

**CookBERT – Adapting BERT for the Cooking Domain**

Bachelor thesis in Media Informatics at the Institute for Language, Literature and Cultural Studies (I:IMSK)

Submitted by: Pascal Strobel

Address: Gluckstraße 3, 93053 Regensburg

E-Mail (university): pascal.strobel@stud.uni-regensburg.de

E-Mail (private): paschistrobel@web.de

Matriculation number: 2106133

First reviewer: Prof. Dr. Udo Kruschwitz

Second reviewer: PD Dr. David Elsweiler

Supervisor: Alexander Frummet (M. Sc.)

Current semester: 7

Submitted on: 30.2.2016

Contents

[1 Introduction 8](#_Toc98010590)

[2 Related Work 11](#_Toc98010591)

[2.1 Conversational Artificial Intelligence 11](#_Toc98010592)

[2.1.1 Introduction 11](#_Toc98010593)

[2.1.2 Inner Workings 13](#_Toc98010594)

[2.1.3 Application in the Cooking Domain 13](#_Toc98010595)

[2.2 Language Processing with Deep Neural Networks 14](#_Toc98010596)

[2.2.1 Word Embeddings 14](#_Toc98010597)

[2.2.2 Artificial Neural Networks 16](#_Toc98010598)

[2.2.3 Recurrent Neural Networks 19](#_Toc98010599)

[2.2.4 Encoder-Decoder 21](#_Toc98010600)

[2.2.5 Attention 22](#_Toc98010601)

[2.2.6 Transformers 24](#_Toc98010602)

[2.2.7 Transfer Learning 27](#_Toc98010603)

[2.3 BERT 29](#_Toc98010604)

[2.3.1 Inner Workings 30](#_Toc98010605)

[2.3.2 BERT in Conversational AI 31](#_Toc98010606)

[2.3.3 Domain Adaption 32](#_Toc98010607)

[2.4 Cooking datasets 32](#_Toc98010608)

[2.4.1 RecipeNLG 32](#_Toc98010609)

[2.4.2 Cookversational Search 33](#_Toc98010610)

[2.4.3 DoQA 33](#_Toc98010611)

[2.4.4 FoodBase 34](#_Toc98010612)

[2.5 Summary and Key Differentiators 35](#_Toc98010613)

[3 Methodology 36](#_Toc98010614)

[3.1 Preparing the Data for Domain Adaptive Pre-Training 36](#_Toc98010615)

[3.2 Analyzing Domain Similarity 38](#_Toc98010616)

[3.3 Domain Vocabulary Insertion 39](#_Toc98010617)

[3.4 DAPT 40](#_Toc98010618)

[3.5 Implementation Details 41](#_Toc98010619)

[3.6 Finetuning 41](#_Toc98010620)

[3.6.1 Intent Classification 41](#_Toc98010621)

[3.6.2 Named Entity recognition 43](#_Toc98010622)

[3.6.3 Question Answering 43](#_Toc98010623)

[4 Evaluation 43](#_Toc98010624)

[4.1 Multi-class Classification 43](#_Toc98010625)

[4.2 Named Entity Recognition 44](#_Toc98010626)

[4.3 Question Answering 45](#_Toc98010627)

[5 Discussion 45](#_Toc98010628)

[6 Limitations 46](#_Toc98010629)

[7 Conclusion 47](#_Toc98010630)

[Bibliography 48](#_Toc98010631)

[Anhang A: Bausteine wissenschaftlicher Arbeiten 56](#_Toc98010632)

[A1 Theoretische Arbeit 56](#_Toc98010633)

[A2 Konstruktive Arbeit 56](#_Toc98010634)

[A3 Empirische Arbeit 56](#_Toc98010635)

[Erklärung zur Urheberschaft 58](#_Toc98010636)

Abbildungsverzeichnis (optional, in der Regel nicht notwendig)

[Abbildung 1: Blumen (Quelle, Jahr, Seitenzahl) **Fehler! Textmarke nicht definiert.**](#_Toc359834290)

Tabellenverzeichnis (optional, in der Regel nicht notwendig)

[Tabelle 1: Empfohlener Textumfang **Fehler! Textmarke nicht definiert.**](#_Toc359834281)

**Zusammenfassung**

Abstract

* Recent Fortschritt in NLP hat sich auch auf CA ausgewirkt.
* While Cas were rule-based früher, sind sie heute größtenteils basierend auf neural networks
* Vor allem das von Google vorgestellte BERT Besonders huge pretrained neural network language models like BERT haben aufgrund ihrer herausragenden Performance die Aufmerksamkeit auf sich gezogen und warden für unterschiedlichsten Aufgaben in Cas eingesetzt
* viele Cas warden in ganz bestimmten kontexten eingesetzt
* In der Literatur hat sich aber immer wieder gezeigt, dass BERT domänenspez Wissen fehlt und somit seine Performance limitiert ist.
* In dieser Arbeit wird CookBERT vorgestellt, ein domänenspezifische BERT, der via DAPT an die Kochdomäne angepasst, um die Performance von downstream tasks von gleichdomänigen Cas zu verbessern
* Die Arbeit zeigt, dass CookBERT promising results erzielt und das standard BERT in zwei von drei Cas relevant tasks der cooking domain significant outperformed, was die Effizient der Domänenadaption unterstreicht und zu weiterer Forschung für andere Domänen, sowie die Integration von CookBERT in ein tatsächliches System, motiviert.
* Irgendwie die Kochdomäne reinbringen

# Introduction

Conversational agents (CAs) like Amazon’s Echo[[1]](#footnote-1) or Apple’s Siri[[2]](#footnote-2) become more and more pervasive and are applied in a broad range of contexts, including health (Ni et al., 2017; Xu et al., 2019), elderly care (Bickmore et al., 2005), education (Graesser et al., 1999; Winkler et al., 2020), customer service (Cui et al., 2017) and home cooking (Angara et al., 2017; Chu, 2021). Although there are various types of conversational agents, which are titled and categorized very inconsistently in literature and media, they all provide an alternative to traditional methods for humans to seek for information by making the search process more conversational, mainly via written or spoken natural language (McTear, 2020, pp. 12–13). Users benefit from this more natural interaction as it promises an increased ease of use and speed of user requests as well as a convenient usage (Brandtzaeg & Følstad, 2017). While early approaches to create CAs were mainly based on handcrafted rules (e. g. Weizenbaum’s ELIZA (1966)), this has shifted in recent years towards the utilization of large-scale pretrained language models which can gain a superb grasp of human language.

One of the most promising models in recent development is Bidirectional Encoder Representations from Transformers (BERT) - a huge neural network proposed by the Google AI team (Devlin et al., 2018) which is pretrained on 3.3 billion words from BooksCorpus (Zhu et al., 2015) and English Wikipedia. It builds upon previous approaches on pretraining contextual representations (Dai & Le, 2015; Howard & Ruder, 2018; Peters et al., 2018; Radford et al., 2018), but what really sets it apart is that it’s “the first deeply bidirectional, unsupervised language representation, pretrained using only a plain text corpus.” This bidirectionality, combined with the self-attention mechanism, provides a better grasp of word meanings and context, which is reflected in achieving state-of-the-art performance on eleven natural language processing (NLP) tasks (Devlin et al., 2018). BERT’s outstanding performance and its open sourcing ensured that it was subsequently integrated into CA pipelines where it also achieves promising performance for a variety of tasks (Chao & Lane, 2019; Chen et al., 2019; Vakulenko et al., 2021; Voskarides et al., 2020).

As mentioned before, many CAs are applied in a specific context or domain and thus have to deal with domain specific data. For example, a conversational cooking assistant will mostly encounter cooking-related information needs like questions about the preparation or the quantity of ingredients (Frummet et al., 2021) but this is probably not the case for a customer-service chatbot for an e-commerce website. However, one of BERT’s limitations is the lack of domain specific knowledge, since pretraining was only performed on text data of the general domain, which can in turn lead to performance loss on the downstream tasks it is applied for (Gururangan et al., 2020; Lee et al., 2020).

Proceeding from this, the goal of this bachelor thesis is (overcome this limitation) the adaptation of BERT for one particular domain, i.e., the domain of cooking. (in order to provide a sophisticated model that can be utilized in conversational agents for this domain.) The cooking domain was chosen because it is considered a pertinent context for CAs. Firstly, cooking provides situations where traditional search is rather inconvenient, as users are multi-tasking, and their hands are occupied. Moreover, it has been argued in the past, that aiding in the kitchen (e. g. via recipe recommendations) could potentially lower the barriers to healthier cooking and thus improve the nutrition of people (Elsweiler et al., 2015; Elsweiler et al., 2017; Freyne & Berkovsky, 2010). There also seems to be an strong demand for CAs in the kitchen, e. g. Google Home Devices were used for more than 16 million recipes during 2018th Christmas season, passing one million on Christmas day alone (Huffman, 2019). By adapting BERT for the cooking domain, it is hoped to increase its natural language understanding for this domain and with it the performance on downstream tasks relevant for kitchen conversational agents, which is an important step towards building a truly conversational system. The concrete contributions of this thesis include:

* the introduction of CookBERT, a domain adapted BERT model for the cooking domain,
* the evaluation of CookBERT on three conversational agent relevant tasks, as well as a comparison to the similar FoodBERT (Pellegrini et al., 2021) and standard BERT model,
* additional evidence that further pretraining on domain specific data is a viable strategy to obtain domain specific language representation models in a fast and cheap way.

The remainder of this thesis is organized as follows. In section 2, ….

# Related Work

To set the context for this thesis and motivate the research question as well the methodological decisions taken, this chapter covers the background and related work from research contributions across diverse fields of computer and information science, ranging from conversational AI to word embeddings to the recently popular transformer neural networks. The chapter is arranged as follows:

* Section 2.1 enthält den Background zu BERT und seine core concepts, die zum Verständnis diese Arbeit beitragen
* Section

## Conversational Artificial Intelligence

This chapter gives a general overview of conversational artificial intelligence as well as the research on their integration in the context of cooking. More precisely, section 2.1.1 provides a brief introduction to the topic of conversational artificial intelligence. In section 2.1.2, the inner workings of such systems are presented, and section 2.1.3 covers the research done towards the usage of conversational artificial intelligence systems in the kitchen.

* History/ Anwendungsgebiete/ von rule-based zu statistischen Approaches
* Motivate the use of neural networks/BERT / show that these latest neural networks are state of the art in such CAs
* Funktionsweise
  + Zwei Architekturen werden vorgestellt, um einen oberflächlichen Überblick zu bekommen, wie Conversational agents die Conversationen handhaben und welche Aufgaben CAs innerlich bewältigen müssen, (which will also help to choose the tasks that CookBERT is tested on)
* CAs in the kitchen
  + Motivate the domain of choice
  + Why is kitchen believed to be a fertile context?

### Introduction

Conversational Artificial Intelligence, widely known by the acronym conversational AI, is “the study of techniques for creating software agents that can engage in natural conversational interactions with humans” (McTear, 2020, Preface). Compared to the typical interaction via command line or graphical user interfaces, such systems provide the opportunity for more natural interaction via conversation and thus promise an increased ease of use and speed of user requests as well as a convenient usage (Brandtzaeg & Følstad, 2017). Conversational AI systems are sometimes even referred to as “the new world of HCI” , with well-known personalities from the tech-industry like Satya Nadella (Microsoft CEO) or Mark Zuckerberg (Facebook CEO), who praise them as a solution to the current app-overload problem and even compare the upcoming developments of such systems to earlier, major revolutions like the introduction of the graphical user interface or the web (Følstad & Brandtzæg, 2017).

Generally speaking, the chatbot ELIZA (Weizenbaum, 1966) is considered the first implementation of a conversational AI system. It was designed to simulate a psychologist by using handcrafted rules for pattern matching and substitution to give the illusion of understanding of the conversation. More sophisticated systems appeared in the following years, but the general approach was still based on handcrafted rules, which is why these early conversational AI systems suffered from susceptibility for unexpected input, little scalability for domains other than the one they were created for, and the general lack of understanding, since these handcrafted rules simply could not cover the complexity of human language (McTear, 2020, pp. 23–24). It was only around the turn of the millennium that research shifted towards the development of statistical, data-driven systems that make use of machine learning (McTear, 2020, p. 19), and it is generally agreed that in 2011, when Apple release their personal assistant Siri, conversational AI systems became so mature, that they from now on became part of everyday life (McTear, 2020, p. 12). Especially new progress in deep learning in recent years, some of which will be discussed in section 2, ensured that conversational AI can nowadays be found in almost every context, ranging from personal digital assistants for phones such as Apple’s Siri to conversational agents for smart homes such as Amazon’s Alexa to customer service chatbots for e-commerce (Cui et al., 2017) and primary care (Ni et al., 2017). But the application context is not the only aspect that differentiates conversational AI systems. Many different implementations of such systems exist, each of which can be distinguished by multiple factors such as their knowledge, the service they provide, the primary goal that they try to achieve or by the way they process input and generate output (Nimavat & Champaneria, 2017). Additionally, it is to mention that the designation of conversational AI systems is very inconsistent in literature and media, and various terms like chatbot, conversational user interface, personal digital assistant, or voice assistant are often used interchangeably (McTear, 2020, pp. 12–13). In the further course of this thesis, *conversational agent* *(CA)* is used as a generic term for such conversational AI systems.

### Inner Workings

As mentioned in the previous section, there are many different implementations of CAs, all of which have to meet custom requirements. For this reason, there is no such thing as a one-size-fits-all architecture. However, to give a shallow overview over the inner workings and tasks that need to be solved, a general architecture, as proposed by Adamopoulou and Moussiades (2020), is presented in the following.



Figure 1: General architecture of conversational agents (Taken from (Adamopoulou & Moussiades, 2020, p. 380))

In the first step, a user request is received via the user interface, e.g., a physical device like Amazon’s Echo or a mobile app. In case of a voice user interface, the request is first converted to text and then fed forward to the natural language understanding (NLU) component.

The NLU component analyses the user request in order to gain some understanding of its meaning. Typical tasks that are fulfilled here are intent detection and entity extraction. Intent detection deals with determining the underlying goal that the user tries to achieve and is often formulated as a classification task, where the user’s utterances are assigned one or multiple labels. Entity extraction is the act of locating and classifying entities in a sequence of text. It often comes in the form of slot filling, where the information for predefined slots needs to be extracted. E.g., for the predefined slots *departure\_location* and *destination\_location*, and a request “book a flight from London to Paris”, the system would extract the two entities “Paris” and “London”, respectively. The intent detection component might classify this request as *book flight*. (McTear, 2020, pp. 46–47)

The dialogue management component is responsible for the state and flow of the conversation. It keeps track of the user’s intents and the identified entities as well as the information that is missing for filling specific slots. It is thus also responsible for requesting that missing information by asking follow-up questions, which are generated by the response generation component discussed below. If the dialog manager has all the required information, an action can be performed (e.g., booking a flight) or the appropriate information can be retrieved from some sort of data source. (Adamopoulou & Moussiades, 2020, p. 380)

The response generation component generates a response that is as close to human language as possible. There are generally three approaches to do this: rule-based, retrieval-based and the utilization of a generative model. As the name suggests, the rule-based approach makes use of a hand-written, predefined set of rules to generate a response and thus suffers from the limitations mentioned in the previous section. In case of the retrieval method, the user’s utterances can be thought of as a query and the goal is to retrieve an appropriate response from a corpus of conversations (Jurafsky & Martin, 2021, p. 530). Another possibility lies in the combination of information retrieval and machine reading comprehension as done by Yang et al. (2019) and Chen et al. (2017). Such approaches first retrieve relevant documents from an external knowledge source like Wikipedia and a so-called reader component then extracts candidate answer spans from those articles. The task of answer span extraction is commonly referred to question answering in literature. (Adamopoulou & Moussiades, 2020, pp. 378–379)

### Application in the Cooking Domain

In past research, the kitchen has been seen as a fertile context for CAs, as traditional search methods are unavailable or rather inconvenient due to users multitasking and having their hands occupied (Angara et al., 2017; Barko-Sherif et al., 2020; Frummet et al., 2021). In addition, it has been argued that assistance in the kitchen could potentially lower the barriers to healthier cooking and thus improve people’s nutrition (Elsweiler et al., 2015; Elsweiler et al., 2017; Freyne & Berkovsky, 2010). Lastly, the placement of general domain smart speakers like Amazon’s Echo in the kitchen is widespread and they are often used for cooking related requests, like setting a timer, getting recipe suggestions or requesting recipe instructions (Huffman, 2019; Kinsella & Mutchler, 2020, p. 7). For example, Google Home Devices were used for more than 16 million recipes during 2018th Christmas season, passing one million on Christmas day alone (Huffman, 2019).

Despite these arguments pro CAs for the kitchen, the research in that direction is rather sparse. Chu (2021) proposed RecipeBot, a conversational agent that recommends recipes based on user requests about aspects like the region, type or ingredients. It accepts voice-based or textual requests. These requests run through tasks like intent detection and named entity recognition to extract relevant information. This information is then sent to a database to receive an appropriate response.

A more sophisticated CA for the kitchen is *Foodie Fooderson* (Angara et al., 2017). Foodie Fooderson provides personalized recipe suggestions as it stores user preferences such as allergies or dietary goals in a personal context sphere. It uses IBM Watsons conversational services to design the structure of conversations between the system and the users as a workspace and consists of the three building blocks intents, entities, and dialog. Just as seen before, the intent block is for classifying the users intent and entity for information extraction, more precisely for keyword identification. The dialog block provides the structure for the flow of the conversation in the form of nodes and edges. Based on information from the intent and entity block, specific nodes are triggered which then defines the response of the system.

Neben solchen implementierten Systemen gibt es auch noch research

## Language Processing with Deep Neural Networks

This chapter covers the basics, that are relevant for understanding the processing of natural language with deep neural networks. Section 2.2.1 starts with the idea to represent text in a more machine-readable form, so that artificial neural networks (ANN) can process it more efficiently. What ANN are and how they work is addressed in section 2.2.2. The following sections present various network architectures that emerged for processing sequential data, ranging from recurrent neural networks and the more optimized long short-term memory neural networks in section 2.2.3 to the encoder-decoder architecture in section 2.2.5 to the recently popular transformer architecture with its attention mechanism in section 2.2.5 and 2.2.6, respectively. The technique of transferring gained knowledge from one task to another is highly relevant for current deep learning models and is therefore also covered in section 2.2.7.

### Word Embeddings



Figure 3: Two-dimensional projection of word embeddings. Note how similar words are nearby in space. (Taken from Jurafsky and Martin (2021, p. 107) as a simplified representation of Li et al. (2016))

In order for computers to be able to deal with text and process it efficiently, it needs to be presented in a different way. The representation should reflect the meaning of the text and the individual words as well as possible, and similar words should have a similar representation. The solution to capture the meaning of words that still exists to this day, stems from the so-called distributional hypothesis, formulated by several linguists in the 1950s (Firth, 1957; Harris, 1954; Joos, 1950). The assertion here is that the meaning of words is given by their context, i.e., words that occur in similar contexts tend to have similar meaning. The instantiation of this hypothesis is what is known as *word embeddings* – in simple words vectors of numbers that capture the meaning of words. An example of what embeddings can look like in 2-dimensional space is given in figure 1.

Context-free Embeddings

One of the simplest approaches is to create a so-called term-term matrix, where each row and each column represents a single word of the vocabulary and each cell of holds information about how often the row and column words appear together in close proximity in the text corpus. Each row then corresponds to the embedding for the word it is labelled with, which leads to the vector length being equal to the vocabulary size. As the number of words in the vocabulary is generally quite high and most of the words do not co-occur in the corpus, this results in long, sparse vectors with most entries being zero. (Jurafsky & Martin, 2021, pp. 110–111)

More sophisticated algorithms arose that narrow down the embedding dimension, including GloVe (Pennington et al., 2014) and Word2Vec (Mikolov, Chen et al., 2013; Mikolov, Sutskever et al., 2013). Although this results in dense vectors, which tend to capture the meaning of words quite well, such approaches have one major limitation: they are static, which means they have a fixed embedding for a word even though it can have different meanings, such as the word “tie” in the sequences “game ended in a tie” and “tie my hair back”.

Contextual Embeddings

The solution for this is contextual embeddings, which have made significant progress, especially with the introduction of *Embeddings from Language Models,* short *ELMo* (Peters et al., 2018). Instead of assigning a fixed embedding for each word, ELMo looks at the other words of the sequence in order to encode a word’s meaning into an embedding and can thus distinguish the word “tie” in the two sequences above. To do this, the authors concatenated two independently trained LSTMs (see section …) which were trained on language modelling (predict next word given previous words) and backwards language modelling (predict preceding word given posterior words), respectively. The contextualized embedding of a word is then created by extracting the hidden state for each layer and calculating the weighted sum of them.

In the following chapters, textual input into neural networks always refers to the corresponding embeddings rather than text in its “human readable” form.

### Artificial Neural Networks



Figure 4: Deep neural network with three hidden layers. Arrows indicate the direction of the information flow. (Taken from IBM Cloud Education (2020a))

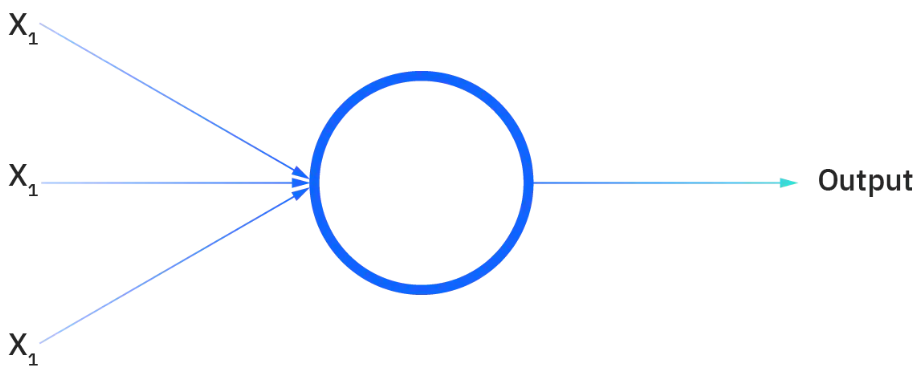


Figure 5: Structure of a single artificial neuron. (Taken from IBM Cloud Education (2020a)) 🡪 Austauschen, da auf dem Bild ein Perzepton ist

Artificial neural networks (ANNs) are mimics of the human brain, allowing computers to learn patterns from data. The typical ANN consists of three parts: one input layer, one output layer, and at least one hidden layer. ANNs with more than one hidden layer are referred to as deep neural networks (DNNs). Each layer consists of nodes, so-called neurons. In the simplest case, each neuron of a layer is connected with each neuron of the following layer without any backward jumps (see section 2.2.3), which is referred to as feed-forward neural network, as the output of the neurons from one layer is always fed forward and is the input for the neurons of the next layer. An example for a deep feed-forward neural network with two hidden layers is given in figure 2.



Figure 6: Commonly used activation functions (Taken from Roffo (2017))

Each neuron takes a single or multiple numeric values as inputs, each of which has a weight assigned to it that can be seen as the importance of the given input for the output that the neuron will compute. In order to calculate the output of a neuron, a bias is added to the sum of the weighted inputs and the result is then fed through a so-called activation function :

The activation function determines the degree of activation of the neuron. The choice of the activation function depends on the problem that needs to be solved. Generally speaking, non-linear activation functions are used, as linear activation functions would result in unrestricted outputs tending towards infinity, which leads to the fact that the neural network would be nothing more than a linear classifier and could no longer model complex, non-linear problems. Some commonly used activation functions are illustrated in figure. The bias is an individual numeric value for each neuron. It allows the activation function to be shifted and thus is crucial for successful learning of the neural network. The weights and biases in ANNs are typically randomly initialized or initialized to zero, respectively and then gradually adjusted when training the network.

Training Artificial Neural Networks

Training ANNs in simple terms is the process of finding the appropriate values for the weights and biases in order to map a set of inputs to a set of desired outputs. The most common algorithm for training neural networks is *error back-propagation* and was introduced for this purpose by Rumelhart et al. (1985).

When training an ANN, the training data is first passed forward through the network, starting from the input layer to the hidden layers and lastly to the neurons of the output layer which then produce the final output predictions of the network. Subsequently, these predictions are compared to the ground truth for the input and the error between them is calculated with a loss function. This error is then back propagated through the network and informs each neuron about their parameter distance to the ground truth value, thus allowing the adjustment of their weights and biases. As the goal of training a neural network is to adjust its parameters so that the output of the loss function is minimized, it can be considered an optimization problem. The direction of less error can thus be determined with optimization algorithms like *stochastic gradient descent*.

### Recurrent Neural Networks



Figure 7: (Taken from IBM Cloud Education (2020b))

Recurrent neural networks (RNNs) are specific types of ANNs. Compared to traditional ANNs they have connections between neurons of one layer to neurons of the same or previous layers, which gives them a kind of memory, as these cycles enable them to use information from previous inputs to influence the current input and output. This property is particularly useful when dealing sequential data, including speech, which can be viewed as a sequence of words, as RNNs no longer assume independence between individual inputs, and the output of the network now rather depends on the prior elements within the sequence. Figure 5 illustrates a RNN in its unrolled representations and shows that it becomes a feedforward NN made of as many replicas of the original layer as necessary in order to process all time steps of a given sequence. Each of these replicas has the same parameters which is another special feature of RNNs. The weights of those parameters are still adjusted via backpropagation and stochastic gradient descent, but unlike traditional ANNs, the backpropagation through time algorithm is used, which sums up the errors at each time step. For long sequences the gradients have to be passed back through many time steps, thus, RNNs tend to suffer from exploding (gradient is too large, creating an unstable model) or vanishing (gradient it too small and the network no longer learning) gradients.

Long Short-Term Memory



Figure 8: Structure of a single LSTM cell (Based on Oinkina (2015))

Long short-term memory (LSTM) is a special RNN architecture that tackles these gradient issues and was first introduced by Hochreiter and Schmidhuber (1997). As with traditional RNNs, LSTM can be represented as a chain of replicas, each of which is referred to as a cell in the following. The difference is that these LSTM cells not only contain one, but four network layers, which can be seen in figure 6. In addition to the normal hidden state , a LSTM cell also maintains a cell state at every time step, enabling it to remember certain information from the input sequence. The information flow is regulated by three gates, i.e., the input gate , the forget gate , and the output gate . They are all composed of a sigmoid neural network layer and a pointwise multiplication operation and enable information on the cell state to be added or removed. A single LSTM cell can then be described as:

where denotes the input vector at time step . can be seen as a supportive gate that computes how much to write to the cell state . , , , and , *, ,* are respectively the weight matrices and bias vectors to compute forget, input, output, and candidate gate vectors, and are learned during training. The sigmoid function is denoted as .

While the cycles in traditional RNNs let them maintain some sort of memory, they tend to struggle with long-term dependencies in the input sequence, which is not the case for the more sophisticated LSTM as their architecture was specifically designed to learn them.

### Encoder-Decoder



Figure 9: Encoder-decoder architecture (taken from Zhang et al. (2021))

* State durch context vector austauschen

Encoder-decoder (Sutskever et al., 2014) is a specific neural network architecture that was proposed to tackle sequence-to-sequence problems. The power of this architecture lies in its ability to map sequences of variable-length to each other, which was previously not possible with the existing neural network architectures. Since human language can be viewed as a sequence of words, the encoder-decoder architecture is very well suited for this and is used, for example, in text summarization (Nallapati et al., 2016), machine translation (Wu et al., 2016), speech recognition (Bahdanau et al., 2016) or video captioning (Venugopalan et al., 2015).



Figure 10: Machine translation illustrated as a sequence-to-sequence learning problem with a RNN encoder and a RNN decoder (taken from Zhang et al. (2021))

* Hidden states in die encoder states eintragen (also auf die Pfeile zwischen den Encoder blöcken) + context vector kennzeichnen

The architecture consists of two major components, illustrated in figure …. The first one is the encoder, which processes every item of the variable-length input sequence and captures it into a single, fixed dimensional representation vector, also known as context vector, which acts as the final hidden state of the encoder. This context vector is subsequently fed into the second component, the decoder, which then generates a variable-length output sequence. As the Encoder and decoder blocks are typically implemented with a recurrent neural network (RNN) architecture, especially LSTM (Hochreiter & Schmidhuber, 1997), the input processing of the encoder and the output generation of the decoder is done step by step in an auto-regressive manner, meaning that the they use information from previous steps to output the hidden state and predicted word, respectively (see Bild 2). While the default encoder-decoder architecture works fine for short input sequences, it struggles with longer ones, because it’s difficult for the encoder to compress all the contextual information of the long sequence into a single fixed size vector, which thus motivates optimization by means of “attention".

### Attention



Figure 11: Attention visualized in practical use with machine translation. X-axis and y-axis correspond to the words in the source sentence (English) and generated translation (French), respectively. Pixels indicate the focus of attention in grayscale (Taken from Bahdanau et al. (2014)).

Attention was introduced and refined by Bahdanau et al. (2014) and Luong et al. (2015), respectively. It is a technique that allows sequence-to-sequence models to better deal with long input sequences, as it enables the network to focus only on certain parts of the input sequence as needed. Thus, the model can keep track of all inputs that are believed to be crucial for determining the output. Figure 3 illustrates the impact of attention in practical use with machine translation. To correctly translate the English sequence “European Economic Area” into French, the order of the words needs to be reversed. By paying attention to the respective proper input words, the model is able to generate the desired output.

In order to integrate attention into an encoder-decoder model, two aspects need to be changed. On the one hand, the encoder not only passes its last hidden state (the context vector) to the decoder, but all of its hidden states that were output when processing the input sequence step by step. Note that each of these hidden states is specifically associated with a particular word of the input sequence, namely the word that was being processed at the time. The decoder, on the other hand, assigns a score to these handed over hidden states and multiplies it by its softmaxed score to boost the hidden states with high, and tone down the hidden states with low scores. The scoring is done for each step of the decoder. (Alammar, 2018b)

A more recent form of attention is self-attention. Self-attention has been proposed in several papers, in which it is also sometimes referred to as intra-attention (Cheng et al., 2016; Lin et al., 2017; Parikh et al., 2016; Paulus et al., 2017; Vaswani et al., 2017). It differs from the standard attention mechanism in that it applies attention within the same sequence, rather than across two different sequences. When processing each word of the input sequence, attention is paid to other input words that are assumed to be relevant for the “understanding” of the current word. Roughly speaking, self-attention is a mechanism to enrich the currently processed word with contextual information from its environment, which is particularly useful when facing disambiguation or for the resolution of coreferences and pronouns.

A deeper insight into the inner workings of self-attention and also multi-head self-attention in the context of the famous Transformer architecture is given in the next section.

### Transformers



Figure 12: Transformer architecture (Taken from Vaswani et al. (2017, p. 3))

The Transformer was proposed in the well-known “Attention is All You Need” paper by Vaswani et al. (2017). It is a special network architecture, namely the first to be solely based on the attention mechanism, not combining it with recurrence nor convolution. Besides the fact that the Transformer achieves superior performance in machine translation, it is above all the good parallelizability and the associated significant speed boost when training deep learning models that makes it stand out from previous approaches.

Architecture

Overall, the Transformer follows the encoder-decoder architecture, as it consists of an encoding and decoding component, shown on the left and right side in figure 4, respectively. The encoding component is a stack of six encoders, and the decoding component a stack of six decoders. Each of the encoders can in turn be broken down into a multi-head self-attention sub-layer (detailed explanation follows below) and a simple fully connected feed-forward sub-layer. Apart from an extra multi-head self-attention sub-layer, the decoders are built the same. The additional layer of every decoder each accept the output of the last encoder of the encoder stack and use it to help the decoder focus on appropriate places in the input sequence. Since the architecture does not rely on recurrence nor convolutions, it adds positional encodings that encode the necessary information about the position of the words and distance to other words in the input sequence to the encoder and decoder inputs. As can be seen in figure 4, there are also shortcut connections for each sublayer in the encoders and decoders to the next normalization layer, which allow forward and backward passes of information and are mechanisms to avoid the problem of vanishing and exploding gradients. Layer normalization, proposed by Ba et al. (2016), normalizes each feature of the activations to zero mean and unit variance. This is done to tackle the problem of covariate shift, which refers to the distribution shift of training and test data. (Alammar, 2018a; Rush, 2018; Vaswani et al., 2017)

****Self-Attention****

The self-attention mechanism of the Transformer architecture, which is more precisely referred to as *scaled dot-product attention* by Vaswani et al. (2017), consists of six steps and can be illustrated with the abstracted concept of *query*, *key,* and *value* vectors. In step one, these three vectors are initially created by multiplying the embedding vector by three weight matrices that were learned during the training of the network (Note that only the first encoder starts with the original embeddings, all other encoders start with the output of the preceding encoder). In step two, the scaled dot-product attention score is calculated for every word of the input sequence by taking the dot product between the query vector of the currently processed word and the respective key vectors of the other words. The next two steps include the division of the calculated score by a fixed number (typically the square root of the key vector dimension) and feeding it through a softmax function to get more stable gradients and normalize the score, respectively. The resulting softmax score that is now assigned to each word in the input can be seen as the weight that each input word has on grasping the actual meaning of the currently processed word. In the fifth step, each value vector is multiplied by its softmax score and then all value vectors are summed to produce the final output of the scaled dot-product attention layer, which is the contextualized embedding for the currently processed word. Note that in the actual implementation, matrices are used for calculation, since they enable faster processing. This means that all embeddings are packed into a single input matrix, with each row corresponding to a word of the input sequence. After calculating the key, value and query matrices, the output of the scaled dot-product attention layer can be calculated with a shortened equation:

where , and denote the query, key, and value matrix, respectively, and denotes the key dimension. The general flow of information is visualized again in figure 5. (Alammar, 2018a; Rush, 2018; Vaswani et al., 2017)

Multi-head self-attention



Figure 13: Comparison of scaled dot-product attention (left) and multi-head scaled dot-product attention (right). (Taken from Vaswani et al. (2017, p. 4))

Multi-head self-attention improves the scaled dot-product attention mechanism, as it runs the mechanism multiple times in parallel, each with different query/key/value weight matrices that were learned during training the network. In case of the Transformer which uses eight attention heads, this results in eight output matrices. To ensure that the upcoming feed-forward layer receives its desired input, namely a single matrix, the eight scaled dot-product attentions are concatenated and linearly transformed. Vaswani et al. (2017) claim that this multi-head approach allows to “jointly attend to information from different representation subspaces at different positions” (p. 4) and thus exceeds the performance of single-head attention. The general information flow of the multi-head approach compared to a single-head scaled dot-product attention layer is visualized again in figure 5. (Alammar, 2018a; Rush, 2018; Vaswani et al., 2017)

### Transfer Learning

Transfer learning is a concept that describes the process of transferring knowledge gained by a model during the training on a source task and domain to a different, but related target task or domain. Note that in most cases throughout this thesis, knowledge relates to the representations learned by neural network models. Transfer learning is particularly helpful in cases when specific tasks or domains suffer from the problem of data scarcity, and it can also eliminate the need for training a new model from scratch each time a new task or domain is to be solved. (Ruder, 2019, pp. 42–43)



Figure 14: Transfer learning taxonomy (Taken from Pan and Yang (2010) in the truncated version of Ruder (2019))

In the work of Pan and Yang (2010), different setting for such knowledge transfer are identified and categorized. Their proposed taxonomy is adopted in this thesis and is shown in a simplified version in figure 12. In the following, the two transfer learning scenarios relevant to this thesis are explained, namely *sequential transfer learning* and *domain adaption*.

Sequential transfer learning

The setting of sequential transfer learning occurs when the source task differs from the target task and the aim is to transfer the knowledge of a model that was trained on the source task to the target task in order to improve its performance. Note that the tasks in this setting are learned sequential, otherwise there would be talk of multi-task learning. Sequential transfer learning can be useful in several situations, e. g. if there is a lot of data available for the source task but only little for the target task, or if adaption to many different tasks is necessary. Generally speaking, this kind of transfer learning consists of two phases, i.e., pretraining and adaption. (Ruder, 2019, pp. 63–64)

During the pretraining step, the model is trained on the source task, which is typically selected in a way that representations can be learned that are helpful for a broad range of different tasks. A sufficiently large amount of data should also be available for the source task so that the model can learn solid representations. This makes pretraining comparatively expensive, but it generally only has to be done once. (Ruder, 2019, pp. 65–66)

The adaption phase includes the transfer of the knowledge that the model gained during pretraining to the target task. Compared to pretraining, this process is usually less expensive. In order to adapt the pretrained model to another task, either feature extraction or finetuning can be used. With feature extraction, the learned representations are fed into a separate model as additional features. With finetuning, on the other hand, the pretrained model with its learned representations and parameters is used as a starting point and then updated through training on the target task. Thus, finetuning trains the pretrained model directly on the data of the target task and no separate model is required. (Ruder, 2019, p. 77)

Domain Adaptation

Often the distribution of the data seen during training differs from the distribution of the data to which the model is applied. In this case, domain adaption is appropriate. Domain adaption aims to learn better representations for a specific target domain, unlike sequential transfer learning, which tends to aim to learn more general useful representations. Furthermore, domain adaption assumes both, the source, and the target domain, to draw their features from the same feature space, e.g., that both domains consist of text of the same language. (Ruder, 2019, p. 86)

ULMFiT

A major step towards efficient transfer learning in natural language processing was the publication of Universal Language Model Fine-Tuning (ULMFiT) by Howard and Ruder (2018). With ULMFiT they proposed the concept of pretrained language models. Their method is based on the sequential transfer learning approach. First, they train a general domain language model on a large text corpus, which is based on a variant of LSTM (see section …). The pretrained model can then be finetuned for any classification task, without the need for custom feature engineering or additional in-domain documents or labels. In their work, the researchers managed that their finetuned model, finetuned on only 100 examples, achieved the same performance as a model that was trained from scratch on 100 the number of examples.

## BERT

With the publication of *Bidirectional Encoder Representations from Transformers (BERT)* by the Google AI team (Devlin et al., 2018), a small revolution in the field of NLP was triggered. BERT is a huge neural network model that builds upon influential work on pretraining contextual representations, particularly Semi-supervised Sequence Learning (Dai & Le, 2015), GPT (Radford et al., 2018), ELMo (Peters et al., 2018) and ULMFiT (Howard & Ruder, 2018). BERT’s outstanding performance is reflected in the fact that it achieved state-of-the-art performance on eleven different NLP tasks and makes it the most popular NLP model in recent years. This first section of this chapter explains the inner workings of BERT in order to justify its outstanding performance compared to other models. Section 2.3.2 provides an overview of BERT’s application in conversational AI and section 2.3.3 covers different approaches in literature to overcome the limitation of lacking domain specific knowledge.

### Inner Workings



Figure 15: Overview of pretraining model architectures. BERT is deeply bidirectional, GPT is unidirectional, and ELMo is shallowly bidirectional (Taken from Devlin et al. (2018, Appendix A))

BERT’s architecture s is simply a stack transformer-encoders (see section …). BERT originally comes in two different versions, BERTBASE and BERTLARGE, which differ in the number of layers (i.e., transformer-encoder blocks) , their hidden size , their number of self-attention heads and their total number of parameters , with , , , for BERTBASE and , , , for BERTLARGE. What makes BERT so powerful in comparison to previous models is that it is “the first deeply bidirectional, unsupervised language representation, pre-trained using only a plain text corpus” (Devlin & Chang, 2018). That means it uses both the previous and following context of a word to generate a representation, while starting from the very bottom of a deep neural network, what distinguishes it from ELMo’s approach (see section 2.2.1). In order to learn these representations, BERT was pretrained on a huge amount of general text data (3.3B words), originating from BooksCorpus (Zhu et al., 2015) and Wikipedia articles, using two unsupervised tasks, namely masked language modelling (MLM) and next sentence prediction (NSP).

Masked Language Modelling

MLM is not a new invention and was first proposed a long time ago under the name of Cloze-Procedure (Taylor, 1953). With BERT’s MLM, 15% of the tokens of an input sequence are randomly selected. 80% of these selected tokens are masked, i.e., replaced with a [MASK] token, 10% are replaced by a random token of the vocabulary and 10% stay unchanged. The model’s job is then to predict the masked tokens of the sequence based on their previous and next contexts. MLM allows the model to learn representations in both directions simultaneously and without the risk of knowing before-hand what token is coming next. (Devlin et al., 2018, Pre-training BERT)

Although MLM might not be that useful of a task on its own, it is great for learning representations that are later on helpful for a broad range of different tasks, and there is also sufficient training data available, so it ticks the boxes for successful pretraining as discussed in section Transfer Learning.

Next Sentence Prediction

NSP is about predicting whether two sentences are adjacent and is used by BERT in order to understand the relationship between sentences, which is not directly covered by MLM. During training, the model is presented with two truly adjacent sentences 50% of the time and for the other 50%, the second sentence is picked randomly from the pretraining corpus. (Devlin et al., 2018, Pre-training BERT)

Input and Output Representations

BERT does not input the embeddings of whole words but rather uses WordPiece (Wu et al., 2016) embeddings with a 30,000 token vocabulary. This means the input sequence is segmented into single tokens based on this existing vocabulary. If a word as a whole is not included in the vocabulary, it is split into multiple subtokens, e.g., . Note that all subwords except for the first one start with . Additionally, BERT adds a *[CLS]* token at the beginning of each sequence and a *[SEP]* token at the end of each sentence, respectively. The [CLS] is added, so that the final hidden state that corresponds to it can be used as the aggregate sequence representation for classification tasks. [SEP] is added to differentiate sentences within the input sequence which is necessary for specific tasks such as NSP or question answering.

The final representation of a token results from the sum of its token, segment (encodes information about what sentence the token is in) and positional embedding (encodes information about the position of the token in the input sequence).

### BERT in Conversational AI

BERT’s outstanding performance followed by its open sourcing (Devlin & Chang, 2018) ensured that is was also applied in conversational AI research.

* Aufgrund des open sourcings von BERT konnten viele Researchers die Ergebnisse reproduzieren und BERT für neue tasks ntuzen, wobei BERT bestleistungen in …, …, … oder … erzielt hat.
* Es hat nicht lange gedauert, dann wurde BERT auch für Conversational AI genutzt

### Domain Adaption



Figure 16: Common training schemes (Taken from Konle and Jannidis (2020))

* Auswirkungen der Datenmenge auch kurz anschneiden am Ende
* Camembert: NSP bringt nichts, nur MLM ist ausreichend

## Cooking datasets

Despite the fact that cooking has recently received some attention for NLP research, the number of sophisticated datasets in this domain is rather small. This shrinks even more when only considering datasets that are somehow relevant for conversational AI and thus could be used to train and test CookBERT. Available datasets that meet this criterion and are therefore utilized in this thesis are presented in the next sections.

### RecipeNLG

*RecipeNLG* (Bień et al., 2020) is a cooking recipe dataset for semi-structured text generation. It contains over 2.2 million distinct recipes and is assumed to be the largest publicly available dataset for the cooking domain. RecipeNLG builds upon the preceding Recipe1M+ dataset (Marin et al., 2019) and extends it with over one million cleaned, deduplicated recipes scraped from multiple cooking websites. Each entry of the dataset contains the following information: the title of the recipe, a list of ingredients and quantities, a list of instructions, the link to the recipe, information about its source (gathered or originating from Recipe1M+ dataset) and a list of automatically extracted food entities. Bień et al. (2020) also trained two GPT-2 language models on their and the Recipe1M+ dataset, respectively, in order to compare their ability to generate recipes only based on food entities. They found that the model trained on RecipeNLG both made fewer linguistic errors and performed better for all translation metrics than the model trained on Recipe1M+, emphasizing the higher quality of their dataset.

### Cookversational Search

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Utterance** | **Level 0** | **Level 1** | **Level 2** | **Level 3** | **Level 4** | **Level 5** |
| “Um can you find me dishes with asparagus with many dairy products.” | Fact | Recipe | Recipe Retrieval | Recipe Request | Recipe Request with Ingredients | Explicit |
| “Um – How do you prepare bulgur?” | Competence | Cooking technique | Cooking technique – Ingredient | – | – | – |

Table 1: Excerpt from cookversational search dataset. (Based on Frummet et al. (2021, p. 11))

*Cookversational search* is the resulting dataset of the work Frummet et al. (2021), in which the information needs that arise during cooking were examined (see section …). The human-labelled dataset is intended for the task of text-classification. It consists of 2675 user utterances, available in German (original language) and English (automatic translation), for which the underlying information need is to be classified. Labels are provided for six different levels of information needs, with some levels sometimes not having a label assigned. Embedded history information for single utterances, as used in the paper, is not included directly in the dataset, but the information to do this manually is. An excerpt of the dataset is given in table 1. 🡪 In anhang packen

### DoQA

|  |  |  |
| --- | --- | --- |
| **Context** | **Question** | **Answer** |
| “I think grilling is probably a bad plan for duck legs; the fat content is a real danger like you said, and duck legs are tough enough you probably want to confit them or braise them. If you absolutely have...” | “Tips for grilling duck legs?” | “I think grilling is probably a bad plan for duck legs” |
| “You can let it ripe at room temperature. If you want to slow down the ripening process, put it in the fridge, although this will affect the mango negatively…” | “What will be the negative effects of the refrigerator on the mango?” | CANNOTANSWER |

Table 2: Excerpt from DoQA dataset. (cite)

*DoQA* (Campos et al., 2020) is a dataset for accessing domain specific *Frequently Asked Question* *websites*, commonly known as FAQs, via conversational QA. It contains a total of 10917 QA pairs from 2437 dialogues from the three domains cooking, travelling and movies. With 7320 QA pairs, the largest proportion is given for the cooking domain, which is advantageous since this is also the domain of choice in this thesis. The dialogues were created via Wizard of Oz method with crowdsourcing, where the crowdworkers, which were divided into users and experts, had to ask questions about given FAQ posts or extract the answer span that is given in the original post, respectively. Since the underlying data for DoQA originates from real users with real information needs, the authors claim that “[C]ompared to previous work, DoQA comprises well-defined information needs, leading to more coherent and natural conversations with less factoid questions” (Campos et al., 2020, p. 1). Furthermore, the dataset contains answerable and non-answerable question. The underlying task to be solved with the dataset is, given a context and a question, to extract the passage from the context that contains the answer. An excerpt from DoQA that illustrates this task is given in table 1. 🡪 In anhnag packen

### FoodBase

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Spread | spinach | dip | over | the | pizza | crust |
| **Food-classification** | O | B-FOOD | I-FOOD | O | O | B-FOOD | I-FOOD |
| **Hansard-parent** | O | B-AG.01.h | I-AG.01.h | O | O | B-AG.01.n | I-AG.01.n |
| **Hansard-closest** | O | B-AG.01.h.02.c | I-AG.01.h.02.c | O | O | B-AG.01.n.11 | I-AG.01.n.11 |
| **FoodOn** | O | B-NCBITaxon\_3562 | I-NCBITaxon\_3562 | O | O | O | O |
| **SNOMED CT** | O | B-256329006 | I-256329006 | O | O | B-227757007 | I-227757007 |

Table 3: Excerpt from the FoodBase corpus, annotated for 5 different tasks by Stojanov et al. (2021)

*FoodBase* (Popovski, Seljak, & Eftimov, 2019) is a corpus for annotated food entities, available in a curated and uncurated version. In case of both versions, food entities were automatically annotated from cooking recipes with FoodIE (Popovski, Kochev et al., 2019), a rule based named entity tagger. Unlike the uncurated version, the curated version was then manually reviewed by experts to remove false positives and add false negatives, respectively. This leads to a total of 1000 curated and 21790 uncurated recipes. Each recipe belongs to one of five categories, with the distribution being stratified in the case of the curated corpus. The semantic tags used correspond to those of the hierarchical *Hansard* *corpus[[3]](#footnote-3).* It should also be noted that individual food entities have been assigned multiple appropriate semantic tags.

As an extension of the FoodBase corpus, *FoodOntoMap* (Popovski, Koroušić Seljak, & Eftimov, 2019) was published. This resource provides data normalization of FoodBase’s food entities according to different ontologies. More specifically, it provides a mapping between the semantic tags of Hansard, *FoodOn* (Dooley et al., 2018), *OntoFood* and *SNOMED CT* (Donnelly, 2006) ontology.

Stojanov et al. (2021) use both of these resources by combining and modifying them for their experiments. Their adapted dataset consists of the 1000 recipes from FoodBase, as well as five different semantic tagging tasks for each entity, which were partly taken from FoodOntoMap and partly constructed themselves: The task of **food-classification** is about distinguishing between food and non-food entities, whereby every food entity of the FoodBase corpus was simply labelled with a *FOOD* tag. For the **Hansard parent** task, the Hansard corpus labels from FoodBase were condensed into 48 superordinate semantic tags from the same ontology. When there were originally multiple labels for a single entity, the Hansard parent tag was chosen based on the first one listed. **Hansard closest** includes 92 different tags from the Hansard corpus.Here, for each FoodBase entity, the closest Hansard tag to the original tag in terms of cosine similarity between their BERT embeddings was chosen. The **FoodOn** task is about tagging the recipes with 205 tags from the FoodOn ontology. The corresponding FoodOn label was determined with the FoodOntoMap resource. The last task, **SNOMED CT**, is about distinguishing 207 tags from the eponymous ontology. FoodOntoMap was also used here to select the appropriate tag. Furthermore, the authors converted the tags for all five tasks to the commonly used IOB (inside, outside, and beginning) tagging format (Ramshaw & Marcus, 1999).[[4]](#footnote-4) Table 3 shows the respective annotations for each of the five tasks for a sample sentence. (Stojanov et al., 2021)

## Summary and Key Differentiators

# Methodology

## Preparing the Data for Domain Adaptive Pre-Training

In order to perform domain-adaptive pretraining, the RecipeNLG dataset (see section …) was used, more precisely the recipe instructions from it.

* Das und nächste Sektion unter einem Punkt “Prerequisites for DAPT” zusammenfassen
* Für das DAPT wird der RecipeNLG Datensatz (siehe Section in related work verwendet)
* Da für das Pretraining natürliche Sprachdaten/ Textdaten und nicht nur einzelne kontextfreie Wörter benötigt werden, werden vor allem die Rezeptinstruktionen verwendet.
* Um DAPT zu ermöglichen wurden die Daten, die aktuell noch in einzelne Instruktionen aufgeteilt waren zusammengefügt.
* jedes Rezept wird als eigenes, unabhängiges Dokument betrachtet und daher alle Rezeptinstruktionen zusammengefügt und jeweils in eine Zeile einer Textdatei geschrieben (was später furs Pretraining sinnvoll ist, da Zeile für Zeile angeguckt wird)

In order to adapt BERT for the cooking domain, the RecipeNLG dataset (Bień et al., 2020) was utilized.

* Mit über 2.2 Millionen unqie recipes is it assumed to be the largest publicly available dataset in the domain, und damit knapp 2\* so groß wie der vorgänger, Recipe1M+
* RecipeNLG is an expension of Recipe1M+, which Pellegrini et al. (2021) utilized to create their FoodBERT.
* RecipeNLG enthält gesäuberte, deduplizierte Rezepte
* The dataset contains
* Title: Rezepttitel
* Ingredients: Zutaten mit mengenangaben
* Directions: List of Instruktionen 🡪 das habe ich verwendet
* Link: link zum Rezept
* Source: Gathered (74%) oder von Recipes1M (26%)
* NER: named food entities; extracted mit einem NER
* However, only the instructions were of interest for the unsupervised pretraining.
* The im imperative formulierten instructions liegen als liste von einzelnen Anweisungen vor, ein Beispiel ist in Abb. Gegeben.
* The quality and influence of the instruction characteristics are discussed in section 4.4 (limitation section)
* Overall statistics of the instruction data can be found in table … (Anzahl Rezepte, durchschnittliche Instructions, Anzahl Wörter gesamt, Durchschnittliche Anzahl pro Rezept)

Mit 2.2 mio unique recipes ist es mehr als doppelt so groß wie recipe1M+

* Wie in section … schon erklärt, sind führen generell mehr Daten bei DAPT auch zu besserer Performance (wie eigentlich immer in machine learning). Deshalb wird ein möglichst großer Korpus für DAPT ausgewählt.
* FoodBERT verwendet den Recipe1M Datensatz bzw. Die Instruktionen davon
* In meinem Ansatz wird ein noch größerer Korpus verwendet
* Datenmengen bei DAPT in der Literatur
* FinBERT: TRC2-financial, 46.143 documents, 29 million words, 400k sentences
* HateBERT: RAL-E, 1.478.348 messages, 43.379.350 million tokens,
* BioBERT: PubMed Abstracts = 4.5B words + PMC Full-text articles 13.5B words
* CSBERT: 40.505.050 dialogues and 317.093.459 turns coming from all available customer service intents
* TOD BERT: 100.707 Dialoge, 1.388.152 Utterances von verschiedenen task-oriented dialogue Datensätzen
* MenuNER: YELP Dataset (only reviews from the restaurant category), 15.000.000 sentences
* Pretraining BERT on domain resources for short answer grading: Textbooks und QAs mit 1.1m, 0.6m und 1.3m Wörtern
* Außerdem gibt es Paper, die sagen, dass mehr pretraining Data (When do you need billions of words of pretraining data) bzw. Mehr DAPT data (sinnvoll ist)

## Analyzing Domain Similarity



Before the actual DAPT, the similarity of the target domain (cooking) and BERT’s pretraining domain was analyzed. The approach for the analysis is adopted from Gururangan et al. (2020) and quantifies the domain similarity based on the vocabulary overlap of the pretraining corpora. Therefore, RecipeNLG, Recipe1M+ and the WikiBook from CookBERT, FoodBERT and the standard BERT respectively were used for corpus from BERT pretraining were used for the analysis. As BERT’s original pretraining data is not publicly distributed, a Wikipedia dump (Merity et al., 2016)(515MB) and randomly sampled books from the “Homemade BookCorpus” (Kobayashi, 2018)(444MB) were used to reconstruct a similar corpus. From RecipeNLG and Recipe1M+, the recipe instructions were used as corpus data (1GB and 619MB respectively). For each corpus, the vocabulary, consisting of unigrams (after lowercasing and removal of stopwords and punctuation) was then created for each of the three corpora.

The vocabulary overlap between the corpora was then determined based on the 10000 most frequent unigrams of each domain and is illustrated in Fig 4. It shows a strong overlap between Recipe1M+ and RecipeNLG, which is not surprising given the fact that both corpora are from the cooking domain and Recipe1M+ is a subset of RecipeNLG. In contrast, the overlap between the WikiBooks corpus and the two cooking corpora is quite small, emphasizing the data shift between the cooking domain and the general text domain. Furthermore, this simple analysis indicates the degree of benefit to be expected by adapting BERT for the cooking domain, as the potential for DAPT is higher, the more dissimilar the domains (Gururangan et al., 2020, p. 3).

* Unbedingt noch „A survey on Transfer Learning“ angucken!!

## Domain Vocabulary Insertion

|  |  |
| --- | --- |
| **Word** | **Tokenized representation** |
| baguette | bag ##uet ##te |
| cranberry | cr ##an ##berry |
| caramelized | cara ##mel ##ized |
| zucchini | zu ##cchi ##ni |
| preheat | pre ##hea ##t |
| tortilla | tor ##till ##a |
| eggplant | egg ##pl ##ant |

The influence of out-of-vocabulary words was proven to have negative influence on NLP models (<https://aclanthology.org/P11-2071.pdf,> <https://arxiv.org/pdf/1802.02614.pdf>). Even though BERT deals quite well with OOV words by splitting them up into smaller subtokens (see section …), kann die insertion of domain specific vocabulary as an adaption strategy auch hier zu einer besseren Performance führen (SciBERT und ExBERT).

Auch für die Kochdomäne scheint dieser Schritt sinnvoll zu sein, da viele gängige kochspezifische Vokbalen nicht im BERT Vokabular enthalten sind und dementsprechen in “unrepresentative” Subtokens zerlegt werden, as shown in figure

Um das Vokabular von CookBERT zu erweitern, wurden alle Wörter aus dem in Sektion 3.2 erstellten RecipeNLG Vokabular, die mindestens 1000 mal vorkamen und noch nicht Teil des BERT Vokabulars sind, hinzugefügt. Da die weights für neu eingefügten Wörter neu initialisiert werden, wurde darauf geachtet, dass sie einigermaßen häufig im Corpus vorkommen, um gute Repräsentationen lernen zu können.

Insgesamt wurden 1229 kochspezifische Wörter zum Bereits existierenden Vokabular hinzugefügt, was zu einer neuen Gesamtgröße von 30000 Vokabeln führt.

* Alle Wörter mit einer Häufigkeit von mindestens 1000, und die noch nicht im BERT vocab enthalten waren 🡪 Das hat den grund, dass die Wörter häufig genug vorkommen, um gute Repräsentationen zu erlernen, nachdem die Gewichtungen für diese quasi von 0 initialisiert werden müssen. Kommen sie zu selten vor, dann kann dies evtl. nicht richtig erlernt werden
* Wie in der vorherigen Section schon erwähnt wurden jedes Rezept erst tokenisiert, lowercased, und stop-words und punctuation wurde entfernt.
* Insgesamt wurden 1229 Wörter zum bestehenden Vokabular hinzugefügt
* Irgendwo noch mit reinbringen, wie FoodBERT das gemacht hat: die hatten eien Liste mit Zutaten und haben dann alle, die öfter als 10 mal vorkamen, zum Vokabular hinzugefügt. 🡪 Allerdings: es gibt auch viele Wörter, die nicht zutaten sind, die im Kochjargon vorkommen (skillet, …), diese werden aber vernachlässigt
* Fügen alle Zutaten hinzu, wenn diese mind. 10 mal vorkommen. Allerdings nehme ich an, dass dies nicht ausreichend ist, da die repräsentationen komplett neu initialisiert wurden und erst trainiert werden müssen, wobei 10 beipsiele nicht ausreichen
* With the insertion of our custom legal vocabulary, the tokenization provides different segmentation of words affecting the input sequence for BERT and we expect to slightly improve the performance in the same way SciBERT did by creating their own SciVocab (subsection 3.2.2).
* Effects of inserting domain vocabulary (S. 52)

## DAPT

* BERT kommt in mehreren Ausführungen (BERT large, BERT base, BERT base cased, BERT base uncased, …)
* BERTBASE: (L=12, H=768, A=12, Total parameters = 110M)
* BERTLARGE (L=24, H = 1024, A=16, Total Parameters = 340M)
* L: number of layers, H = hidden size, A = number of atterntion heads
* Als ausgangspunkt wurde der BERT base uncased checkpoint verwendetDAPT: **BERTBASE\_UNCASED, da**
* Cased model würde zwischen Bread und bread unterscheiden, obwohl beide dasselbe Konzept sind. Während in anderen Sprachen die groß und kleinschreibung eine wichtigere Rolle spielt (z. b. German) ist das im englischen moistens nicht wirklich der Fall, weswegen uncased verwendet wurde
* BERTbase, da obwohl BERTlarge yields better results, aufgrund der erhöhten Komplexität ein deutlich höherer Ressourcenaufwand
* If your task has a large domain-specific corpus available (e.g., "movie reviews" or "scientific papers"), it will likely be beneficial to run additional steps of pre-training on your corpus, starting from the BERT checkpoint. (https://github.com/google-research/bert#pre-training-tips-and-caveats)
* The learning rate we used in the paper was 1e-4. However, if you are doing additional steps of pre-training starting from an existing BERT checkpoint, you should use a smaller learning rate (e.g., 2e-5).
* Um DAPT durchzuführen, wurde das Model weiter auf den MLM tasks trainiert. NSP wurde nicht verwendet, da nicht hilfreich (siehe RoBERTa und CamemBERT)
* Result of DAPT 🡪 Beispielsätze zur Demonstration einfügen

## Implementation Details

* Evtl als unterpunkt zu 3.4 packen oder ans ende der Methodology als eigenen Puntk, in dem dann die Implementation details sowohl zum DAPT als auch furs Finetuning stehen.
* Learning rate, epochen, Model startpunkt, …
* Dauer des Learning vorgangs
* Verwendete Library: Huggingface
* Verwendete Umgebung: Google Colab
* GPU: P100 GPU von Google Colab+

## Finetuning

### Intent Classification



Figure 17: Frequencies of level 1 information needs (Taken from Frummet et al. (2021))

* Multi-class classification problem
* Siehe paper von Frummet für Vorgehen (an dem orientiere ich mich eben)
* Datensatz von Frummet
* Alles so wie Frummet gemacht
* 85% train, 15% test
* No resampling
* No stopword removal
* Stratified sampling for 10 fold cross validation
* To avoid catastrophic forgetting: lower learning rate of 2e-5
* Training for 4 epochs, dropout probability of 10%, batch\_size 32
* Early stopping was included
* Wegen computing limitations wurde eine maximale Sequ. Length von 256 verwendet. D.h. wenn mehrere Turns mit angehänt wurden, wurden nur die letzten 256 tokens verwendet.
* Auswertung mit drei contexten:
  + 1. No context
    2. 1 prev turn
    3. All prev turns
* Anders als Frummet gemacht:
* Frummet hat 11 binary classifiers mit jeweils einem classificationHead der Dimension 768,2. Ich habe nur einen classifier mit classification Head mit dimension 768,11.
* Class weights were adjusted by FARMs datasilo 🡪 evlt auch machen, siehe <https://discuss.huggingface.co/t/class-weights-for-bertforsequenceclassification/1674/7>

### Named Entity recognition

### Question Answering

# Evaluation

## Multi-class Classification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **Condition** | **Precision** | **Recall** | **F-Measure** | **95%-CI** |
| BERT base | no context | 47.94% | 48.68% | 46.15% | [41.15%;51.16%] |
| 1 prev turn | 46.29% | 49.84% | 45.38% | [40.06%;50.70%] |
| CookBERT | no context | 48.58% | 55.65% | 50.72% | [45.54%;55.90%] |
| 1 prev turn | 52.26% | 59.30% | 54.05% | [48.93%;59.16%] |
| FoodBERT | no context | 42.41% | 49.81% | 44.32% | [38.92%;49.73%] |
| 1 prev turn | 36.89% | 44.49% | 38.09% | [32.64%;43.55%] |

Table 4: Multi-class classification experiment results after 10-fold cross validation grouped by model and condition.

The results for the precision, recall and F-measure for the multi-class classification are listed in table 4. Accuracy was not considered as it is not a reliable metric in this case due to the high imbalance of the dataset. It shows that CookBERT performs best for both conditions, followed by BERT base which outperforms FoodBERT in both conditions.

To check whether the performance of any model is significantly different from others with respect to the two conditions, a one-way ANOVA, followed by a pairwise post-hoc t-test with Bonferroni-adjusted values was conducted. When comparing the model’s performance for the *no context* condition, no significant differences were found (F = 1.5789635379047495, p = 0.2077566090518626). In case of the *1 prev turn* condition, CookBERT performs significantly better than FoodBERT (p = 0.000103). The results between CookBERT and BERTbase (p = 0.062466) and BERTbase and FoodBERT (p = 0.177768) are not significant.

To assess the overall performance of the models, the same statistical procedure as mentioned above was applied. This indicates that CookBERT performs significantly better than BERT base (p = 0.033649) and FoodBERT (p = 0.000105). No significant difference was found between FoodBERT and BERTbase (p = 0.267011). The overall performance over both conditions is 52.38% for CookBERT, 45.77% for BERT base and 41.21% for FoodBERT.

In addition, a one-way ANOVA was conducted to check if the best model’s performance, namely CookBERT’s, is significantly different between both conditions. Although CookBERT performed better for the *1 prev turn* condition (M = 54.05%) than for the *no context* one (M = 50.72%), these findings are not significant (F = 0.8188998154262772, p = 0.3665010230982727).

## Named Entity Recognition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **Task** | **Precision** | **Recall** | **F-Measure** | **95%-CI** |
| BERT base | Food-classification | 90.68% | 96.06% | 93.29% | [92,87%;93.71%] |
| FoodOn | 65.24% | 73.10% | 68.94% | [67.04%;70.83%] |
| Hansard-parent | 80.35% | 88.68% | 84.31% | [83.54%;85.08%] |
| Hansard-closest | 70.79% | 79.98% | 75.10% | [73.87%;76.34%] |
| SNOMED CT | 63.04% | 70.65% | 66.62% | [64.49%;68.75%] |
| CookBERT | Food-classification | 92.25% | 96.52% | 94.47% | [94.17%;94.76%] |
| FoodOn | 69.75% | 77.51% | 73.42% | [71.91%;74.93%] |
| Hansard-parent | 82.72% | 89.18% | 85.83% | [84.69%;86.97%] |
| Hansard-closest | 72.21% | 80.41% | 76.08% | [74.60%;77.56%] |
| SNOMED CT | 68.58% | 75.51% | 71.87% | [69.99%;73.75%] |
| FoodBERT | Food-classification | 85.28% | 94.24% | 89.53% | [88.90%;90.17%] |
| FoodOn | 58.73% | 61.03% | 59.85% | [56.56%;63.13%] |
| Hansard-parent | 68.41% | 80.62% | 74.01% | [72.13%;75.90%] |
| Hansard-closest | 59.55% | 67.52% | 63.28% | [60.43%;66.13%] |
| SNOMED CT | 53.63% | 51.84% | 52.67% | [49.17%;56.17%] |

Table 5: Named entity recognition experiment results after 10-fold cross validation grouped by model and task.

The results for the precision, recall and F-measure for the named entity recognition task are listed in table 5. Just as with the classification evaluation in the previous section, the accuracy was not taken into account here either, as it is not a reliable measure, since most of the tags are assigned with the outside tag and the accuracy is therefore very high for all systems and tasks. As can be seen in the table, the order of the best-performing models is the same across all tasks: CookBERT achieves the best performance on all tasks, followed by BERT base, and FoodBERT consistently performs the worst.

In order to investigate if these performance differences on each task are significant between the three models, a one-way ANOVA as well as a pairwise post-hoc t-test with Bonferroni-adjusted values was conducted.

Comparing CookBERT to FoodBERT, CookBERT performs significantly better on four tasks, including *Food-classification* (p = 1.372926e-11), *FoodOn* (p = 3.122529e-07), *Hansard-parent* (p = 1.259690e-09) and *SNOMED CT* (p = 6.706296e-09). In comparison to BERT base, CookBERT performs significantly better on the three tasks Food-classification (p = 1.959512e-04), FoodOn (p = 0.001684) und SNOMED CT (p = 0.001708), but this is not the case for the Hansard-parent task (p = 6.731523e-02). Similar to CookBERT, BERT base performas significantly better than FoodBERT on the four tasks *Food-classification (*p = 4.758076e-09)FoodOn (p = 0.000113), Hansard-parent (p = 3.269430e-09) und SNOMED CT (p = 1.258617e-06). *Hansard-closest* is the only task where no significant performance differences between any of the three models could be found (F = 1.3139640931845475, p = 0.26668879504706366).

The overall performance for named entity recognition on the FoodBase corpus, given as the macro-averaged F-measure over all tasks, is 80.33% for CookBERT, 77.65% for BERT base and 67.87% for FoodBERT.

## Question Answering

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Exact match** | **F-Measure** | **95%-CI** |
| BERT base | 14.06% | 32.39% | [31.25%;33.54%] |
| CookBERT | 12.51% | 30.64% | [29.50%;31.78%] |
| FoodBERT | 10.81% | 27.51% | [26.51%;28.50%] |

Table 6: Question answering experiment results after 10-fold cross validation.

The results for the exact match and F-measure for the question answering task are listed in table 6. Between the three models, BERT base achieves the highest and FoodBERT the lowest scores for both metrics. A one-way ANOVA followed by a pairwise post-hoc t-test reveals that there is no significant difference in CookBERT’s and BERTbase’s performance (p = 0.072057). However, both CookBERT and BERTbase perform significantly better than FoodBERT (p = 0.000558 and p = 0.000003, respectively)

# Discussion

* Können Hypothesen verworfen werden?

Tabellenrahmen

* Normal: 1.5
* Dick: 2.25
* Schriftgröße: 9

# Limitations

This section reflects on the limitations of this thesis. First of all, the data used for domain adaptive pretraining is not considered optimal, as it only consists of recipe instructions. This is good in the sense that it contains a lot of cooking-specific vocabulary to adjust BERT’s weights to the cooking domain accordingly, and that there are also sufficient training samples for the newly added vocabulary. However, they are not very natural, since the instructions are mostly formulated as imperative, the syntax is often incorrect due to omitted articles and pronouns, and words are often replaced by abbreviations. A good example is the sentence “add egg and 3 tbls butter to batter” which should actually read “add the egg and 3 tablespoons of butter to the batter” correctly. This unnaturalness could have a negative impact on CookBERT’s linguistic competence as it might be adopted when performing DAPT. In section Discussion this is also listed as a possible reason why CookBERT performs worse in question answering than BERT base. This limitation is difficult to remedy because only few datasets exist that provide sufficient data for DAPT. So, the best approach would be to collect a huge amount of natural, cooking specific data yourself, for example from subtitles of cooking shows, from cooking podcast transcripts, or from general cookbooks (that do not only contain recipe instructions).

Another limitation of this thesis is the relatively small number of tasks that CookBERT’s performance was evaluated on, which is due to the scarcity of suitable cooking datasets. Despite promising results, it is thus difficult to make general statements about CookBERT’s performance. In addition, some of the datasets used for evaluation, do not perfectly match the desired requirements. FoodBase, like RecipeNLG, only contains text in the form of recipe instructions, which thus is unnatural and not the kind data a CA for the kitchen would encounter.

* Keine optimalen Hyperparameter beim Finetuning und kein early stopping applied. Optimale Hyperparameter could significantly improve a model’s performance and could therefore have an impact on the results obtained in section 4. Early stopping is a technique for overfitting.
* Es wurde sich nur der Aspekt der Kochdomäne angesehen. Allerdings könnte es auch sehr interessant sein, BERT für DIaloge anzupassen (siehe existing dialogue BERT)
* Es wurde sich nur kochspezifische Performance angesehen. Allerdings wäre auch eine untersuchung der allgemeinen Performance bei Konversationen interessant (hat die Adaption jetzt zu Vergessen geführt)

The last limitation to mention is the fact that only one model was adapted for the cooking domain. Dadurch konnte nur geguckt werden, wie sich das DAPT auf das Base model ausgewirkt hat. Bei anderen Models gäbe es aber vielleicht andere Ergebnisse. Zudem wurde BERT nicht mit weiteren Modellen verglichen.

* Es ist nicht klar, woher die bessere/ schlechtere Performance kommt.
* Begrenzte Ressourcen für Pretraining/ Finetuning
  + Aus resourcengründen konnten manche Modelle nicht ausreichend Finegetuned werden. Das eine Paper zeigt, dass für FoodBase korpus an die Hundert Epochen trainiert werden kann ohne zu überfitten. Aufgrund der zeitlichen Begrenzung der BA, sowie der Tatsache, dass 10-fold cross validation zu Evaluierung verwendet wurde (und somit jeweils 10 mal finegetuned werden muss 🡪 10 mal pro model pro condition für insgesamt 5 conditions und 3 models 🡪 150 verschiedene Modelle) 🡪
* Nicht optimales vorgehen bei Finetuning: kein Hyperparamter search
  + Es wurde sich nur an der literatur orientiert

# Conclusion

* Vorschlag: andere Datenquelle zum Pretrainnen hernehmen, welche näher an der natülichen Sprache ist 🡪 Kommentare von Rezepten, Koch FAQs, Untertitel von Kochshows, …

# Bibliography

Adamopoulou, E., & Moussiades, L. (2020, June). An overview of chatbot technology. *IFIP International Conference on Artificial Intelligence Applications and Innovations*, 373–383.

Alammar, J. (2018a). *The Illustrated Transformer [Blog post]*. https://jalammar.github.io/illustrated-transformer/

Alammar, J. (2018b). *Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models With Attention) [Blog post]*. https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Angara, P., Jimenez, M., Agarwal, K., Jain, H., Jain, R., Stege, U., Ganti, S., Müller, H. A., & Ng, J. W. (2017). Foodie fooderson a conversational agent for the smart kitchen. *CASCON*, 247–253.

Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016, July 21). *Layer Normalization*. http://arxiv.org/pdf/1607.06450v1

Bahdanau, D., Cho, K., & Bengio, Y. (2014, September 1). *Neural Machine Translation by Jointly Learning to Align and Translate*. http://arxiv.org/pdf/1409.0473v7

Bahdanau, D., Chorowski, J., Serdyuk, D., Brakel, P., & Bengio, Y. (2016, March). End-to-end attention-based large vocabulary speech recognition. *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 4945–4949. https://doi.org/10.1109/ICASSP.2016.7472618

Barko-Sherif, S., Elsweiler, D., & Harvey, M. (2020, March). Conversational Agents for Recipe Recommendation. *Proceedings of the 2020 Conference on Human Information Interaction and Retrieval*, 73–82. https://doi.org/10.1145/3343413.3377967

Bickmore, T. W., Caruso, L., & Clough-Gorr, K. (2005). Acceptance and usability of a relational agent interface by urban older adults. *CHI'05 Extended Abstracts on Human Factors in Computing Systems*, 1212–1215. https://doi.org/10.1145/1056808.1056879

Bień, M., Gilski, M., Maciejewska, M., Taisner, W., Wisniewski, D., & Lawrynowicz, A. (2020). RecipeNLG: A Cooking Recipes Dataset for Semi-Structured Text Generation. *Proceedings of the 13th International Conference on Natural Language Generation*, 22–28. https://aclanthology.org/2020.inlg-1.4

Brandtzaeg, P. B., & Følstad, A. (2017). Why People Use Chatbots. In I. Kompatsiaris, J. Cave, A. Satsiou, G. Carle, A. Passani, E. Kontopoulos, S. Diplaris, & D. McMillan (Eds.), *Lecture Notes in Computer Science. Internet Science* (Vol. 10673, pp. 377–392). Springer International Publishing. https://doi.org/10.1007/978-3-319-70284-1\_30

Campos, J. A., Otegi, A., Soroa, A., Deriu, J., Cieliebak, M., & Agirre, E. (2020). DoQA -- Accessing Domain-Specific FAQs via Conversational QA. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 7302–7314. https://doi.org/10.48550/arXiv.2005.01328

Chao, G.-L., & Lane, I. (2019, July 6). *BERT-DST: Scalable End-to-End Dialogue State Tracking with Bidirectional Encoder Representations from Transformer*. http://arxiv.org/pdf/1907.03040v1

Chen, D., Fisch, A., Weston, J., & Bordes, A. (2017, March 31). *Reading Wikipedia to Answer Open-Domain Questions*. http://arxiv.org/pdf/1704.00051v2

Chen, Q., Zhuo, Z., & Wang, W. (2019). *BERT for Joint Intent Classification and Slot Filling*. http://arxiv.org/pdf/1902.10909v1

Cheng, J., Dong, L., & Lapata, M. (2016, January 25). *Long Short-Term Memory-Networks for Machine Reading*. http://arxiv.org/pdf/1601.06733v7

Chu, J. (2021, September 24–26). Recipe Bot: The Application of Conversational AI in Home Cooking Assistant. In *2021 2nd International Conference on Big Data & Artificial Intelligence & Software Engineering (ICBASE)* (pp. 696–700). IEEE. https://doi.org/10.1109/ICBASE53849.2021.00136

Cui, L., Huang, S., Wei, F., Tan, C., Duan, C., & Zhou, M. (2017). Superagent: A customer service chatbot for e-commerce websites. *Proceedings of ACL 2017, System Demonstrations*, 97–102.

Dai, A. M., & Le, Q. V. (2015). Semi-supervised Sequence Learning. *Advances in Neural Information Processing Systems*, *28*, 3079–3087. http://arxiv.org/pdf/1511.01432v1

Devlin, J., & Chang, M.-W. (2018). *Open Sourcing BERT: State-of-the-Art Pre-training for Natural Language Processing.* Google AI. https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018, October 11). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. http://arxiv.org/pdf/1810.04805v2

Donnelly, K. (2006). Snomed-CT: The advanced terminology and coding system for eHealth. *Studies in Health Technology and Informatics*, *121*, 279–290.

Dooley, D. M., Griffiths, E. J., Gosal, G. S., Buttigieg, P. L., Hoehndorf, R., Lange, M. C., Schriml, L. M., Brinkman, F. S. L., & Hsiao, W. W. L. (2018). Foodon: A harmonized food ontology to increase global food traceability, quality control and data integration. *NPJ Science of Food*, *2*, 23. https://doi.org/10.1038/s41538-018-0032-6

Elsweiler, D., Harvey, M., Ludwig, B., & Said, A. (2015). Bringing the "healthy" into Food Recommenders. *DMRS*, 33–36.

Elsweiler, D., Trattner, C., & Harvey, M. (2017). Exploiting food choice biases for healthier recipe recommendation. *Proceedings of the 40th International Acm Sigir Conference on Research and Development in Information Retrieval*, 575–584. https://doi.org/10.1145/3077136.3080826

Firth, J. R. (1957). A synopsis of linguistic theory, 1930-1955. *Studies in Linguistic Analysis*.

Følstad, A., & Brandtzæg, P. B. (2017). Chatbots and the new world of HCI. *Interactions*, *24*(4), 38–42. https://doi.org/10.1145/3085558

Freyne, J., & Berkovsky, S. (2010). Intelligent food planning: personalized recipe recommendation. *Proceedings of the 15th International Conference on Intelligent User Interfaces*, 321–324. https://doi.org/10.1145/1719970.1720021

Frummet, A., Elsweiler, D., & Ludwig, B. (2021, December 9). *"What can I cook with these ingredients?" -- Understanding cooking-related information needs in conversational search*. http://arxiv.org/pdf/2112.04788v2

Graesser, A. C., Wiemer-Hastings, K., Wiemer-Hastings, P., & Kreuz, R. (1999). AutoTutor: A simulation of a human tutor. *Cognitive Systems Research*, *1*(1), 35–51. https://doi.org/10.1016/S1389-0417(99)00005-4

Gururangan, S., Marasović, A., Swayamdipta, S., Lo Kyle, Beltagy, I., Downey, D., & Smith, N. A. (2020, April 23). *Don't Stop Pretraining: Adapt Language Models to Domains and Tasks*. http://arxiv.org/pdf/2004.10964v3

Harris, Z. S. (1954). Distributional Structure. *WORD*, *10*(2-3), 146–162. https://doi.org/10.1080/00437956.1954.11659520

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

Howard, J., & Ruder, S. (2018, January 18). *Universal Language Model Fine-tuning for Text Classification*. http://arxiv.org/pdf/1801.06146v5

Huffman, S. (2019). *Here’s how the Google Assistant became more helpful in 2018.* Google. https://www.blog.google/products/assistant/heres-how-google-assistant-became-more-helpful-2018/

IBM Cloud Education. (2020a, August 17). *Neural Networks*. https://www.ibm.com/cloud/learn/neural-networks

IBM Cloud Education. (2020b, September 14). *Recurrent Neural Networks*. https://www.ibm.com/cloud/learn/recurrent-neural-networks

Joos, M. (1950). Description of Language Design. *The Journal of the Acoustical Society of America*, *22*(6), 701–707. https://doi.org/10.1121/1.1906674

Jurafsky, D., & Martin, J. H. (Eds.). (2021). *Speech and Language Processing: 3rd ed. draft*. https://web.stanford.edu/~jurafsky/slp3/

Kinsella, B., & Mutchler, A. (April 2020). *Smart Speaker Consumer Adoption Report Executive Summary.* Voicebot.ai.

Kobayashi, S. (2018). *Homemade BookCorpus*. https://github.com/BIGBALLON/cifar-10-cnn

Konle, L., & Jannidis, F. (2020). Domain and Task Adaptive Pretraining for Language Models. *CHR*.

Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H., & Kang, J. (2020). Biobert: A pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics (Oxford, England)*, *36*(4), 1234–1240. https://doi.org/10.1093/bioinformatics/btz682

Li, J., Chen, X., Hovy, E., & Jurafsky, D. (2016, June). Visualizing and understanding neural models in nlp. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 681–691. https://doi.org/10.18653/v1/N16-1082

Lin, Z., Feng, M., Santos, C. N. d., Yu, M., Xiang, B., Zhou, B., & Bengio, Y. (2017, March 9). *A Structured Self-attentive Sentence Embedding*. http://arxiv.org/pdf/1703.03130v1

Luong, M.-T., Pham, H., & Manning, C. D. (2015, August 17). *Effective Approaches to Attention-based Neural Machine Translation*. http://arxiv.org/pdf/1508.04025v5

Marin, J., Biswas, A., Ofli, F., Hynes, N., Salvador, A., Aytar, Y., Weber, I., & Torralba, A. (2019). Recipe1m+: A Dataset for Learning Cross-Modal Embeddings for Cooking Recipes and Food Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *43*(1), 187–203. https://doi.org/10.1109/TPAMI.2019.2927476

McTear, M. (2020). Conversational AI: Dialogue Systems, Conversational Agents, and Chatbots. *Synthesis Lectures on Human Language Technologies*, *13*(3), 1–251. https://doi.org/10.2200/S01060ED1V01Y202010HLT048

Merity, S., Xiong, C., Bradbury, J., & Socher, R. (2016, September 26). *Pointer Sentinel Mixture Models*. http://arxiv.org/pdf/1609.07843v1

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013, January 16). *Efficient Estimation of Word Representations in Vector Space*. http://arxiv.org/pdf/1301.3781v3

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, *26*.

Nallapati, R., Xiang, B., & Zhou, B. (2016). Sequence-to-sequence rnns for text summarization.

Ni, L., Lu, C., Liu, N., & Liu, J. (2017). MANDY: Towards a Smart Primary Care Chatbot Application. In J. Chen, T. Theeramunkong, T. Supnithi, & X. Tang (Eds.), *Communications in Computer and Information Science. Knowledge and Systems Sciences* (Vol. 780, pp. 38–52). Springer Singapore. https://doi.org/10.1007/978-981-10-6989-5\_4

Nimavat, K., & Champaneria, T. (2017). Chatbots: An overview types, architecture, tools and future possibilities. *Int. J. Sci. Res. Dev*, *5*(7), 1019–1024.

Oinkina. (2015). *Understanding LSTM Networks*. https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Pan, S. J., & Yang, Q. (2010). A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*(10), 1345–1359. https://doi.org/10.1109/TKDE.2009.191

Parikh, A. P., Täckström, O., Das, D., & Uszkoreit, J. (2016, June 6). *A Decomposable Attention Model for Natural Language Inference*. http://arxiv.org/pdf/1606.01933v2

Paulus, R., Xiong, C., & Socher, R. (2017, May 11). *A Deep Reinforced Model for Abstractive Summarization*. http://arxiv.org/pdf/1705.04304v3

Pellegrini, C., Özsoy, E., Wintergerst, M., & Groh, G. (2021, February 11–13). Exploiting Food Embeddings for Ingredient Substitution. In *Proceedings of the 14th International Joint Conference on Biomedical Engineering Systems and Technologies* (pp. 67–77). SCITEPRESS - Science and Technology Publications. https://doi.org/10.5220/0010202000670077

Pennington, J., Socher, R., & Manning, C. (2014, October). Glove: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543. https://doi.org/10.3115/v1/D14-1162

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018, February 15). *Deep contextualized word representations*. http://arxiv.org/pdf/1802.05365v2

Popovski, G., Kochev, S., Korousic-Seljak, B., & Eftimov, T. (2019, February). FoodIE: A Rule-based Named-entity Recognition Method for Food Information Extraction. *ICPRAM*, 915–922.

Popovski, G., Koroušić Seljak, B., & Eftimov, T. (2019). *Foodontomap: Linking Food Concepts across different Food Ontologies.* https://doi.org/10.5281/zenodo.2635437

Popovski, G., Seljak, B. K., & Eftimov, T. (2019). Foodbase corpus: A new resource of annotated food entities. *Database : The Journal of Biological Databases and Curation*, *2019.* https://doi.org/10.1093/database/baz121

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). *Improving language understanding by generative pre-training.*

Ramshaw, L. A., & Marcus, M. P. (1999). Text chunking using transformation-based learning. *Natural Language Processing Using Very Large Corpora*, 157–176.

Roffo, G. (2017, June 1). *Ranking to Learn and Learning to Rank: On the Role of Ranking in Pattern Recognition Applications*. http://arxiv.org/pdf/1706.05933v1

Ruder, S. (2019). *Neural transfer learning for natural language processing* [Doctoral dissertation]. NUI Galway.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). *Learning internal representations by error propagation.* California Univ San Diego La Jolla Inst for Cognitive Science.

Rush, A. M. (2018, July). The Annotated Transformer. *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, 52–60. https://doi.org/10.18653/v1/W18-2509

Stojanov, R., Popovski, G., Cenikj, G., Koroušić Seljak, B., & Eftimov, T. (2021). A Fine-Tuned Bidirectional Encoder Representations From Transformers Model for Food Named-Entity Recognition: Algorithm Development and Validation. *Journal of Medical Internet Research*, *23*(8), e28229. https://doi.org/10.2196/28229

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Dvances in Neural Information Processing Systems*, *27*.

Taylor, W. L. (1953). “Cloze Procedure”: A New Tool for Measuring Readability. *Journalism Quarterly*, *30*(4), 415–433. https://doi.org/10.1177/107769905303000401

Vakulenko, S., Longpre, S., Tu, Z., & Anantha, R. (2021). Question Rewriting for Conversational Question Answering. *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, 355–363. https://doi.org/10.1145/3437963.3441748

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017, June 12). *Attention Is All You Need*. http://arxiv.org/pdf/1706.03762v5

Venugopalan, S., Rohrbach, M., Donahue, J., Mooney, R., Darrell, T., & Saenko, K. (2015). Sequence to sequence-video to text. *Proceedings of the IEEE International Conference on Computer Vision*, 4534–4542.

Voskarides, N., Li, D., Ren, P., Kanoulas, E., & Rijke, M. de (2020). Query Resolution for Conversational Search with Limited Supervision. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 921–930. https://doi.org/10.1145/3397271.3401130

Weizenbaum, J. (1966). ELIZA-a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, *9*(1), 36–45.

Winkler, R., Hobert, S., Salovaara, A., Söllner, M., & Leimeister, J. M. (2020). Sara, the Lecturer: Improving Learning in Online Education with a Scaffolding-Based Conversational Agent. *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 1–14. https://doi.org/10.1145/3313831.3376781

Wu, Y., Schuster, M., Chen, Z., Le V, Q., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, Ł., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., . . . Dean, J. (2016, September 26). *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. http://arxiv.org/pdf/1609.08144v2

Xu, L., Zhou, Q., Gong, K., Liang, X., Tang, J., & Lin, L. (2019). End-to-End Knowledge-Routed Relational Dialogue System for Automatic Diagnosis. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*, 7346–7353. https://doi.org/10.1609/aaai.v33i01.33017346

Yang, W., Xie, Y., Lin, A., Li, X., Tan, L., Xiong, K., Li, M., & Lin, J. (2019). End-to-End Open-Domain Question Answering with BERTserini, 72–77. https://doi.org/10.18653/v1/N19-4013

Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2021, June 21). *Dive into Deep Learning*. http://arxiv.org/pdf/2106.11342v2

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., & Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *Proceedings of the IEEE International Conference on Computer Vision*, 19–27.

Anhang A: Bausteine wissenschaftlicher Arbeiten

## A1 Theoretische Arbeit

1. Fragestellung (Ziele, Motivation)
2. Überblick über Stand der Forschung und Technik (dabei Bewertung der Ansätze, Beispiele, Identifikation von Defiziten)
3. Synthese: Erstellung einer Gesamtschau (allgemeine Prinzipien, Beschreibung einer eigenen Sicht auf das Problem, Formulierung von Empfehlungen )
4. Zusammenfassung (Was wurde in der Arbeit erreicht, Erklärung des Nutzens für andere)
5. Ausblick (optional)

## A2 Konstruktive Arbeit

1. Problemstellung (Ziele, Ausgangspunkt, Vorgesehener Benutzerkreis, Bedürfnisse der Benutzer)
2. Stand der Forschung und Technik (Bisherige Lösungen, Defizite)
3. Eigenes Konzept (Lösungsansatz, allgemeines Prinzip, Werkzeuge z.B. Programmiersprachen )
4. Vorgehensweise (Beschreibung der durchgeführten Arbeitsschritte)
5. Ergebnis (Vorstellung des System z.B. Screenshots mit Erläuterungen)
6. Evaluation des System (optional, was soll evaluiert werden, welche Methode, Ablauf, Ergebnisse)
7. Zusammenfassung (Was wurde in der Arbeit erreicht; Erklärung des Nutzens für andere)
8. Ausblick (optional)

## A3 Empirische Arbeit

1. Fragestellung der Arbeit (Was soll untersucht werden, warum)
2. Stand der Forschung und Technik (Bewertung der Untersuchungs-Ansätze und Ergebnisse, Identifikation von Defiziten)
3. Präzisierung der Fragestellung (Hypothesen)
4. Untersuchungsmethodik
5. Untersuchungsablauf (Untersuchungsmaterial, Raum, Probandenrekrutierung etc.)
6. Ergebnisse (Darstellung der Ergebnisse in sinnvoller Reihenfolge, Gesamtüberblick, Einzelergebnisse z. B. geordnet nach Testcases)
7. Zusammenfassung (Was wurde erreicht, Rückbezug zu Zielen, Hypothesen, Nutzen, Erkenntnisse für weitere Untersuchungen)
8. Ausblick (optional)

Erklärung zur Urheberschaft

Ich habe die Arbeit selbständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie alle Zitate und Übernahmen von fremden Aussagen kenntlich gemacht.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt.

Die vorgelegten Druckexemplare und die vorgelegte digitale Version sind identisch.

|  |  |  |
| --- | --- | --- |
| Ort, Datum |  | Unterschrift |

Erklärung zur Lizenzierung und Publikation dieser Arbeit

**Name:** Pascal Strobel

**Titel der Arbeit:** CookBERT

In der Regel räumen Sie mit Abgabe der Arbeit dem Lehrstuhl für Medieninformatik nur zwingend das Recht ein, dass die Arbeit zur Bewertung gelesen, gespeichert und vervielfältigt werden darf. Idealerweise liefern Seminararbeiten, Projektdokumentationen und Abschlussarbeiten aber einen Erkenntnisgewinn, von dem auch andere profitieren können. Wir möchten Sie deshalb bitten, uns weitere Rechte einzuräumen, bzw. idealerweise Ihre Arbeit unter eine freie Lizenz zu stellen.

Die in unseren Augen praktikabelsten Lösungen sind vorselektiert.

Hiermit gestatte ich die Verwendung der **schriftlichen Ausarbeitung** zeitlich unbegrenzt und nicht-exklusiv unter folgenden Bedingungen:

Nur zur Bewertung dieser Arbeit

Nur innerhalb des Lehrstuhls im Rahmen von Forschung und Lehre

Unter einer Creative-Commons-Lizenz mit den folgenden Einschränkungen:

BY – Namensnennung des Autors

NC – Nichtkommerziell

SA – Share-Alike, d.h. alle Änderungen müssen unter die gleiche Lizenz gestellt werden.

(An Zitaten und Abbildungen aus fremden Quellen werden keine weiteren Rechte eingeräumt.)

Außerdem gestatte ich die Verwendung des im Rahmen dieser Arbeit erstellen **Quellcodes** unter folgender Lizenz:

Nur zur Bewertung dieser Arbeit

Nur innerhalb des Lehrstuhls im Rahmen von Forschung und Lehre

Unter der CC-0-Lizenz (= beliebige Nutzung)

Unter der MIT-Lizenz (= Namensnennung)

Unter der GPLv3-Lizenz (oder neuere Versionen)

(An explizit mit einer anderen Lizenz gekennzeichneten Bibliotheken und Daten werden keine weiteren Rechte eingeräumt.)

Ich willige ein, dass der Lehrstuhl für Medieninformatik diese Arbeit – falls sie besonders gut ausfällt - auf dem Publikationsserver der Universität Regensburg veröffentlichen lässt.

Ich übertrage deshalb der Universität Regensburg das Recht, die Arbeit elektronisch zu speichern und in Datennetzen öffentlich zugänglich zu machen. Ich übertrage der Universität Regensburg ferner das Recht zur Konvertierung zum Zwecke der Langzeitarchivierung unter Beachtung der Bewahrung des Inhalts (die Originalarchivierung bleibt erhalten).

Ich erkläre außerdem, dass von mir die urheber- und lizenzrechtliche Seite (Copyright) geklärt wurde und Rechte Dritter der Publikation nicht entgegenstehen.

Ja, für die komplette Arbeit inklusive Anhang

Ja, für eine um vertrauliche Informationen gekürzte Variante (auf dem Datenträger beigefügt)

Nein

Sperrvermerk bis (Datum):

Regensburg, 10.03.2022

|  |  |  |
| --- | --- | --- |
| Ort, Datum |  | Unterschrift |

Inhalt des beigefügten Datenträgers

Beispiel (Ordner + Beschreibung):

|  |  |  |
| --- | --- | --- |
| /1\_Ausarbeitung | Die schriftliche Ausarbeitung als PDF und DOC | |
| /2\_Code | Quellcode und kompilierte Anwendung des Prototypen | |
| /3\_Studie/Design | Fragebogen und Script für die Benutzerstudie | |
| /3\_Studie/Rohdaten | Rohdaten der Studie im CSV-Format, inkl. Beschreibung der Felder | |
| /4\_Quellen | Alle in der Arbeit zitierten Quellen im PDF-Format | |
| /5\_Bilder | Alle selbst erstellten und aus anderen Quellen übernommenen Bilder | |
| /6\_Vorträge | Folien von Antritts- und Abschlussvortrag im PDF-Format | |
| /7\_Sonstiges | Notizen aus Besprechungen, Gedanken, … | |
|  | |

[Datenträger (CD, SD-Karte, o.ä.) hier oder auf Umschlaginnenseite einkleben]

1. https://www.amazon.com/smart-home-devices/b?ie=UTF8&node=9818047011 [↑](#footnote-ref-1)
2. https://www.apple.com/siri/ [↑](#footnote-ref-2)
3. <https://www.english-corpora.org/hansard/> (Retrieved on March 5, 2022) [↑](#footnote-ref-3)
4. The adapted FoodBase corpus from Stojanov et al. (2021) is publicly available at: <https://github.com/ds4food/FoodNer> (Retrieved on March 5, 2022) [↑](#footnote-ref-4)