technische universität
dortmund

fakultät für informatik

Bachelor's Thesis

Development of a GPT-based chatbot for university preboarding

Nenad Kalicanin

November 6, 2024

Supervisors:
PD Dr. Frank Weichert
Prof. Dr. Heinrich Müller

Computer Science VII
Computer Graphics
TU Dortmund

Contents

Mathematical Notation

Notation	Meaning
$\mathbb{N}$	set of natural numbers $1, 2, 3, \ldots$
$\mathbb{R}$	set of real numbers
$\mathbb{R}^d$	d-dimensional space
$\mathcal{M} = \{m_1, \ldots, m_N\}$	unordered set $\mathcal{M}$ of N elements m_i
$\mathcal{M} = \langle m_1, \ldots, m_N \rangle$	ordered set $\mathcal{M}$ of N elements m_i
$\mathbf{v}$	vector $\mathbf{v} = (v_1, \ldots, v_n)^T$ with N elements v_i
$v_i^{(j)}$	i-th element of the j-th vector
$\mathbf{A}$	matrix $\mathbf{A}$ with entries $a_{i,j}$
$G = (V, E)$	graph G with node set V and edge set E
$\mathcal{X} = (A, B)$	entity $\mathcal{X}$ with attributes A and behaviors B
$\mathcal{L} = [l_1, \ldots, l_N]$	list $\mathcal{L}$ of N elements l_i

1 Introduction

This chapter provides an abstract overview and motivation for the content of this thesis. The background for this thesis is based on the potential computer science has offered in recent years. Most importantly, artificial intelligence (AI) stands out for its enormous capabilities. Chatbots are developed by companies, e.g. all major tech companies Meta, Google, and Microsoft are leveraging the technology of machine learning. [1]

1.1 Background

The motivation stems from the aim to create useful technology and the availability of easily implementable code. For many years, in constant exchange, neural networks (NN) that are able to contextualize and weigh information resolved into nowadays chatbots. Large language models (LLMs) saw enhancements in comprehension and understanding. Complex tasks, summaries, and translations benefited from increased precision and a generally higher level of capability. Improvements were also observed in the input selection. LLMs were enhanced to be multi-modal, so they could process audio, video, and images, providing reliable data and conclusions for a wide range of questions [1].
Ultimately, TU Dortmund should benefit from these notable and potent software advancements. I suggest a proof-of-concept as the basis of this thesis, which answers questions from *tu-dortmund.de* visitors.
There is a never-ending demand for finishing trivial tasks, automating tasks, or assisting humans in solving complex tasks by breaking them down into sub-tasks that are processable for the chatbot. One of those trivial but relevant tasks in a young adult's life is to inform oneself about the future paths one can take. For example, suppose there is the possibility of starting to study at a university. In that case, one should be well-informed about the challenges ahead, and finding this information can be difficult at times. A solution could be a way to just ask someone any question on a current search of information to skip the searching of websites,

This Thesis is motivated to provide information of such a chatbot and how a proof-of-concept implementation would behave.

1.2 Goal

To depict a clear overview of chatbot history, technology and capability combined with a proof-of-concept that utilizes all the provided information to create a simple yet useful application that simplifies the task, for example, a student searching for detailed information on the university website.
This overview should explain all the necessary major points to understand the creation of today's state-of-the-art LLMs through training, the underlying technologies, neural networks, and the many methods that are encompassing all these technologies to make machine learning possible and the transformer architecture efficient.

1.3 Structure of Thesis

Outlining the thesis, it continues after this introductory first chapter with the second chapter, the history of *Interactive Communication Models*, showing the past efforts and approaches of chatbots. There will be a range of different chatbots shown, with their underlying technology dating back from 1900 until today.
Concluding the introduction to chatbots, Chapter three displays the achievements of transformer architecture and important information about ChatGPT. This leads to an understanding of the structure of natural language processing, which finishes the third chapter.
Language Processing is the fourth chapter, explaining the process of LLM creation by explaining the architecture and the basic data preparation, as well as the pre- and fine-tuning which are crucial for the quality of the resulting LLM.
Chapter five shows the underlying technologies LLMs are build upon and the aspects of machine learning itself. It starts with an introduction to machine learning, followed by the definition of neural networks to finalize the picture of chatbot creation and functionality. The core feature of this technological development is the transformer architecture, explained lastly in the fifth chapter in great detail, completing the informational background about chatbots.
The sixth chapter shows the GPT-based preboarding chatbot extension developed for this thesis as a proof-of-concept, its use-case and the technology used for development.

Concluding this development in regard of all previous information the evaluation takes critique and gives further information about the results of this proof of concept and the thesis itself.
At last the summary give an outlook of further technologies and advancements in recent history.

2 Interactive Communication Models

Large language models (LLM) developed for useful applications, e.g. ChatGPT[1], rely on a long history of scientific studies. The historic beginnings of artificial intelligence (AI) emerged around 1950 with Alan Turing. In the last 70 years, technology has enabled the processing of large quantities of data and strong processing units, which have been crucial for the development of AI [2].

The great topic is AI and the achievement of robots or machines solving tasks far more complex than just answering questions. AI is defined into two categories: *Narrow AI* and *General AI*. In the form of NLP tasks, the GPT model should be categorized into the section of *Narrow AI* because the capabilities of a *General AI* should be similar and strong to those of an intelligent human being, but that is not the case, just predicts the next word based on its pretraining. This strong intelligence, or artificial general intelligence (AGI) is still not developed but highly pursued to outsource all possible tasks a human can outsource to those machines. Nevertheless, the *Narrow AI* is still capable of finishing tasks faster and there are countless examples in medicine with the generation of protein structures where this *weak AI* revolutionizes human existence and scientific knowledge, self-driving vehicles, and speech recognition [3].

The chatbots used today with neural networks (NN) are complex statistical and probabilistic constructs and serve mostly natural language processing (NLP), therefore the importance of the generation of texts from several perspectives and reasons prevails. Beginning with mathematical models, the idea of generating human-like texts followed probably more than just 100 years of scientific development. The chapter starts with Andrej Markow and ends with the current state-of-the-art chatbot ChatGPT [4].

[1] https://chatgpt.com/

2.1 Classical Approaches

Andrej Andrejewitsch Markov invented the statistical model for predicting random sequences in 1906 to model stochastic processes. This marked the earliest form of language modeling and the beginning of chatbot development. For auto-completion tasks and other machine learning (ML) aspects, the Markov Chain was used and implemented in the Loebner prize-winning software HeX in 1907 [4].
An example of the statistical model is the use of the Markov Chain to express the correlation of words in a sentence, for example, "The winner in the battle." This sentence is broken down into words, and these words are represented by a set of states $S = \{s_1, s_2, \ldots, s_n\}$, and a transition is the probability of the following of one word to another.
This correlation is expressed in the following transition matrix 2.1 $\mathbf{P}$ as:

$$\mathbf{P} = \begin{bmatrix} P_{11} & P_{12} & \cdots & P_{1n} \\ P_{21} & P_{22} & \cdots & P_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ P_{n1} & P_{n2} & \cdots & P_{nn} \end{bmatrix} \tag{2.1}$$

The set is distinct, so the word the is used in the sentence twice but will be mathematically represented once. For example, the following states:
- (s_1): "The"
- (s_2): "winner"
- (s_3): "in"
- (s_4): "battle"

The transition matrix 2.2 P would be:

$$\mathbf{P} = \begin{bmatrix} 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.2}$$

The first Row represents the appearance of "the/The" and the following rules are considered:
- ($P_{11} = 0$): "The" is never followed by "The".
- ($P_{12} = 0.5$): "The" is followed by "winner" 50% of the time.
- ($P_{13} = 0.5$): "the" is followed by "in" 50% of the time.
- ($P_{14} = 0$): "the" is never followed by "winner"

Probabilistic and generative chatbots profited from early on from Markov chains. It shows key proofs that inspired later inventions and concepts in conversational AI, and is still present in modern chatbot design as well [4].

The 1950s Turing test tried to emphasize what an accurate chatbot should be, indistinguishable from human answering. In his Paper, Alan Turing presents an interrogator asking questions to a human and a computer without knowing who he is talking to. Since then, this test has not been passed by any machine for 70 years, yet a crucial goal for AI Development. Nevertheless, many doubt that it is an absolute measurement of intelligence, but it remains certainly a milestone [4].

ELIZA, developed by Joseph Weizenbaum in 1966 was a pioneering chatbot brought to the public and was not based on a statistical approach but on pattern-matching techniques with a set of rules and templates for answering, which were mapped to certain input words or phrases. A match is searched by analyzing the input for keywords, patterns, or structures. These chatbots employed pattern recognition algorithms that drove small-scale applications. The lack of understanding was professionally hidden through asking open-ended questions to drive the conversation further and by profound pattern recognition and smart templates, that substituted the input keywords into the template, conclusions could be reached as well. The initial construction was a therapist chatbot to relieve the patients from their challenges by talking to them. This lack of understanding was not a problem at all, because in some cases people really thought they were chatting with a real human, but its finite templates still cannot track all the possibilities available in a human conversation [4].

A rather unusual chatbot was Racter which appeared in 1983 with its random generation of novel text and prose. It did not serve any intelligent purpose but prompted interesting texts and the authors Chamberlain and Etter were even able to generate a whole book with this chatbot. Equipped with context-free grammar rules and randomness, it was able to generate texts which amount were groundbreaking at that time.[4].

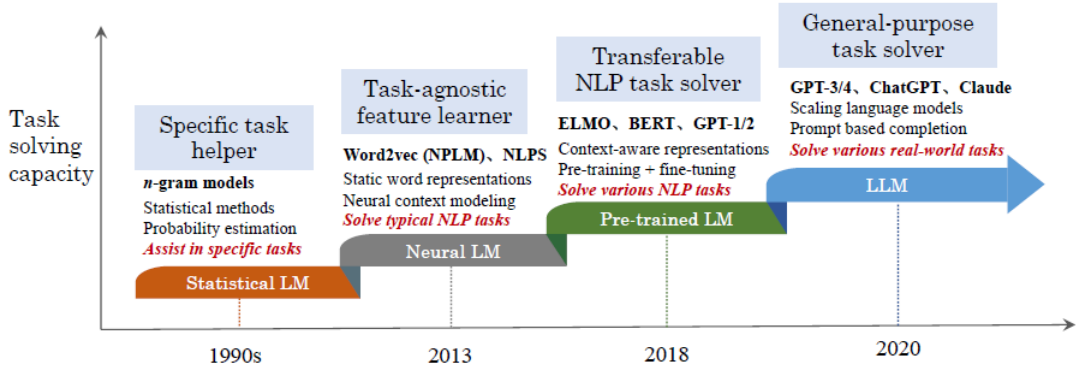

Figure 2.1: The history of language models beginning from 1990s [5].

Figure 2.1 shows four different types of language models developed through the years from 1990 until 2020 and afterward. First, the statistical language model with n-gram models as the representative in the chart, Neural language models with Word2sec; pre-trained language models with GPT-1; and finally LLMs with GPT-3.

A Microsoft Office Product to assist users in writing experience named Clippy was introduced in 1997 and years later removed, because of many critiques of its bad code, design, and unappealing experience in conversation and looks. Just the initial phrase ”Dear Mr.X” would result in the chatbot suggesting help for letter writing. The answers were predictable and generic and a lack of retention of past conversations often resulted in looping answers. [6].

2.2 Machine Learning Approaches

ML was enhanced by deep learning in the early 2000s and gave computers the ability to comprehend information and interpret from various sources for example texts, images, audio, and videos. Enterprises financed and pushed the development of AI to address complex challenges. The results of ML increased in quality due to larger datasets and profound learning algorithm [4].

20 years after ML was focused and practically implemented to solve real-world problems, the first LLMs were developed such as Bard and ChatGPT. After the invention of transformer architecture (TA) in 2017, chatbots based on showed incredible capacities to process natural language tasks. The handling of long and complex inputs through selective attention and parallelization offered superior performance, thus its use is inevitable today. The name Large Language Model derives from its training

of a growing large corpus of textual data for the creation of pattern recognition and creating a correlation between certain facts and words [4].

A DARPA project called PAL or *Personalized Assistant that Learns* emerged in the early 2000s. An AI assistant ought to help military commanders assist with decision-making was the goal of the PAL managed by the nonprofit research group SRI International. PAL was used as a groundwork for CALO(Cognitive Agent that Learns and Organizes) a predecessor to Siri.

The 150 million-dollar project *Cognitive Assistant that Learns and Organizes*, or CALO, took 5 years and 300 researchers across 22 institutes to develop. This AI assistant could learn and execute routine tasks automatically. AI capabilities should enable the chatbot to interact with humans, understand their intentions, and react to them. It was around 2003, when machine learning and deep learning started emerging, that this project made a significant impact on the further development of virtual assistants e.g. Apple's Siri. This context-aware chatbot capable of learning and reacting intelligently created many new conversational agents and brought a broad spectrum of AI fields together. After that time of development on CALO, the two entrepreneurs Adam Cheyer, Dag Kittlaus, and the computer scientist Tom Gruber founded Siri Inc. in 2007 to create a virtual personal assistant to act as a chatbot agent for conversing and performing tasks [4].

In 2011, Apple introduced Siri, the virtual personal assistant, to the iPhone 4S. Apple's microphones back then captured audio (16,000 samples per second) and analyzed it in 0.2-second frames. A Deep Neural Network (DNN) converted these frames' sound spectrums into probabilities to differ between "Hey Siri", silence, and others. Apple takes care of user privacy and stores the voice data on devices, to guarantee security. Apple's voice trigger system leverages accuracy, efficiency, and privacy, making it a robust and user-friendly system. Over time, Siri adapts to the user's voice, enhances accuracy, and reduces false triggers. Siri improves accuracy and minimizes battery drain with a multiple-staged approach. First, it detects potential trigger phrases and then rejects false positives [4].

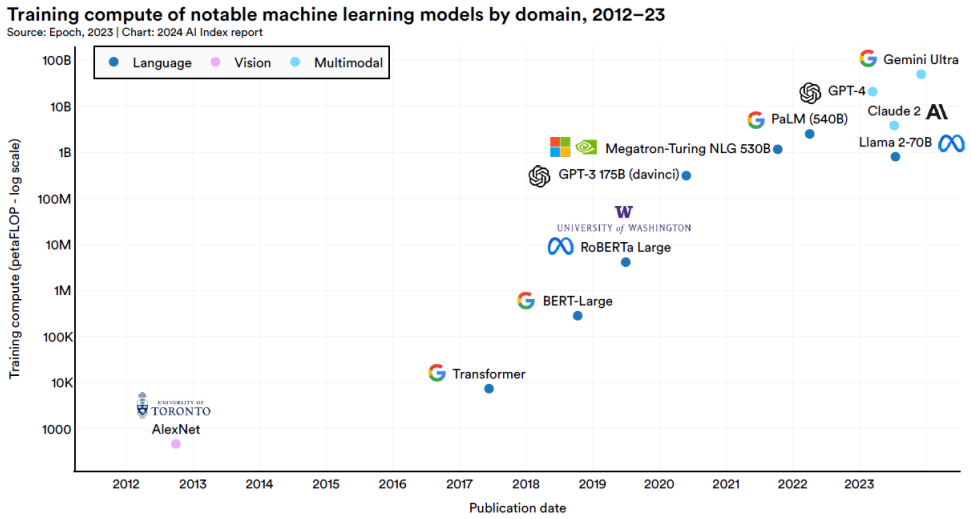

Figure 2.2: The models from 2012 until 2023 measured by the computational amount needed for training in petaFlop per day [1].

Many different new ML models have been established for public usage as chatbots, as shown in Figure 2.2. This chart starts in 2012 and ranges until 2023, with the most entries shown after 2017 when Google Deep Mind employees published the transformer architecture, thus revolutionizing the way these ML models are built. The most computationally costly ML model is Gemini Ultra with close to 100B petaFLOPS a day [1]. Where FLOPS means "Floating Point Operations per second" to estimate the necessary computing power, and peta means 10^{15}[7].

2.3 GPT

The abbreviation GPT stands for *Generative Pretrained Transformer*. *Generative* refers to creating, coherent, and contextually appropriate texts. *Pretrained* means training of the model before usage on a large set of data. The *Transformer* stands for the latest architecture used. This combination of prerequisites has been worked on over the years but finalized by the Google AI Team in a specific Paper and brought to life through OpenAI [3].

Training NN with large corpses of data with the TA enables robust language skills, strong reasoning, and contextual comprehension capabilities. The continuous progress of GPT-1 until GPT-4o has set a milestone after every iteration. Starting with GPT-1 with about 117 million parameters until the public release into 1.5 billion parameters and with the GPT-3 release a model with 100 times the parameters. The TA utilizes adapted techniques, especially for natural language generation and

manipulation and excels at long-range textual sequences, crucial for language tasks. Core components are transformer blocks using self-attention mechanisms to focus on the most important parts of input text while taking everything into account, with scores fed into feed-forward networks to filter correlation and not just linear nature between input and output [4].

3 ChatGPT

ChatGPT, a chatbot, uses a generative pretrained large language model (LLM) that has a feed-forward neural network (FFNN) in its decoder-only architecture. [8]. Based on the transformer architecture (TA) the conversational AI produces a text, by predicting the next token on a given input question [9].

3.1 ChatBot and Architecture Inventors

ChatGPT is used for different natural language processing (NLP) tasks, fine-tuned for machine translation, and more, using Google's TA, published in 2017 in *Attention Is All You Need* by Ashish Vaswani and colleagues. In earlier years, NLP tasks were solved by the encoder-decoder architecture on recurrent or convolutional networks [9]. Models trained on sequential data using recurrent NNs, which are able to solve different tasks with their long short-term memory NNs, are overwhelmed by the fact that the sequential nature of recurrent models computing the symbol at the hidden layered h_t position does not allow parallelization [9].
The power of GPT *Generative Pretrained Transformer* is in this parallelization that enables not just a faster outcome but a more profound one. The outputs' context is created based on a deeper correlation between each input embedding [9].
NLP models were trained on large amounts of annotated data related to a specific task before GPT. Since it is difficult to access a large quantity of labeled data to precisely train a model, and because these NLP models, besides their trained tasks, were unable to complete other tasks, they were restrictive. These drawbacks were first solved by the solution OpenAI presented with the model GPT-1, trained on unlabeled data and fine-tuned by prompting from users to solve tasks of question-answering, categorization, and sentiment analysis. GPT-1, released by OpenAI in 2018 the first model ever that was able to read text and respond to queries [10]. From that on OpenAI developed its GPT LLM further adding new technologies and solving many problems as shown in Figure 3.1.

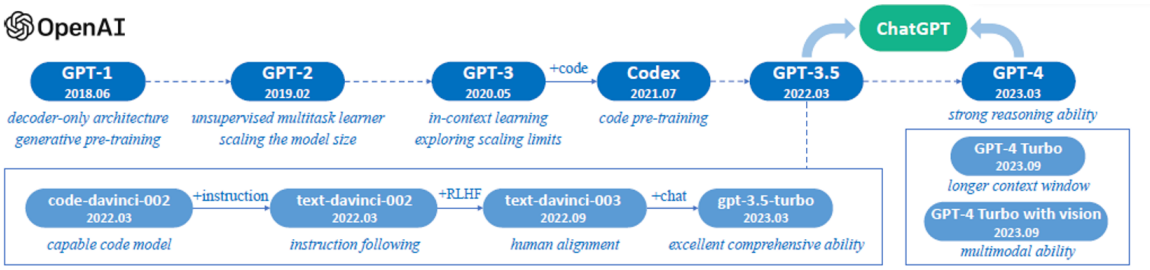

Figure 3.1: A basic map of the evolution of ChatGPT [5].

3.2 Initial Transformer Achievements

Until 2017, those chatbots achieved a fraction of the tasks with a lot more training than the TA introduced by Googles' employees the same year [9].

As shown in Table 3.1 the marked numbers for a direct exchange with the competing models and one-third of training costs, measured in FLOPS. Where FLOPS means "Floating Point Operations per second" to estimate the necessary computing power [7].

The measurement was the task of translating the text into another language, the BLEU score (Bilingual Evaluation Understudy) [11]. It scored a new best on English-French translation scoring with a 41.8 BLEU after just 3.5 days of training on eight P100 graphical processing units (GPU). This score is considered *good* for the BLEU standards where 50 is *excellent* [9].

These statistics show definite improvements over other architectures as explained the parallelization process enables this faster and more cost-efficient approach. The major impact was the self-attention mechanism that allowed the NN insight into

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet	23.75			
Deep-Att + PosUnk		39.2		$1.0 \cdot 10^{20}$
GNMT + RL	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Table 3.1: The transformer superior results at machine translation [9].

the whole context of the sentence while predicting the next token for the next word, which enabled better understanding and generation of language. This mechanism is responsible for the decoder that generates the output text given of the input representation [10].

3.3 Technical Details: A GPT Training

LLMs need an immense amount of training and many different parameters that are changed during the training. Table 3.2 provides an overview of the hyperparameters for the training of the models mentioned. These do not change they are preconfigured [12].
The following GPT-3 is not created by OpenAI, but rather a trained autoregressive language model from the paper "Language Models are Few-Shot Learners" to measure in-context learning abilities [12].
The technical details are similar to the actual GPT-3 training done by OpenAI with for example the same amount of n_{params} and the training dataset "Common Crawl" [10].

The names indicate briefly the model and its size.
n_{params} are model parameters representing the model's complexity and capacity of the learned weights.
n_{layers} is the amount of transformer layers.
d_{model} shows the dimensionality of vectors and the model's hidden state.
n_{heads} the number of heads in the transformer's attention layer.
d_{head} is dimensionality for each head calculated by division of d_{model} and n_{heads}.
Batch size defines each training batch containing this number of tokens.
Learning rate means the rate of model parameter update in training.
These previously explained hyperparameters are listed in Table 3.2 provided by the "Language Models are Few-Shot Learners" paper discussing their personal GPT-3 model training [12].

3.4 Natural Language Processing

NLP is a section of AI and Linguistics that emerged in the process of making computers understand words in human language, to solve user's work or make communication with machines in human language possible. It is divided into natural

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

Table 3.2: Overview of technical details for "GPT3". Models trained on 300 billion tokens [12].

language understanding (NLU) and natural language generation (NLG) [13]. NLP tasks include summarizing, answering questions, and others. These tasks require information handling, contextualization, and semantic understanding capabilities.

NLU consists of phonology(sound of words), morphology(form of words), pragmatics(expression and understanding), and syntax and semantics as depicted in Figure 3.2.

NLG prerequisites the understanding of provided context through input sequences, with the following: Fulfilling several tasks by generating texts for sentence completion, summarizing, dialogue generation, and machine translation [14].

Noah Chomsky invented a lot of new ideas and concepts regarding syntax in 1965. There are two levels of NLP tasks that can be categorized as *high level* and *low level*, where speech recognition would be a *higher level* task and *lower level* would be core natural language tasks. Some of the tasks are combined to solve a larger problem, and many of them are interwoven [13].

Part of speech tagging is naming every word according to its grammatical correspondence [13]. Morphological segmentation means breaking down words into smaller morphemes that still contain meaning and can be evaluated and identified as belonging to a certain class [13]. Named entity recognition (NER) refers to the capability to find important names in a text that are connected to certain entities [13]. Coreference resolution refers to understanding the references to the same object in a sentence or larger corpus of texts [13]. Discourse analysis identifies how parts of a text fit together and are connected to create a well-structured discourse [13]. Machine translation refers to machines being able to convert one human language to another [13]. Optical character recognition (OCR) converts words in a picture into readable text by determining the image text's corresponding words [13]. Automatic summarization refers to generating a correct summary from text and detailed information [13]. Most of these techniques are used in LLMs today and some of them are used in the phases of NLP.

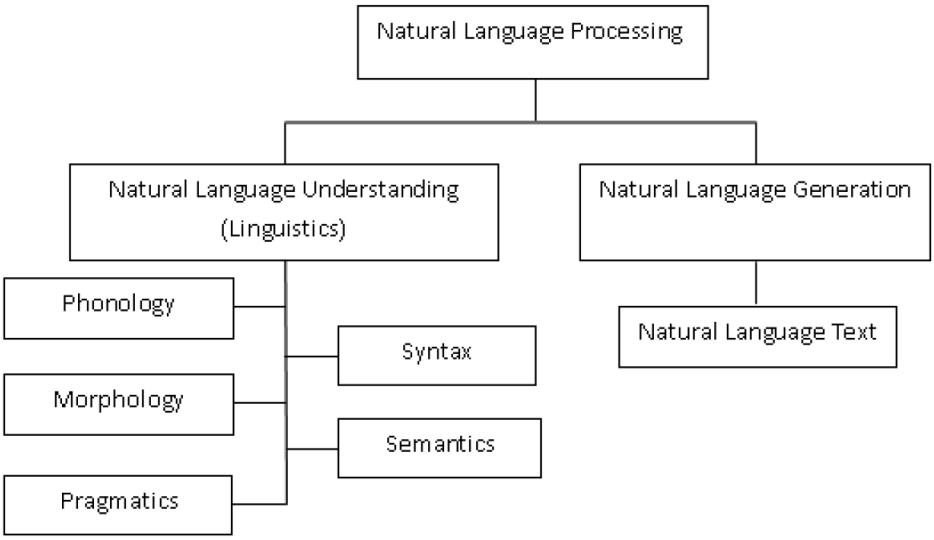

Figure 3.2: A basic map of natural language processing [13]

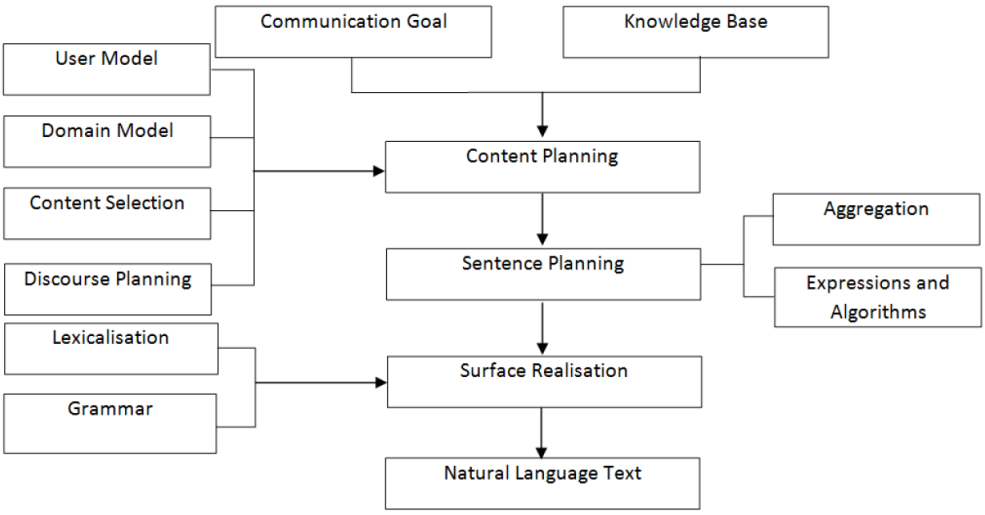

Figure 3.3: The phases of natural language processing [13].

3.5 Phases of NLP Architecture

Starting with content planning as shown in Figure 3.3, its goal is to set a concrete plan for the text overall. The topic or domain is set, and it takes all the necessary knowledge into account through content selection and sets a certain communication goal. The user's preferences, capabilities and knowledge level, are taken into account, and a discourse can be planned [13]. To resolve the text into a plan, it has to be broken down into many sentences that have to be aggregated from smaller fragments

and, through certain techniques and algorithms, generated from knowledge into concrete statements [13]. After a lot of sentences are generated, they are adapted to the user by lexicalization, finding the fitting words, and being structured with grammar rules to provide a natural language Text [13].

4 Large Language Models

The brief explanation of a model could be: numbers $x = \{x_1, \ldots, x_N\}$ and vectors $\mathbf{v} = \{\mathbf{v}_1, \ldots, \mathbf{v}_N\}$ are used as data points $(d_1, \ldots, d_N)$ to plot into a mathematical coordinate system distances and correlations of data [10]. All these probabilities $(p_1, \ldots, p_N)$ create the possibility for a large language model (LLM) to predict certain outputs by choosing the token with the highest percentage of likeliness. It is a statistical approach to compare it to the reasoning of the human brain [14].

This chapter gives an overview of methods and techniques used in combination with a neural network (NN) $NN = (V, E)$ to train a LLM. The architecture is explained, its encompassing set of calculations $\mathcal{C} = \{c_1, \ldots, c_N\}$, a brief introduction to common datasets $\mathcal{D} = \{d_1, \ldots, d_N\}$, preparation of data, and the LLM training itself.

4.1 Large Language Model Architecture

A LLM is an artificial intelligence (AI) created to process human language. It is made possible by NN specifically in the state-of-the-art transformer architecture (TA). LLMs can be classified into single-modal and multi-modal separating ones trained on a single type of data and others on multiple types of data, e.g. images. They are basically trained in two main phases, pre-training and fine-tuning, where fine-tuning means training for a specific downstream task. Communication with LLMs is done by prompting meaning to start a query to its knowledge base [14].

Specific calculations $\mathcal{C}$ are taking a crucial role in processing the data $\mathcal{D}$ and training the NN. To transform the input query sequence from natural language into something computable and efficient, the first step in the process is tokenization, beginning with the processing of the user's input, $\mathcal{Q} = [q_1, \ldots, q_N]$. Tokenization is a pre-processing task in LLM training that converts text into smaller pieces called tokens. $\mathcal{T} = [t_1, \ldots, t_N]$. A token t_i can be represented by a single character or symbol, or even by subwords. Schemes used for tokenization are, for example, *word-*

piece, *byte pair encoding*, and *unigramLM*. After these tokens are stored or passed into the TA, they need to be addressable and indexed by encoding positions [14].

At the start of TA, the inputs ought to be marked for later use, thus encoding positions are necessary in the TA because the TA processes inputs in parallel and independently of each other, but does not naturally possess positions for these inputs. This information is stored in embedding vectors $\mathbf{V} = [v_1, \ldots, v_N]$ that are added to the token embedding. After the embeddings $\mathbf{E} = [e_1, \ldots, e_N]$ are ready for using the main component of TA, the multi-head attention blocks in the encoder or decoder process them [14].
The tasks of the attention mechanism are to define the importance of specific tokens and filter the relevant ones and their dependencies on others. The following list provides different attention mechanisms in the TA [14]:

- Self-Attention: Defines attention scores with queries, keys, and values from the same section
- Cross Attention: In encoder-decoder architectures are used with encoder outputs into decoder as queries and the decoder evaluating with the inputs key-value pairs [14].
- Sparse Attention: Sliding windows are used for speed gains with sparse attentions iteratively computation of attention because for large sequences self-attention becomes infeasible with a time complexity of $O(n^2$ [14].

After evaluating the attention scores they have to be adapted for later use through normalization. The activation function is used in the feed forward neural network (FFNN) after the self-attention layer to help train the model by assisting in reducing the error of prediction, therefore taking a crucial part in curve-fitting the data [14]. The following two activation functions in Equations 4.1 and 4.2 are important and widely spread, the initial TA is based on *Rectified Linear Unit* ReLU which is used for the FFNN in TA [14]:

$$\text{ReLU}(x) = \max(0, x). \tag{4.1}$$

The $\max(0, x)$ function returns the maximum value between the arguments provided therefore all negative values will result in 0 delivering only positive numbers. x is the input value to the ReLU function.

This is in comparison to the more complicated function *Gaussian Error Linear Unit* GeLU [14]:

$$\text{GeLU}(x) = x \cdot \Phi(x). \tag{4.2}$$

x is the input value to the GeLU function. where $\Phi(x)$ is from the standard distribution the cumulative distribution [14].

The outputs of the FFNN are normalized by the layer normalization function in favor of increased accuracy. Layer normalization computes the results of the activation function and standardizes outputs by making the final result consist of a computation with the mean and variance as shown in Equation 4.3. It is a more efficient approach to minimizing the loss function, thus leading to faster convergence, and is an integrated component of transformers.
The following function displays the layer normalization used in GPT Models [15]:

$$\mathbf{H}^{(l)} = \text{LayerNorm}\left(\mathbf{H}^{(l-1)} + \text{SubLayer}(\mathbf{H}^{(l-1)})\right). \tag{4.3}$$

The $\mathbf{H}$ are the outputs of the layer and the previous layer, depending on the exponent. LayerNorm is a fraction of the mean and the variance from the outputs of the activation function, as displayed[16]:

$$y = \frac{x - \mathbb{E}[x]}{\sqrt{\text{Var}[x] + \epsilon}} \cdot \gamma + \beta. \tag{4.4}$$

- Residual connections: Outputs of each sub layer are summed up with their inputs. This helps with better gradients and closer correlation.
- LayerNorm: This is the layer normalization to stabilize the loss function.
  - Numerator: Calculates the mean of the activation function output
  - Denominator: Calculates the variance of the activation function output
  - Gamma and Delta: γ and β are learnable parameters [16].

After training takes place, the evaluation of the training results is determined by testing and interacting with the LLM.

To communicate with the LLM, prompting is necessary. Prompting describes the query for a response from the LLM, it is the way to communicate with the LLM and another technique for the NN to learn. By giving feedback to every prompt, the NN can adapt its weights and paths to the new input given by the prompt, which, on the one hand, removes the necessity for large-scale multi-modal data and enables task-specific learning [14]. The actual output of the LLM response is text completion. Text completion is the basic process used by LLMs to generate output. Out of likeliness to other texts, it will probabilistically choose the next token or word. This has the effect of no real reasoning or guarantee of truthfulness, it hardly depends on the quality of datasets, training, and other factors [14]. A crucial ability of the LLM is inference. Inference describes the ability of a trained Model to draw conclusions from unseen data [14].

Summarizing the architecture of LLMs, firstly, the TA is most efficient for LLM creation [9]. The LLM is the product of a trained NN on a large corpus of data, which adapted its weights in training time. It turns into a file of weights that can be accessed. These weights remember what the NN was taught. And those weights are able to predict the next word or token, choosing the highest probability. This text completion is not intelligent but more statistical [14].

4.2 Large Language Model Preparation

Training a NN hundreds of thousands of times, maybe a million times, and should certainly happen on quality datasets. The dataset not only has to be fitting for the resulting behavior, but it should also not be redundant and free from harmful and unnecessary information for the end user.

4.2.1 Data Preprocessing

Data preprocessing ensures the data fed into the NN is optimal for better results. Here are three example approaches: Quality filtering approaches are classifier-based and heuristic-based. Classifier-based approaches rely on high-quality data and afterward, require training in the prediction of the quality of texts. Heuristics-based sets of rules and statistics for the output texts serve as quality standards [14]. Data deduplication is necessary because duplicated data affects model performance negatively. Before training the LLM, the training data needs to be deduplicated in a preprocessing step at several levels, including sentences, documents, and datasets [14]. Privacy

reduction is crucial because one of the largest data sources for LLM training is the World Wide Web, which contains a lot of personal and private information that needs to be filtered, for example, names, addresses, and phone numbers [14].

4.2.2 Datasets

The evolution from pre-trained language models (PLM) to LLMs was brought about by increasing hyperparameters into billions and increasing training datasets of many terabytes. The more data, the better the LLM will perform, and the better the quality and diversity which are all crucial aspects of LLM Training. Next to handmade datasets, as mentioned, the World Wide Web is an excellent source for training data, e.g., Wikipedia[1] is used as a pretraining dataset in some cases [14].
Evaluating the limitations and proficiency is a crucial part of LLM training. Across various tasks, it should perform well and its ability to generate, interact with, and comprehend human language is measured. Mostly, two categories define evaluation and evaluation datasets. Firstly, natural language understanding (NLU), and secondly, natural language generation (NLG). Both indeed correlate with each other, so the borders may be marginal and are used interchangeably in literature [14].

4.3 Pre-Training

Pre-training can differentiate in tasks given to the model, but are nonetheless language tasks to train loss function calculation and more for different possible situations that can be encountered.

4.3.1 Pre-Training tasks

Full language modeling is predicting future tokens based on the input of previous tokens or autoregressive language objective [14]. Prefix language modeling uses randomly chosen prefixes with the loss calculation of just remaining target tokens, a non-casual training objective [14]. Masked language modeling is randomly masking spans of tokens with past or future context, where the model predicts the next token [14]. Unified language modeling means a non-casual, casual, and masked language training objective combined.

[1] https://wikipedia.com/

4.3.2 Distributed LLM Training

Many large corpses of data need to be efficiently fed into the NN to update loss functions and adapt weights. It's, of course, most efficient if the processing units (GPU, CPU) are leveraged and organized well. Therefore, different parallelism techniques are being used to achieve this goal. Data parallelism takes the model and makes copies on several devices to parallel train. While training the model, after each iteration, the weights of all devices get synchronized with each other [14]. Tensor parallelism is in a single layer across several processing units, the tensor or the multidimensional array gets chunked and processed in parallel, resulting in horizontal intra-layer parallelism [14]. Pipeline parallelism leverages the processing units by assigning each layer to one or more devices, thus creating vertical parallelism [14]. Model parallelism combining horizontal and vertical parallelism, or tensor and pipeline parallelism, results in model parallelism [14]. 3D parallelism occurs when in training data, model and tensor parallelism are combined [14]. Optimizer parallelism reduces the consumption of memory and communication costs by partitioning optimizer states, gradients, and parameters across devices [14].

4.4 Fine-Tuning

As displayed in Figure 4.1, fine-tuning takes a lot of steps until a model is finished for public usage. It is noticeable beforehand that for optimization, an adapter layer offers parameters that are adjusted while fine-tuning, not the neurons [14]. Transfer learning is a new, smaller task-specific dataset that is used on a pre-trained Model to further train it to increase performance. This fine-tuning is called transfer learning and is shown in Figure 3.1 [14]. Instruction-tuning means giving instructions on what to do when a specific query is asked, and this input and output are given as training data to enable an effective response. Multitask data is provided to ensure a lot of tasks are covered, and this type of fine-tuning improves zero-shot generalization, meaning the capability of a model to react to unseen data and the general downstream task performance. This instruction-tuning is depicted in Figure 3.1 and takes place after pre-training and before alignment-tuning [14]. Alignment-tuning with human feedback makes sure the LLMs are not generating false, harmful, or biased but rather helpful, harmless, and honest. This process involves querying unexpected responses and then aligning the parameters to the standards, thus avoiding them. The standard is *HHH*, meaning honest, harmless, and helpful, and a model is classified as an aligned model when it fulfills this standard. *Reinforcement learning*

with human feedback (RLHF) as shown in Figure 3.1, takes place after instruction-tuning and is part of aligning the model to human standards. It consists of reward modeling and reinforcement learning [14]. Reward modeling takes human preferences and trains a model using a classification objective to generate responses and rank them. To train the classifiers appropriately, the annotations are based on the *HHH* standard [14]. Reinforcement learning is an iterative process that is started in combination with the reward model to further align the model. The proximal policy optimization is applied by evaluating the ranked responses of the reward model into preferred and non-preferred [14].

After fine-tuning, it is efficient to analyze the model once again to filter out which neurons are not contributing to the generation of responses. Those can be removed; therefore, structured pruning takes place. Removing individual weights results in unstructured pruning.

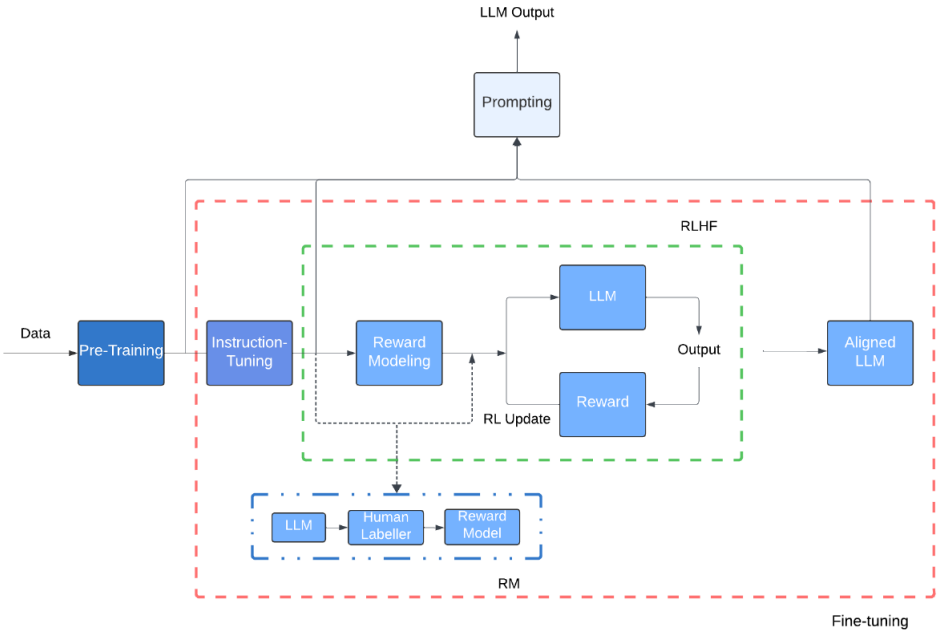

Figure 4.1: The basic training architecture for pre-training and fine-tuning [14].

5 Machine Learning

ChatGPT is a machine-learning (ML) Large Language Model (LLM) type Transformer used for Natural Language Processing (NLP). Machine Learning is the capability of computers to improve performance without being explicitly programmed to by experience through the evaluation of datasets.

5.1 Machine Learning with GPT

In the early 2000s, when ML emerged through technological advancements, the development of artificial intelligence (AI) through this technology emerged as well [4]. Based on a decade-old theory about Neural Networks (NN) and other related technologies, for the sake of global challenges, entrepreneurs, scientists, and developers started building software more competently than ever [4].
The infrastructure and training for ChatGPT are centered around transformer architecture (TA) with feed-forward neural network (FFNN) around all attention layers [9], encompassing a self-supervised pre-training and a supervised Fine-Tuning with Reinforcement Learning from Human Feedback (RLHF) [14].
Necessary for teaching a machine are learning algorithms, datasets with a lot of examples, and hyperparameters [17]. Learning algorithms on specific datasets, so the experience E of supervised learning, for the task T of, e.g., machine translation, will perform in some manner P. One type of learning algorithm is supervised learning, containing labels on the specific examples of the dataset. These labels are correctors helping, and tutoring the machine learning experience [17].
Supervised learning observes examples of random vector $\mathbf{x}$ and the result vector $\mathbf{y}$, to learn to predict $\mathbf{y}$ from $\mathbf{x}$, by estimating $p(\mathbf{y}|\mathbf{x})$ [17].
Pre-training of ChatGPT consisted of training with the Word Wide Web for example, that contains its labels in the dataset, which is referred to as self-supervised learning [15]. Fine-tuning in ChatGPT is done with supervised learning with RLHF to adapt the pre-trained model to human standards of honesty, helpfulness, and harmlessness and to specify the model for certain downstream tasks.

5.2 Neural Networks

The basic of all ML is a NN that tries to mimic our brain functionality. The human brain can, after a couple of attempts, recognize certain objects and name them properly. Even today NN needs several hundred thousand times training and iterations to be capable of achieving similar results in a specific task [2].

5.2.1 Neural Network Architectures

Neural Networks were introduced more than 70 years ago [2] but were limited by the computational power of the machines back then. A machine or Computer $\mathcal{C}$ operates on a machine language $\mathcal{ML}$, a set of commands $\mathcal{C} = \{c_1, \ldots, c_N\}$ and Memory to solve computational problems. The key to machine thinking is independent solving problems on different inputs without being programmed for specific cases [2].
These NNs contain Nodes processing information through concatenation of functions, using vectors and sets to determine in different ways the right outcome or a prediction about it. In addition, the NNs for LLMs have to predict the next word by probability. So an NN needs to understand global dependencies in the input sequence to know what fits next in one output sentence [9].

To understand ChatGPT one should understand not just Deep Learning (DL) for the DNNs, but the complete TA to get a grasp of what happens in pre-training, fine-tuning, and generally after a question is asked to the chatbot. From Rosenblatt's perceptron algorithm and the optimistic beginnings after the first digital computers in the 50s and 60s that until these promising last 10 years, there was not enough data, not enough memory and computing power to enable this brute force narrow artificial intelligence [2].
Today's state-of-the-art machines, operating with high-performance graphic processing units (GPU), can compute efficiently statistics and probabilities in enormous amounts, keeping track of vectors in large dimensions to provide a computational graph with dependencies and statistical correlation. These are used in artificial Neural Networks (ANN) the primitive analogy of our brain's functionality. In constant optimization of learning algorithms, parameters, and methods Models can be created, producing significant assistance in processing many language-based tasks.

Artificial neurons do have a similar structure to our biological neurons, likewise in Figure 5.1, and their dendrites are the inputs of nodes having a weight, which gets

adapted and corrected through training so the NN learns which outputs are correct and responsible paths over the weighted input nodes are used to further optimize the computed function for better predictions [2].

A NN can be seen as a computational graph with elementary units. By connecting each of them one gains a powerful network that could be trained on prediction for example. This ability to accurately compute functions of unseen inputs by training over a finite set of input-output pairs is referred to as model generalization [2].

A perceptron is the simplest form of NN. The Input nodes contain the important information for the calculation happening in the output node. These input nodes are changed and adapted in the training process to get more accurate predictions and correct answers. This output node can use a sign function calculating the classes or the semantics which the machine has to understand and predict by multiplying weights and features and then mapping it to either true or false in a range of 1 and -1 [2]:

$$\hat{y} = \text{sign}\{\mathbf{w} \cdot \mathbf{x}\} = \text{sign}\left\{\sum_{j=1}^{d} w_j x_j\right\}. \tag{5.1}$$

Equation 4.1 consists of the output of the function y. The sign function returns $+1$ for positive classifications and -1 for wrong classified. Vectors for weight $\mathbf{w}$ and input $\mathbf{x}$. In the sum over j = 1 until d the w_j represents individual weights, and x_j represents individual input values adding up to $\sum_{j=1}^{n} w_j x_j$, which computes the product of weights and inputs [2].

In Equation 4.1 the features are represented by the set X and the weights by W, the output y is the prediction. The sign function is the activation function, a choosable type of prediction. There are different activation functions for different machine

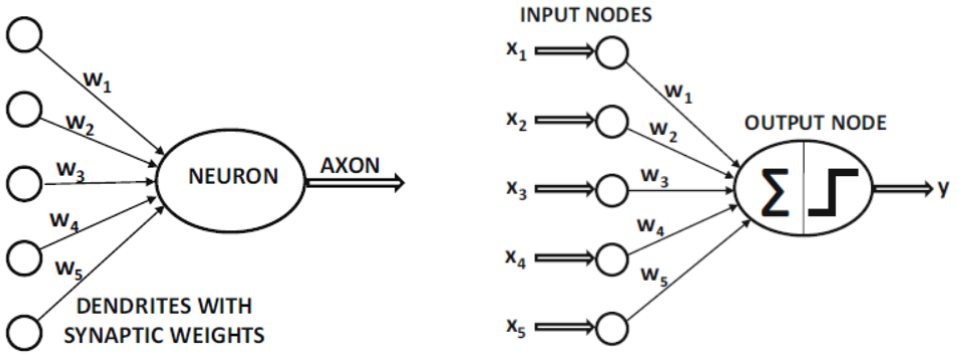

Figure 5.1: On the left a basic model of an artificial neuron, and on the right a perceptron [2].

learning purposes, e.g. logistic regression classifiers, support vector machines, and least-squares regression [2].

5.2.2 Scaled Dot-Product Attention

In the use of classification and weighting in ChatGPT exists another activation function. The classical sign function as presented is used in logistic regression classifiers, but the architecture of GPT is different it has a transformer and an attention mechanism and does not include the sign function as any activation function but rather with this following *Attention* method, and Rectified Linear Unit (ReLu) is used for the FFNN. The *Attention* in Equation 5.2 is used in TA is described as [9]:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}. \tag{5.2}$$

where:

- **Q** (Query) is the matrix of query vectors.
- **K** (Key) is the matrix of key vectors.
- **V** (Value) is the matrix of value vectors.
- d_k is the dimension of the key vectors.
- The softmax is a function applied to the scaled dot-product of the query and key vectors.

This mapping handles specific scaling so that the dimensionality and range of correlations stay relevant for further processing. The queries and keys of dimension d_k are first multiplied, then taking the dot-product scaling into account, and values of dimension d_v are lastly computed with the result of the softmax function which obtains the weights on the values [9].
This process of Equation 5.2 is depicted slightly differently in Figure 5.2, where MatMul, Scale, Mask, SoftMax and MatMUl are the calculations taken by the inputs. MatMul is a matrix multiplication with the sets **Q** and **K** as shown in Equation 5.2. Scale refers to the dot product $\frac{1}{\sqrt{d_k}}$ and Mask sets all invalid paths to illegal connections to $-\infty$ to ensure that only previous inputs or outputs are considered in encoding or decoding [9].
Commonly used attention functions are additive attention and the dot product. The TA uses the dot product function but with a scaling factor of $\frac{1}{\sqrt{d_k}}$. The benefit of

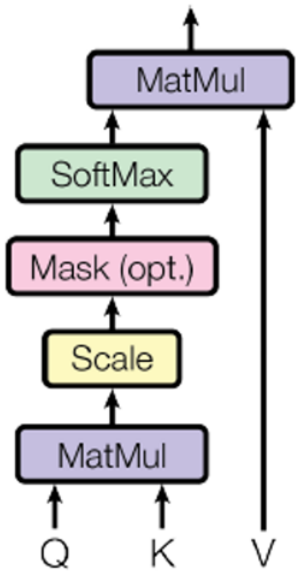

Figure 5.2: The scaled dot-product attention [9].

this scaled dot product above the additive attention, which needs a hidden layer for computation, is that it takes just highly optimized matrix multiplication to generate the proper outputs. Regarding larger values as inputs, the additive attention outperforms not scaled dot product attention, because without the fracture the numbers grow in magnitude [9].

5.2.3 Multilayered Neural Networks

Standard NNs including ChatGPT consist of a multilayered NN, a FFNN using backpropagation to adapt the paths for training with a loss function calculated and optimized at the output nodes. Feed forward stresses the fact that information is *fed forward* successively into the upcoming layers [2].
Figure 5.3 shows the basic construction of multilayered NNs with several input nodes at the start connected to all hidden layers. In this case, there are two hidden layers all connected to one output layer which depends on the architecture containing several output nodes. The hidden layer derives its meaning from the fact that its computing is not available to be seen by the user [2].

5.2.4 Training a Neural Network with Backpropagation

Regarding the earlier layers, there are problems in loss calculation. To solve this problem, a gradient of the loss function is calculated using the backpropagation

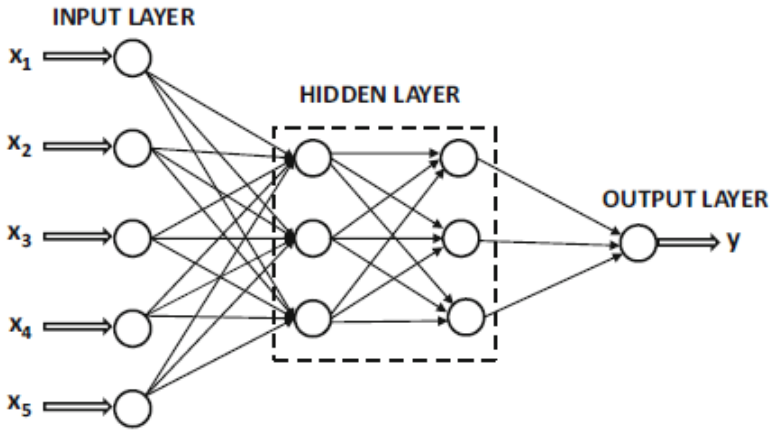

Figure 5.3: The multilayered neural network with input, feed-forward network, two hidden layers and output nodes [2].

algorithm. This cross-entropy loss function describes respectively the difference between the desired output and the real output presented by the results of the softmax probabilities. The backpropagation algorithm uses the chain rule to compute the error gradients. This is a sum of local-gradient products over the paths of the NN. The initial phase of the backpropagation is the forward phase of the FFNN, solving in cascade in the hidden layers all calculations with the current set of weights until the output layer is reached. The gradients computed with the loss function in consideration of all previous inputs now need a backward phase to compute gradients for the weight updates [2].

Now with the local gradients computed in the process of output to input, the NN is learning by adapting. The chain rule allows an overview of the effect of small changes in the network on the result. These gradients are adaptions between the current prediction and real value and get updated in this backward phase, so the NN is learning what it has to predict for next time. All this would linearly be complex and to cover all the paths it would be exponential costs, but it is actually dynamically programmed, resulting in a cost-efficient process. Prerequisites for this dynamic program are an acyclic graph and are based on recursion [2].

Regarding the extension, the question $\mathcal{Q} = \{q_1, \ldots, q_N\}$ with q_n words will be fed into the multi-head attention which is fully connected to FFNN Graph $FFNN = (V, E)$. The words q_i there will be transformed from embeddings represented by vectors into attention scores. These words q_i are split into tokens $\mathcal{T} = \{t_1, \ldots, t_N\}$ or a pair of tokens t_i. After contextualization, which is done by the transformer attention

mechanism, to create relationships between these embeddings, the TA computes every step a set of probabilities $\mathcal{P} = \{p_1, \ldots, p_N\}$ corresponding to each token t_j. These numbers were mutated into probabilities by the softmax function to calculate the importance of the specific next word [9].

5.3 Transformers

TA is the basis for NN to be trained with, resulting in LLMs trained faster and scoring better. The strength of NN with TA over recurrent NN or others lies in the possibility to process input in parallel. The effect is brought by the self attention mechanism. Self attention was already invented and the concept was used in other architectures, nevertheless, none of these concepts relied on attention mechanisms only [9].

5.3.1 Transformers Basics

Self attention resides in the multi-head attention block and relates to all positions of a sequence at the same time and it computes scores for a single representation of this sequence regarding all positions [9]. Using only self attention to compute representations of input and output took away the necessity of recurrent NNs and convolution and enabled parallelization [9]. The encoder-decoder structure takes input sequences $(x1, ..., xn)$ and maps them into scores $z = (z1, ..., zn)$ for further processing. In the finishing process, the output sequence $(y1, ..., ym)$ is generated autoregressive, just relying on the previous input. This goes against the computation of the mulit-head attention that takes all information into consideration with the mechanism of masking disabling the illegal connection in the matrix [9]. The structure enhances to stacked self attention and point-wise, fully connected layers for the encoder and decoder. This is shown in Figure 5.4: On the left the encoder stack and on the right the decoder can be seen [9].

5.3.2 Architecture

The architecture shown in Figure 5.4 depicts two stacks of different layers. On the left with the inputs is the encoder, and on the right is the decoder. The encoder is responsible for providing computed data containing the context of the query for the decoder to calculate the output. Positional encoding is necessary to give a relative or absolute position to the embeddings. Because this transformer works

with parallelization, there is no track of position; thus, positional encoding adds a vector to the embedding to generate a position. Multi-head attention is the core of the TA, with the scaled-dot product calculating the attention scores working in parallel on several layers, which are normalized and fed into the fully connected FFN. There are, lastly, the FFN, several computations normalization and the generation of output probabilities [9].

5.3.3 Embeddings and Softmax

At the beginning of the TA, are inputs the query from the user. These inputs have to be converted into mathematical expressions to be further processed. These conversions are called embeddings and are represented as vectors with a specific dimension provided by the hyperparameter d_{model}. They are further summed up with positional encoding to reference each vector a positional vector [9]. After the processing of the FFNN from the decoder stack and after normalization at the end of the TA, the vectors will be transformed into probabilities. This happens through a simple linear transformation and, again, applying the softmax function [9].

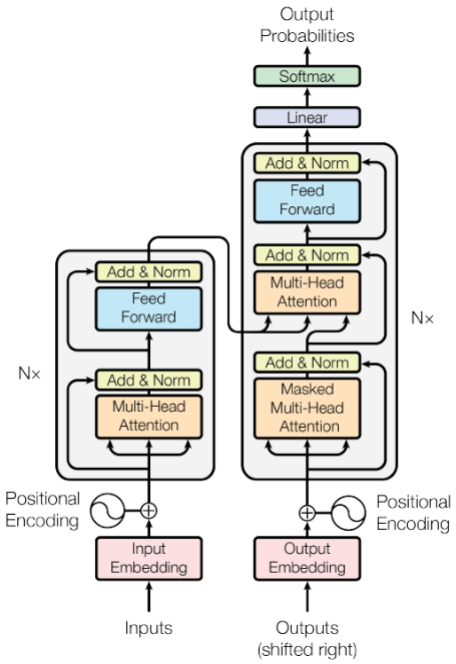

Figure 5.4: The initial transformer architecture [9].

5.3.4 Positional Encoding

The TA does not contain recurrence or convolution, leaving the incoming sequence of the inputs without positioning. This information about relative and absolute positioning is relevant and must be injected in another way [9]. To solve this issue, positional encodings are added to the input embeddings directly at the start after conversion from text to vectors. These positional encodings have the same dimensionality as the hyperparameter d_{model} and can be summed up easily [9].

The following equations are used for generating numbers that are then put together into a vector, thus providing a positional encoding:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{\text{model}}}}}\right). \tag{5.3}$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{\text{model}}}}}\right). \tag{5.4}$$

The positional encoding generated for each position *pos* will be the same for similar dimensions. For example, position 0 always has the vector $\mathbf{v} = (0, 1, 0, 1)^T$ when the dimension $d_{model} = 4$ is chosen [9].

5.3.5 Encoder and Decoder

The encoder consists of x identical layers, each containing two sub layers. Starting with a multi-head self attention mechanism following the point-wise fully connected FFN. Residual connections are used for the two sub layers to be fully connected and in between the outputs are normalized by layer normalization. The dimensionality stays consistent for sub-layers, and embedding layers, they compute outputs for dimension d_{model} [9].

The decoder is composed of x identical layers but with three sub layers each. Two self attention layers followed by one FFNN that computed the final output of the decoder. This additional layer computes again multi-head attention over the output of the encoder. Around all sub layers exist residual connections with outputs computed by layer normalization. To prevent calculation on subsequent positions, the self attention in the decoder had to be modified. This is masking with the outputs already on positional offset by 1 again secure the auto-regressiveness of the computations of the decoder [9].

5.3.6 Multi-Head Attention

The Multi-head attention consists of 3 parts as shown in Figure 5.5, linear projection of the inputs to enable parallelization, instead of computing a single attention with d_{model} dimensional queries, values, and keys [9]. The second part where those inputs are then computed by the mentioned scaled dot-product attention h times with the inputs d_q d_k d_v. Those h computations are also called heads. Lastly, the results get concatenated and again linearly projected [9]. Nevertheless, instead of having more computational costs for multiple computations, the fact that each input dimension is equal to $d_k = d_q = d_v = d_{model}/h$ converts the costs to be similar to a single head computation with a model dimension [9].

This Figure 5.5 is depicted as the following Equation 5.5 [9]:

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \\ \text{where} \quad \text{head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V). \end{aligned} \tag{5.5}$$

The last operation is the linear projection with W^O. Before that the concatenation of all the head computations $\text{head}_1, \ldots, \text{head}_h$) takes place. These are the specific results of the scaled dot-product attention. These were linearly projected to deliver optimized inputs QW_i^Q, KW_i^K, VW_i^V thus completing the Figure and equation equality [9].

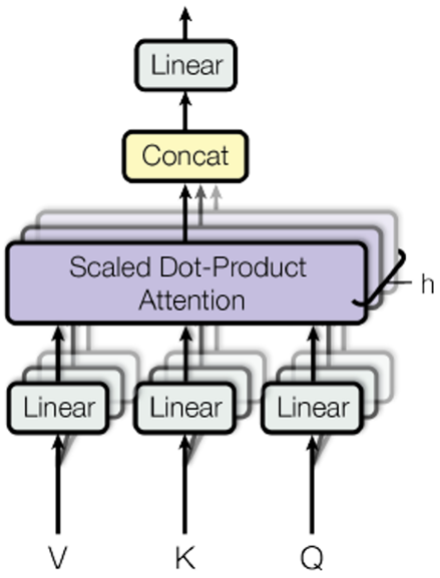

Figure 5.5: The multi-head attention [9].

5.3.7 Position-wise Feed-Forward Networks

The TA consists of many sub layers, and there are two types of sub layers. All self attention layers with multi-head attention are fully connected to the other sub layer type, the FFNN is applied to each position identically and separately [9].
The following extension of the ReLu is used as an activation function in the GPT TA to obtain a result for the FFNN [15]:

$$\mathrm{FFN}(x) = \mathrm{ReLU}(x\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2. \tag{5.6}$$

$\mathbf{W}_i$ a trainable weight input matrix for the ReLu function or product with its outcome. $\mathbf{b}_i$ a bias vector used, to sum up weight inputs or the sum with the outcome of ReLu [15]. This consists of two linear transformations with a ReLU activation in between and has different parameters in each layer.

5.3.8 Applications of Attention

There are three different positions where multi-head attention is processing the further passed input until decoding the output. The encoder self attention layer contains all the queries, keys, and values from its previous layer, and all previous layers of the encoder can be reached by each position in the encoder enabling to capture of global dependencies of the input [9].
Regarding the Decoder up until a position, the self attention layers can attend all previous positions in the decoder. This allows leftward information flow, but this is hindered by the implementation of masking in the scaled-dot product that sets all illegal connections in the softmax input to $-\infty$ thus preserving auto-regression [9].
The last multi-head attention layer is the encoder-decoder attention with the queries from the previous attention layer of the decoder and the value and keys from the encoder's FFNN output. This enables capturing all global dependencies in the decoder for all positions in the input sequence [9].

5.3.9 Benefits of Self Attention Layers

To prove that not only does TA score better and needs less time to be trained, as already shown, Table 5.1 depicts several aspects of 3 different architectures to prove that the general computational costs for the TA are more efficient than the other layer types. To evaluate the self attention layers with recurrent and convolutional layers, first complexity per layer is compared, then sequential operations, and finally

maximum path length [9].

One is the total computational *Complexity per Layer*. As shown in Table 5.1, where compared to TA $O(n^2 \cdot d)$, recurrent layers do cost less for a great number of sequential lengths n with $O(n^2 \cdot d)$. Nevertheless, both outperform convolutional layers, which are exceptionally costly [9].

The sequential length n is most often smaller than d the representation dimensionality, thus giving the computational complexity of the self attention layers an advantage because they perform faster than recurrent layers when the sentence representation is used. This is indeed the case for most state-of-the-art models in machine translation for example *word-piece* and *byte-pair* [9].

In comparison, the *Sequential Operation* necessary to perform computation on the sequence of the self attention layers outperforms the recurrent layers with $O(1)$ to $O(n)$, which allows more computation to be parallelized [9].

In sequence transduction tasks, learning long-range dependencies in networks is crucial for the model and is represented by the third category *Maximum Path Length* [9]. The path for signal traversing forward and backward the network indicates the ability to learn long-range dependencies faster and more efficiently. Those paths should be as short as possible between any input and output position. Self attention layers show $O(1)$ operational time in comparison with recurrent layers $O(n)$ [9].

Layer Type	**Complexity per Layer**	**Sequential Operations**	**Maximum Path Length**
Self attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(\log_k(n))$

Table 5.1: A comparison of layer types [9].

6 GPT-Based Chatbot for University Preboarding

This thesis presented the history of chatbots and the underlying technologies used to build large language models (LLM). The open-source community and Meta Platforms Inc. provide such LLMs for everyone to use even for business purposes. These models, when queried with prompts, deliver impressive results, for example, the Llama 3.1 was recently released. In this GPT-based chatbot for university preboarding are 3 different solutions presented. First, the extension programmed for this thesis is downloadable and runs with a connection to a local LLM. Second, the scenario of such a simple application being integrated into the university website as conversational AI with already-fed information about the university. The third scenario is the usage of ChatGPT through the extension provided, where the currently accessed and necessary information gets prompted to OpenAI.

6.1 Introduction to Extension

The first solution is the extension itself shown in Figure 6.1. It requires some settings to be done beforehand. Nevertheless, it also requires strong computing units to be efficient in usage. The second option is university server-based, hypothetical, and will not be measured in the evaluation. The third option needs an OpenAI-Key to provide the necessary functionalities to converse with ChatGPT over the extension. Universities possess a lot of content that contains the necessary information for the student's future choice of study. Nevertheless, other visitors can also have intentions of finding answers on university websites. Using the LLM ChatGPT, or another bot, one can surely find useful information presented by this chatbot. But in its raw state, it is not practical for querying specific sites and their data, because by this point it does not analyze all the media on this page, it provides location and explanation to web pages.
Probably there will never be such a functionality that the certain chatbot is analyzing

all the media of a site, because the costs would be greater than necessary. Providing this extension, should simplify and create efficient answers by specifying the current context of the visited web page.

6.1.1 Available Content and Potential Users

The internet in general provides many different types of media, such as documents, images, audios/videos, texts, archive files, and many others. In this case, the extension is a proof-of-concept to be able to analyze portable document files (PDF), which are on university websites most used and contain important information.
Potential users are primarily young people who are interested in starting studies at a specific university. They are searching through the catalog of many faculties and degree programs to get an overview of the offerings. This overview is presented by the specific course catalogs for each degree program. In most cases, just one-degree program can have over 20 different modules to offer, which one needs to analyze throughout.

6.1.2 Available Solutions

The three solutions are local LLM through extensions, server-based processing of LLM data integrated on the university website, and extension-based OpenAI data processing.
The first solution is available by downloading an extension for the Chrome browser that enables a computer's fast prompting on any website as depicted in Figure 6.3. It opens with 2 shortcuts where "ALT+A" enables the direct listing of available PDFs

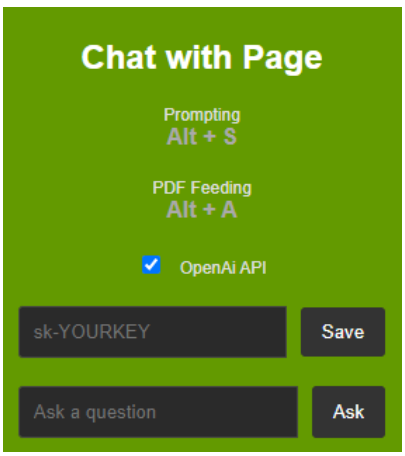

Figure 6.1: An extension popup for university preboarding.

Figure 6.2: An example of a floating button to access prompting window.

on the page to instantly feed to the desired LLM. ”ALT+S” enables the prompting Window. The prerequisites for this local LLM extension solution are a downloaded LLM running on a specific port that has to be registered into the extension, and after that, one should be able to communicate with this LLM. The extension should handle state management and all prompting settings.
The third solution is integrated into the extension if one wants to change from the local LLM to ChatGPT access after registering the OpenAI-Key in the specific input, and with the same inputs as shown in Figure 6.3, there is prompting available.
The second solution would require a chatbot floating button on the university web page to access the university LLM, as depicted in Figure 6.2. This solution provides LLM that could be prepared with the information available on the website and could be used for other use cases as well.

Figure 6.3: The Prompting Window left and PDF-Window on the right.

7 Evaluation

The evaluation of a chatbot depends on many factors, such as performance in Natural Language Processing (NLP), therefore testing the model itself for correctness, robustness, and authenticity. And also evaluating the performance statistics of the chatbot by tracking the processing time. Development of a Generative Pretrained Transformer (GPT)-based chatbot for university preboarding with the technological advancements of 2024 has simplified the implementation of a fully functional chatbot over a local Large Language Model (LLM) by providing easy tools to implement a LLM with a vector database to store and ingest new information quickly from Portable Document Format (PDF) gained through prompting on the university websites to enable memory of the chatbot, thus providing a chatbot to process and retain knowledge.

7.1 Criteria of Chatbot Evaluation

Criteria for Chatbot Evaluation can be deeply analyzed for solving specific tasks, observer behavior, statistics, and many more. For university preboarding, there are specific requirements necessary to enable a user-friendly and helpful chatbot over the university website and data. The chatbot's primary task is to give answers to general questions about the university, to guide the visitor through the page successfully, and to analyze all page's content for more questions to answer. This should be done quickly, reliably, and friendly. Most of the tasks mentioned are done by the underlying LLM chosen to interact with. If one chooses a reliable LLM e.g., the Meta AI (LLaMA) 3.1 currently available for public use provides all the speed, reliability, and friendliness necessary. The following criteria are more detailed and refer to the extension itself implemented for this thesis, where the chatbot is integrated.

The usage of the chatbot should be user-friendly.
The UI should be easy to interact with. The design should be intuitive, not complicated. New users should instantly be able to use it. Responses should be as human-like as possible. There should be general satisfaction in users chatting with chatbots. Chatbot is customizable to certain user types.

The response statistics should be as optimal as possible.
Context correctness assures the correct retaining of information previously fed. Precision to measure the correctness of the response regarding the user's desired output from his query. Succession rate is the rate of queries completed. Response time measures the amount of time necessary to answer a query. Robustness means the chatbot should be able to handle partially wrong inputs and mistakes.

Fulfilled technical criteria indicate a mature application.
Scalable to ensure the chatbot can be used by many users at the same time. Security for the data of users. System adaptation means the chatbot can be used on different systems and platforms uniformly. Error-free applications work without crashing or bugs.

7.2 Results

The results are split into OpenAI integration and local LLM integration because the hypothetical server-based solution is not available for testing. The usage criteria are the same in both integrations because both models are accessed through the same extension.
The evaluation of the chatbot extension developed for university preboarding depends on many criteria, including the LLM chosen, the environment where the LLM is maintained, and the user's computer. The extension itself does not provide much functionality,in decreasing or increasing the quality of response, and its user interface is simple and accessible.

Regarding Usage:
The extension could be available in the Chrome Extension store or ultimately just for the university embedded into the website. The extension itself consists of a popup for registration of OpenAI keys and two windows accessible through the keyboard to access the *Prompting Window* and the *PDF-Selection Window*. These windows are

simple, intuitive, and easy to interact with. The responses are shown in a separate response box above the input in the ”Prompting Window”. All responses in both variations are readable, understandable, and human-like.

Regarding response statistics:
Context correctness: The local LLM is in its basic use without context retention, therefore it forgets the previous query and is just able to respond to a query. The necessity of Context correctness urges the basic approach to be enhanced by context management. There are several options to solve this, a normal list with previous answers or a vector database for efficient storing and retrieval. After creating such a state management system, most of the LLMs would be able to chat properly and be useful for extension and preboarding.
OpenAI shows full context correctness over many requests in one chat.
Precision The precision on successful queries shown in Table 7.1 is the same for all successful queries, suggesting that there are no direct law studies yet, but law science is something similar at the university.
On the other hand, OpenAI queries were ...
Succession rate: Most of the queries succeeded, as shown in Table 7.1. In the unsuccessful ones, the model interpreted ”tu do” as French or the unsuccessful queries as a kind of joke.
Response time: Table 7.1 shows the time for the local LLM to answer completely in the extension, the extension itself uses a Backend and needs time for processing. The responses arrive close to one minute after which is not operable and feels compared to 1 to 3 seconds of ChatGPT responses long. Response times of OpenAI requests are all around 1 second which is operable to work with.
Robust: Table 7.1 shows that the not-successful prompts show that the LLM is not entirely robust. On the other hand, these queries are rather hard to understand and fragile and short. The control results with OpenAI show that the same fragile

Prompt	**First Try**	**Second Try**	**Successful**
Can one studie Law at Tu dortmun?	52s	1m 03s	true
Is possible Tu dortmund law study?	54s	53s	true
Is there law at tu do ?	22s	15s	false
tu do law ?	31s	25s	false
program law at tu do is that possible ?	30s	48s	true
so I wanted to ask if you know the ...	48s	51s	true

Table 7.1: Various prompts on local LLama3.1 regarding studying law at TU Dortmund.

Prompt	First Try	Successful
Can one studie Law at Tu dortmun?	1s	true
Is possible Tu dortmund law study?	1s	true
Is there law at tu do ?	1s	false
tu do law ?	1s	false
program law at tu do is that possible ?	1s	true
so I wanted to ask if you know the ...	1s	true

Table 7.2: Various prompts on OpenAI ChatGPT-4 regarding studying law at TU Dortmund.

queries were not understood by OpenAI as well.

Regarding technical criteria:
Security concerns are handled by OpenAI itself, and on the local LLM, the local computer itself should be safe to keep the data on it safe as well.
System adaptation for the chatbot as an extension should work for computers, not mobiles, and was developed for the "Google Chrome" extension store. Integrated into the website, it should work on all devices able to enter the university website.
Scalability depends on the device on which the LLM is running; a local LLM is not scalable because just the computer itself will always be the only user. If it runs on the university server, it would be as scalable as the server itself allows, and the OpenAI integration should not be problematic because there are not millions of users visiting university websites simultaneously.
Error-free the extension itself could cause errors, the LLMs should be error-free except for misleading or wrong responses which depend on the input.

7.3 Comparing to other Chatbot Alternatives

Alternatives currently available would be to use the free plan of ChatGPT and feed the wanted PDF or other files directly into ChatGPT to get the information about them. This would not be inconvenient; one needs to download and paste it into another browser tab, which should not take much effort and time. After the free plan is used up, there are currently no alternatives except for Anthropic Claude or other online chatbots to process information from a file again on their free offerings. The OpenAI integration of the preboarding extension needs an OpenAI-Key that has available credits, which costs money. The free versions would be GPT4all or other already-developed applications from GitHub. These can process PDFs as well,

are completely free, and should be easy to set up. One would need to download and ingest them into these applications and could receive an answer to questions asked which may or not be inconvenient because reading it would sometimes be faster and more reliable.

8 Summary

8.1 Summary

The GPT-based preboarding chatbot for the university, which is fully integrated into the university website would be a useful offering for visitors to interact with instead of searching and browsing through all the sites. Nevertheless, searching and browsing through the university pages can offer a brief or more detailed overview of other offers the visitor would maybe not have discovered if they used the chatbot to directly enter some page on his search. On the other hand, for analyzing the contents of a page the chatbot offers much assistance to save time searching through files, tables, and texts. The performance of a fully integrated chatbot working on a university server in comparison to an extension is significant because the Large Language Model would often be fed with similar or the same questions, files, and texts, thus developing a faster response. Overall the study of the development of a GPT-based chatbot for university preboarding showed that it is a worthy investment to make and with the available technologies much easier to develop than ever before.

8.2 Outlook

Regarding the future development of artificial intelligence (AI) and chatbots, a very important feature could be personalized chatbots, which were discussed in the thesis [4]. The fully personalized chatbots are technology directly assisting personal development. Integrated with personal data, it learns with the person and can assist in many areas, from education to personal growth. It can answer many questions for the user, thus guiding the user as a personal coach, tutor, or mentor. Such life-encompassing conversational AI integration will be in many areas, such as health, environment, and business [10]

Declaration on the Use of Tools and AI-Assisted Writing Tools

I assure that when using IT/AI-assisted writing tools, I have fully listed these tools with their product names and my source of reference in the following overview of the aids used and/or have marked the respective text passages in the work as composed with IT/AI-generated assistance.

I am aware that deception or attempts at deception will be penalized according to the examination regulations applicable to me.

The following aids and AI-assisted writing tools were used for the processing:

- Chatbot, ChatGPT, 4o, ”Generation of Bibtex, Generation of mathematical latex(matrices, tables, equations, equation explanations)”, ”https://chatgpt.com/”
- Chatbot, Claude, 3.5, ”Generation of Bibtex Citations, Generation of mathematical latex(expressions, matrices, tables, equations, equation explanations)”, ”https://claude.ai/”
- Orthography corrector, LanguageTool, ”improving my mistakes in grammar and orthography in general”, ”https://languagetool.org/de”
- Orthography corrector, Grammarly, ”improving my mistakes in grammar and orthography in general”, ”https://www.grammarly.com/”

List of Figures

Bibliography

[1] AI Index Steering Committee. Introduction to the ai index report 2024, 2024.

[2] Charu C. Aggarwal. *Neural Networks and Deep Learning*. 2018.

[3] Storius Magazine. Transformers are here: Gpt explained, 2024. Accessed: 2024-05-28.

[4] Md. Al-Amin, Mohammad Shazed Ali, Abdus Salam, Arif Khan, Ashraf Ali, Ahsan Ullah, Nur Alam, and Shamsul Kabir Chowdhury. History of generative artificial intelligence (ai) chatbots: Past, present, and future development. *University of Massachusetts Lowell*, 2023.

[5] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, YiFan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models. *CoRR*, abs/2303.18223, 2023.

[6] Luke Swartz. Why people hate the paperclip: Labels, appearance, behavior, and social responses to user interface agents, 2003.

[7] Tulie Finley-Moise. What's the computing difference between a teraflops and a petaflops, 2019.

[8] Mingyu Zong and Bhaskar Krishnamachari. a survey on gpt-3, 2022.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[10] Gokul Yenduri, M. Ramalingam, G. Chemmalar Selvi, Y. Supriya, Gautam Srivastava, Praveen Kumar Reddy Maddikunta, G. Deepti Raj, Rutvij H. Jhaveri, B. Prabadevi, Weizheng Wang, Athanasios V. Vasilakos, and Thippa Reddy Gadekallu. Gpt (generative pre-trained transformer)— a comprehensive review

on enabling technologies, potential applications, emerging challenges, and future directions. *IEEE Access*, 12:54608–54649, 2024.

[11] Microsoft. Bleu score. Accessed: 2024-05-30.

[12] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020.

[13] Diksha Khurana, Aditya Koli, Kiran Khatter, and Sukhdev Singh. Natural language processing: State of the art, current trends and challenges. *Journal of Natural Language Processing*, pages 1–15, 2023.

[14] Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. A comprehensive overview of large language models, 2024.

[15] M. Lee. A mathematical investigation of hallucination and creativity in gpt models. *Mathematics*, 11(2320), 2023.

[16] PyTorch. torch.nn.layernorm, 2023. Accessed: 2024-07-18.

[17] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep learning, 2016.