence models, but finally three employees of Google published a paper about their approach to make use of the LSTM to create a sequence to sequence model, also called encoder-decoder model. The basic idea is that the encoder converts an input text to a latent vector of length N and the decoder generates an output vector of length V by using the latent encoded vector. It is called a latent vector, because it is not accessible during the training time (manipulating it), for example in a normal Feed Forward Neural Network, the output of a hidden layer in the network can not be manipulated. The initial use of encoder-decoder models was for machine translation.

Technologies for a specific field in the machine learning environment and especially text generation can often be used cross functional. The encoder-decoder model found its way into text summarization and automated email reply by Google [Scie 15] as well. Figure 2.5 illustrates the model for Google's automated email reply.

Figure 2.5 makes use of an Long Short Term Memory cell, which captures situations, writing styles and tones. The network generalizes more flexible and accurate than a rule-based model ever could [Scie 15].

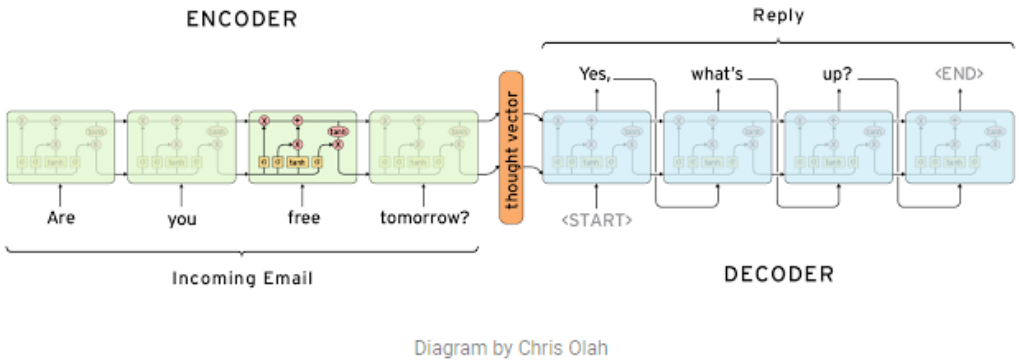

Figure 2.5.: LSTM encoder-decoder model for automated E-Mail reply

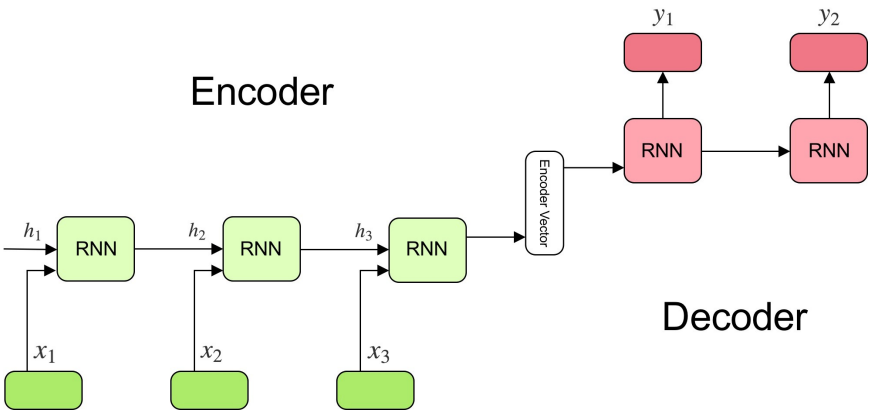

Figure 2.6.: Encoder-decoder sequence to sequence model [Kost 19]

2.1.4. Encoder and Decoder

In the following, the encoder and then the decoder will be explained to have a better insight in how this technology works. The prototype from Chapter 3 is based on this kind of model. As already mentioned a sequence to sequence model is often referred to as a encoder-decoder model. The sequence to sequence model itself is built using a Recurrent Neural Network or a Long Short Term Memory as explained in the last Section 2.1.3.

Figure 2.6 shows, that the encoder decoder model is built up from actually three parts:

- Encoder
- Intermediate (encoder) Vector
- Decoder

Vocabulary Table

aardvark	1.32	1.56	0.31	–1.21	0.31
ate	0.36	–0.26	0.31	–1.99	0.11
...	–0.69	0.33	0.77	0.22	–1.29
zoology	0.41	0.21	–0.32	0.31	0.22

(each row is actually 300 dimensions)

Figure 2.7.: Snippet of an example vocabulary table [Muga 18]

The **Encoder** iteratively integrates the words in a sentence into the hidden state h into the Long Short Term Memory cell. Figure 2.3 shows a single LSTM cell with the input cell state C at the time step *t-1* and the input of the hidden state h at the same time step *t-1*. This is necessary for the cell to compute both the input words, but also the knowledge from prior words. Words are represented as latent vectors in the sequence to sequence models and are stored in a vocabulary table. Each fixed length vector stands for a word in the vocabulary, for example the vector length is fixed to a dimension of 300. In a simple case, the number of words in the vocabulary is fixed to e.g. 50.000 words, hence the dimension of the vocabulary table in Figure 2.7 is [50000 x 300].

A connection of multiple recurrent units (three in Figure 2.6) where each accept a single element as an input, gains information and propagates it forward to the next cell and accordingly the next time step. In the example of Figure 2.6, the hidden state of *h3* is calculated based on the prior two cells.

The **Encoder Vector** is the last hidden state of all the encoder cells, in this example the encoder vector is located at the output of cell three. The vector tries to combine all of the information from the prior encoded words with the purpose to help the decoder make accurate predictions. Basically, the encoder vector is the initial input for the decoder part of the model.

The **Decoder** unrolls the encoder vector from meaning space into a target sentence. The meaning space (shown in Figure 2.9) is a mapping of concepts and ideas that we may want to express to points in a continuous, high-dimensional grid [Muga 18]. The minimum requirement for the meaning space is to consist at least of the last state of the encoder Recurrent Neural Network (encoder vector). The decoder computes a probability distribution for each word in the encoder vector to generate the next state. In the example case, the output is generated by multiplying the hidden state in the encoder vector h by the output matrix of size [300 x 50000]. The product of this matrix multiplication is a vector of size [50,000], that can be normalized with a *softmax* into a probability distribution over words in the

vocabulary. The network can then choose the word with the highest probability, because the softmax squeeses all outputs into a summed up probability of 1. For example:

"Since I am living in Hong Kong, by now I speak fluent ... "

- Cat: 0.01
- running: 0.005
- Cantonese: **0.5**
- Mandarin: 0.3
- French: 0.015

The chosen word is **Cantonese**, because it is the highest probability among all probabilities which are summed up to 100%

2.1.5. Attention

In general, the explanation of the sequence to sequence models just covered the very basic idea of the model. To achieve the state-of-the-art result, not only a single vector can be used for encoding the entire input sequence, but multiple vectors each capable of capturing other information.

In the encoder and decoder model, the length of the state vector h does not change for the input and output. As shown in the example of Section 2.1.3, sentences translated into another language can have a different word length. For the model to automatically adjust the length of the output, is to use the technology called *attention* [Bahd 14] [Vasw 17].

Figure 2.8 shows the basic concept of attention. The Long Short Term Memory is not starting to right before time step $t = 6$ at state h_6. Attention enables the network to look at all prior encoded states of the words, takes the weighted average probability of the vectors and also uses this as additional information. Attention also projects its vectors into the meaning space (Figure 2.9.

Sequence to sequence models can be entirely built up from the attention model [Vasw 17].

2.2. Text Generation

In the modern era of big data, retrieving useful information from a large number of textual documents is a challenging task due to the unprecedented growth in the availability of

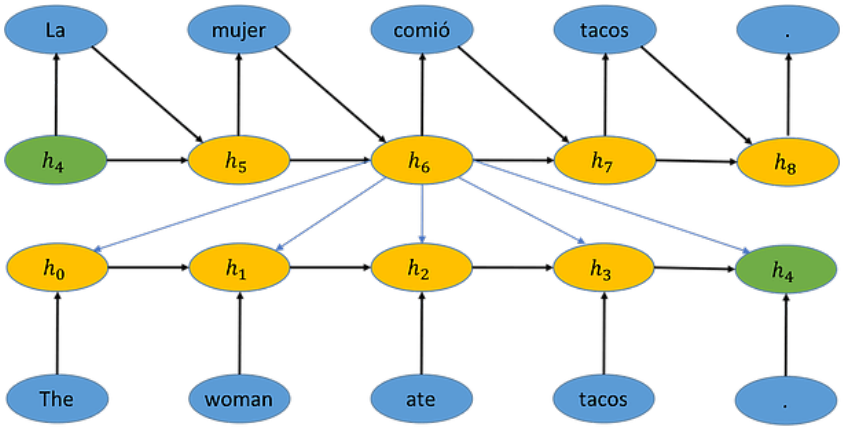

Figure 2.8.: Attention mechanism for Spanish-English translation [Muga 18]

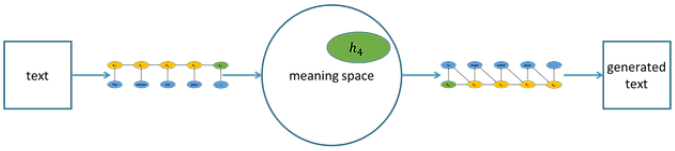

Figure 2.9.: Meaning Space of the Attention model [Muga 18]

online blogs, forums, news, and scientific reports that are tremendous. Automatic text summarization provides an effective and convenient solution for reducing the amount of time it takes to read all of it. The goal of the text summarization is to compress long documents into shorter summaries while maintaining the most important information and semantic of the documents [Rade 02] [Mehd 17]. Having the short summaries, the text content can be retrieved, processed and digested effectively and efficiently. Generally speaking, there are two basic approaches for performing a text summarization: Extractive and Abstractive [Mani 99].

2.2.1. Text Generation Tasks

Traditionally, the nlg problem of converting input data into output text was addressed by splitting it up into a number of subproblems. The following six are frequently found in many nlg systems (Reiter and Dale, 1997, 2000); their role is illustrated in Figure 1:

- Content determination: Deciding which information to include in the text under construction
- Text structuring: Determining in which order information will be pre- sented in the text

- Sentence aggregation: Deciding which information to present in individual sentences
- Lexicalisation: Finding the right words and phrases to express informa- tion
- Referring expression generation: Selecting the words and phrases to iden- tify domain objects
- Linguistic realisation: Combining all words and phrases into well-formed sentences

2.2.1.1. Content Determination

2.2.1.2. Text structuring

2.2.1.3. Sentence aggregation

2.2.1.4. Lexicalisation

2.2.1.5. Referring expression

2.2.1.6. Linguistic realisation

2.2.2. Architectures and Approaches

nlg-survey-long Kapitel 3

- Rule-based, modular approaches
- Planning-based approaches
- Data-driven approaches

2.2.2.1. Rule-based approach

2.2.2.2. Planning-based approach

2.2.2.3. Data-driven approach

2.2.3. Neural Text Generation

2.2.3.1. Supervised Learning

2.2.3.2. Reinforcement Learning

2.2.3.3. GANs

2.3. Current Trends in Text Summarization Technology

2.3.1. Summarization Factors

Single Doc - Multi Doc Input Factors Purpose Factors output factors neural-text-summary

2.3.2. Extractive and Abstractive

2.3.2.1. Excractive

2.3.2.2. Abstractive

2.3.3. Combinational Approach

2.3.4. Reinforcement Learning

2.3.5. Evaluation

ROGUE

Chapter 3.

Prototype

```
1 x = 1
2 if x == 1:
3     # indented four spaces
4     print("x is 1.")
```

Listing 3.1: This is an example of inline listing

You can also include listings from a file directly:

```
1 x = 1
2 if x == 1:
3     # indented four spaces
4     print("x is 1.")
```

Listing 3.2: This is an example of included listing

3.1. Objective

Textsummarization

3.2. Technical concept

Fachkonzept - Proto

3.2.1. Structure

The different steps of Text Generation

- Importing Dependencies
- Loading the Data
- Creating Character/Word mappings
- Data Preprocessing
- Modelling
- Generating text

3.2.2. Neuronal Net

LSTM

RNN

Experimenting with different models

- A more trained model
- A deeper model
- A wider model
- A gigantic model

3.2.3. Process Modeling

Funktionen etc.

3.2.4. Data flow modelling

Diagramm

3.3. Implementation

Code for the Machine Translating

3.4. Evaluation

Print Ergebnisse

Bild

Image Caption

Chapter 4.

Generation of transferable knowledge

Modular expandability of my project. Classification in social context

Appendix A.

Supplemental Information

List of Figures

List of Tables

List of Listings

Bibliography

[Bahd 14] D. Bahdanau, K. Cho, and Y. Bengio. “Neural Machine Translation by Jointly Learning to Align and Translate”. 2014. cite arxiv:1409.0473Comment: Accepted at ICLR 2015 as oral presentation.

[Beng 03] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. “A Neural Probabilistic Language Model”. *JOURNAL OF MACHINE LEARNING RESEARCH*, Vol. 3, pp. 1137–1155, 2003.

[Chen 96] S. F. Chen and J. Goodman. “An Empirical Study of Smoothing Techniques for Language Modeling”. In: *34th Annual Meeting of the Association for Computational Linguistics*, pp. 310–318, Association for Computational Linguistics, Santa Cruz, California, USA, June 1996.

[Hoch 91] S. Hochreiter. “Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München”. 1991.

[Hoch 97] S. Hochreiter and J. Schmidhuber. “Long short-term memory”. *Neural computation*, Vol. 9, No. 8, pp. 1735–1780, 1997.

[Jeka 17] O. D. Jekaterina Novikova and V. Rieser. “The E2E Dataset: New Challenges For End-to-End Generation”. 2017.

[Kost 19] S. Kostadinov. “Understanding Encoder-Decoder Sequence to Sequence Model”. 05 2019.

[Kriz] A. Krizhevsky, V. Nair, and G. Hinton. “CIFAR-10 (Canadian Institute for Advanced Research)”.

[Löh 19] T. Löhr and T. Bohnstedt. “Image Classification on the CIFAR10 Dataset”. 2019.

[Mani 99] I. Mani and M. Maybury. “Advances in Automatic Text Summarization”. pp. 123–136, The MIT Press, 1999.

[McCu 43] W. S. McCulloch and W. Pitts. “A logical calculus of the ideas immanent in nervous activity”. *The bulletin of mathematical biophysics*, Vol. 5, No. 4, pp. 115–133, 1943.

[Mehd 17] M. A. S. S. E. D. T. J. B. G. K. K. Mehdi Allahyari, Seyedamin Pouriyeh. "Text Summarization Techniques: A Brief Survey". *Computation and Lignuistics*, 2017.

[Muga 18] J. Mugan. "Generating Natural-Language Text with Neural Networks". 07 2018.

[Olah 15] C. Olah. "Understanding LSTM Networks". 08 2015.

[Rade 02] D. R. Radev, E. Hovy, and K. McKeown. "Introduction to the Special Issue on Summarization". *Computational Linguistics*, Vol. 28, No. 4, pp. 399–408, 2002.

[Scie 15] G. C. S. R. Scientist. "Computer, respond to this email.". *Google AI Blog*, 11 2015.

[Sing 17] P. Singh. "Neuron explained using simple algebra". 2017.

[Suts 14] I. Sutskever, O. Vinyals, and Q. V. Le. "Sequence to Sequence Learning with Neural Networks". In: Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds., *Advances in Neural Information Processing Systems 27*, pp. 3104–3112, Curran Associates, Inc., 2014.

[Vasw 17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. "Attention is All you Need". In: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., *Advances in Neural Information Processing Systems 30*, pp. 5998–6008, Curran Associates, Inc., 2017.

[Xie 17] Z. Xie. "Neural Text Generation: A Practical Guide". 2017.

[Yann 98] L. B. Yann LeCun, Patrick Haffner and Y. Bengio. "Object Recognition with Gradient-Based Learning". 1998.