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: 28.03.2022

Contents

List of Figures

List of Tables

Zusammenfassung

Konversationsassistenten sind inzwischen allgegenwärtig und in einem breiten Spektrum von Anwendungskontexten wiederzufinden. Während sie früher auf handgemachten Regeln basierten, sind heutzutage hauptsächlich datengesteuerte Deep Learning Ansätze gebräuchlich. Vor allem das 2018 von Google vorgestellte BERT hat sich durch seine bis dahin einzigartigen Eigenschaften und seinem damit verbundenen, hervorragenden Sprachverständnis, für die Verarbeitung natürlicher Sprache etabliert und findet damit auch Anwendung im Bereich der Conversational AI. Allerdings kann es aufgrund BERT's standarmäßiger Ausrichtung auf die eher generelle Sprachdomäne zu Performanceeinbußen bei domänenspezifischen Daten bzw. Aufgaben kommen.

Ziel dieser Arbeit ist es, BERT durch Domänenadaption für die Kochdomäne anzupassen, sodass es für mehrere Aufgaben von Konversationsassistenten dieser Domäne eingesetzt werden kann. Dazu wird die folgende Forschungsfrage gestellt: Wie wirkt sich BERT's Kochdomänenadaption auf die Performance von Aufgaben aus, die für Konversationsassistenten relevant sind. Um die Frage zu beantworten wurde zunächst CookBERT eingeführt, ein domänenspezifisches BERT-Modell, das mithilfe von domänenadaptivem Vortraining und Vokabularerweiterung für die Kochdomäne angepasst wurde. Anschließend wurde CookBERT anhand mehrerer Aufgaben evaluiert, die für Konversationsassistenten von Relevanz sind, darunter die Klassifikation von kochspezifischen Informationsbedürfnissen, das Taggen von Nahrungsmittel-Entitäten, sowie die Extraktion von Antwortspannen (Question Answering). Die Ergebnisse zeigen, dass CookBERT bei der Klassifikation von kochspezifischen Informationsbedürfnissen, sowie dem Taggen von Nahrungsmittel-Entitäten zu signifikanter Performancesteigerung führen kann. Anders ist es beim Question Answering, bei welchem die vorgeschlagene Domänenadaption keine positiven Effekte zeigt. Generell kann eine derartige Domänenadaption also die Performance bei Aufgaben steigern, die für Konversationsassistenten relevant sind, das muss allerdings nicht immer der Fall sein und hängt von der jeweiligen Aufgabe ab.

Weiterführende Forschung könnte CookBERT zusätzlich verfeinern, oder das Modell für eigene Aufgaben anwenden und testen, wie z.B. die Abfrageauflösung.

Abstract

Conversational agents are now ubiquitous and can be found in a broad range of contexts. While they used to be based on handcrafted rules, nowadays mainly data-driven deep learning approaches are common. Above all, the BERT model presented by Google in 2018 has established itself for the processing of natural language due to its unique properties and its associated excellent language understanding and is therefore also used in the field of conversational AI. However, BERT lacks domain specific knowledge since it was only pretrained on text data from the general domain, which can result in performance degradation for domain specific data or tasks.

The aim of this work is to adjust BERT via domain adaptation to the specific domain of cooking, so that it can be used for several tasks by conversation assistants in this domain. For this purpose, the following research question is asked: How does domain adaptation for the cooking domain affect the performance of BERT on tasks that are relevant for kitchen conversational agents. To answer the question, first, CookBERT is introduced, a domain specific BERT model that was adapted for the cooking domain using domain adaptive pretraining and vocabulary expansion. Subsequently, CookBERT was evaluated on several tasks relevant to conversational agents, including the classification of cooking-specific information needs, the tagging of food entities, and the extraction of answer spans (question answering). The results show that CookBERT can lead to a significant increase in performance in the classification of cooking-specific information needs and the tagging of food entities. It is different with question answering, in which the proposed domain adaptation does not show any positive effects. In general, domain adaptation as done in this thesis, can increase the performance of tasks that are relevant for conversation agents, but this does not always have to be the case and depends on the respective task.

Future research could further refine CookBERT or apply and test the model for custom tasks, e.g., query resolution.

1 Introduction

Conversational agents (CAs) like Amazon's Echo or Apple's Siri 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), or customer service (Cui et al., 2017). 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 increased ease of use, and speed of user requests as well as convenient usage (Brandtzaeg & Følstad, 2017). While early approaches for creating CAs were mainly based on handcrafted rules like Weizenbaum's ELIZA (1966), this has shifted in recent years towards the utilization of large-scale pretrained language models which can attain 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), pretrained on a massive amount of plain text data from the general domain. 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" (Devlin & Chang, 2018). This bidirectionality, combined with the so-called 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 subsequent open sourcing ensured that it was also applied in the field of conversational artificial intelligence where it achieves promising performance for a variety of tasks as well (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 information needs that are specific for the cooking domain, 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 e-commerce websites. However, one of BERT's limitations is the lack of domain specific knowledge, since pre-training 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 to overcome this limitation by adapting BERT for a particular target domain, i.e., the domain of cooking. Doing this is expected to increase its natural language understanding of the target domain and with it the performance on tasks relevant to CAs of that 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 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 the 2018th Christmas season, passing one million on Christmas day alone (Huffman, 2019). Providing a cooking domain specific BERT model might be an important step towards building a truly conversational system for the kitchen. Therefore, 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, including information need classification, food entity tagging, and question answering,
- evidence that CookBERT can significantly improve performance in the classification of cooking specific information needs and tagging of food entities, compared to the default BERT model,
- evidence that CookBERT outperforms FoodBERT (Pellegrini et al., 2021), an antecedent cooking domain specific BERT model, in all three evaluation tasks.

The remainder of this thesis is organized as follows. Chapter 2 covers background and related work that is necessary for the understanding of basic concepts of this thesis and for the justification of methodological steps taken, respectively. Chapter 3 outlines these methodological steps taken to develop CookBERT as well as the experimental setup for its evaluation. The actual performance evaluation of CookBERT and the comparison with other BERT based models is subsequently done in chapter 4. Chapter 5 discusses the results of the evaluation, followed by the limitations that this thesis faced in chapter 6. The thesis is rounded off with a conclusion in chapter 7.

2 Related Work

To set the context for this thesis and motivate the research question as well as 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 artificial intelligence to word embeddings to the recently popular transformer neural networks. It starts with the topic of conversational artificial intelligence in section 2.1, in which on the one hand the choice of the evaluation tasks used later in this thesis are justified and on the other hand the cooking domain is motivated once again as the domain of choice. Section 2.2 covers relevant concepts and main architectures for processing natural language with deep neural networks. It serves as the background for understanding the concepts on which BERT is based on. Subsequently, the BERT model itself is addressed in section 2.3. This section is intended to explain why BERT of all things is the model of choice in this thesis. It also contains related work regarding the domain adaptation of BERT, which forms the basis for the methodological decisions taken. In section 2.4, datasets used throughout this thesis are presented. Finally, there is a summary of the most important findings from this chapter, and key differentiators are outlined, which thus leads to the formulation of the underlying research question of this thesis.

2.1 Conversational Artificial Intelligence

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

2.1.1 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, quoted from Khatri et al., 2018). Compared to the typical interaction via command-line or graphical user interfaces, such systems provide the opportunity for a more natural interaction via conversation and thus promise an increased ease of use and speed of user requests as well as 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 released 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 the advances in deep learning in recent years, some of which will be discussed in chapter 2.2, 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 Echo 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.

2.1.2 Inner Workings of Conversational Agents

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 of 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.

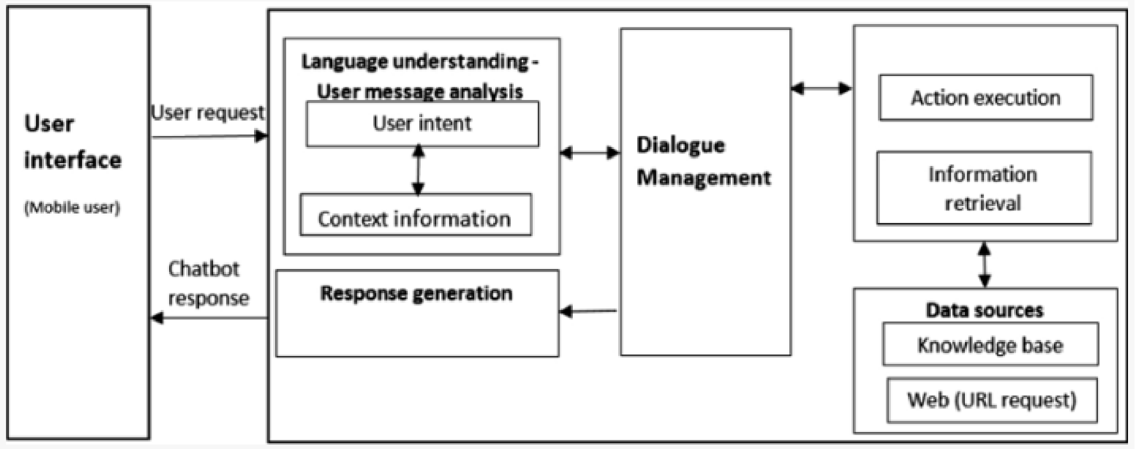

Figure 1: General architecture of conversational agents. (Taken from Adamopoulou and Moussiades (2020, p. 380))

According to Figure 1, 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 the 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 or information need that the user tries to satisfy 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 (Adamopoulou & Moussiades, 2020, pp. 378–379). As the name suggests, the rule-based approach makes use of a handwritten, 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, W. 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 as question answering in literature.

2.1.3 Application in the Cooking Domain

In past research, the kitchen has been considered 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 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 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 and was proposed by 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 Watson's 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 user's intent and the entities block 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.

Crucial work for the future development of CAs for the kitchen was done by Frummet et al. (2021). The researchers conducted an in-situ study in order to "understand information needs arising in a home cooking context as well as how they are verbally communicated to an assistant" (Frummet et al., 2021, p. 1). Based on this, the researchers identified a variety of different information needs and provide a detailed hierarchical taxonomy. As part of their work, they also created the *Cookversational Search* dataset, which is discussed in more detail in section 2.4.2. Furthermore, they use this dataset, to investigate the feasibility of classifying information needs based the user utterances. Several baseline classifiers and BERT models were applied, with the BERT models generally giving better results. The results indicate the feasibility of predicting such information needs automatically, but the researchers emphasize the necessity for further improvements, e.g., through better inclusion of context in the form of conversational history.

Barko-Sherif et al. (2020) also provide pivotal work for the development of future CAs for the kitchen. In particular, they investigated the interaction between humans and a CA for recipe recommendation, using the wizard of oz method. They show that although the conversational interaction with such an agent can be diverse, their proposed framework that models the conversational flow, proves very solid, and many conversations from the study followed it, which is encouraging for the future development of such CAs.

2.2 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 ANNs 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 2.2.4 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.

2.2.1 Word Embeddings

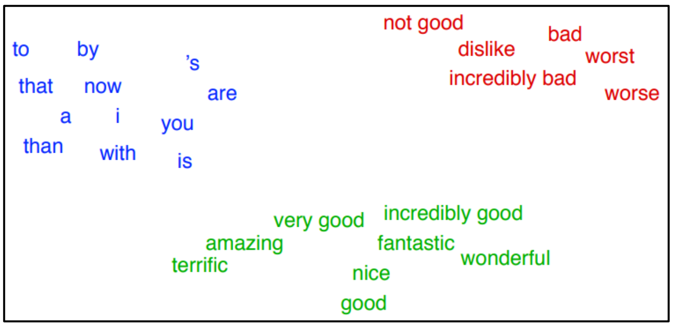

Figure 2: Two-dimensional projection of word embeddings. Note how similar words are nearby in space. (Taken from Jurafsky and Martin (2021, p. 107) as a simplified representation of Li et al. (2016))

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 terms, vectors of numbers that capture the meaning of words. An example of what embeddings can look like in two-dimensional space is given in Figure 2. (Jurafsky & Martin, 2021, pp. 106–107)

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 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*, by 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 2.2.3) which were trained on the task of language modelling (predicting next word in a sequence given previous words) and backwards language modelling (predict preceding word of a sequence 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.

2.2.2 Artificial Neural Networks

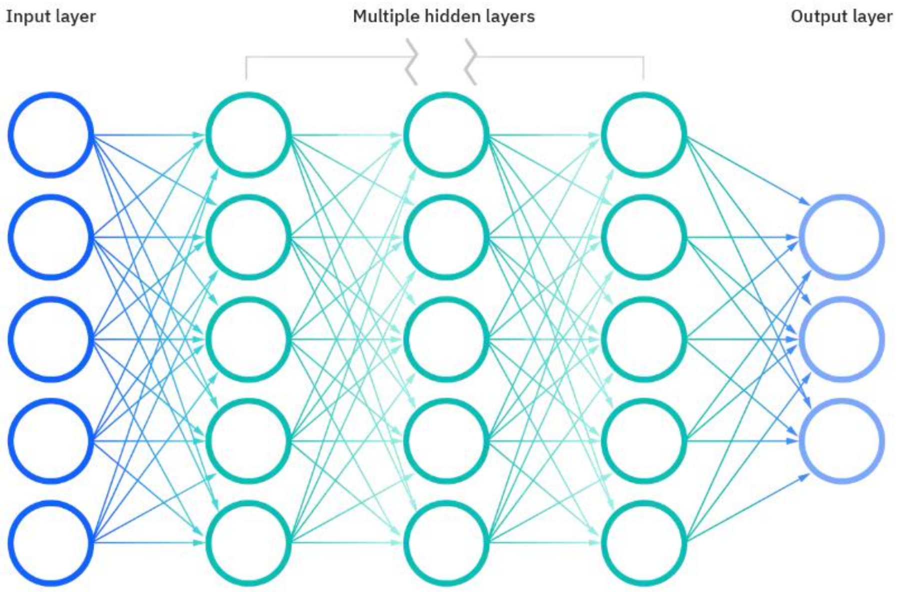

Figure 3: Deep neural network with three hidden layers. Arrows indicate the direction of the information flow. (Taken from IBM Cloud Education (2020a))

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, 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 three hidden layers is given in Figure 3.

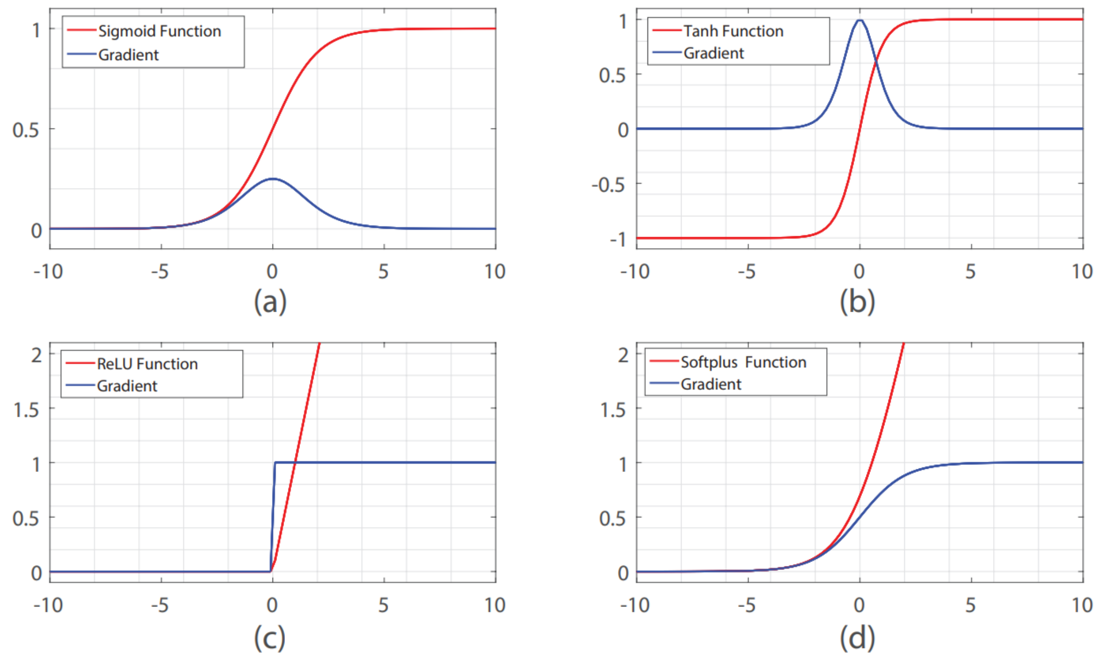

Figure 4: Commonly used activation functions. (Taken from Roffo (2017, p. 40))

Each neuron accepts a single or multiple numeric values $x_{1,}\dots,x_n$ as input, each of which has a weight w 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 y of a neuron, a bias b is added to the sum of the weighted inputs and the result is then fed through a so-called activation function f. The output of a single neuron can thus be determined by the following equation:

$$y = f(\sum_{i=1}^{n} x_i w_i + b)$$

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 4. 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 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*. (Jurafsky & Martin, 2021, pp. 148–155)

2.2.3 Recurrent Neural Networks

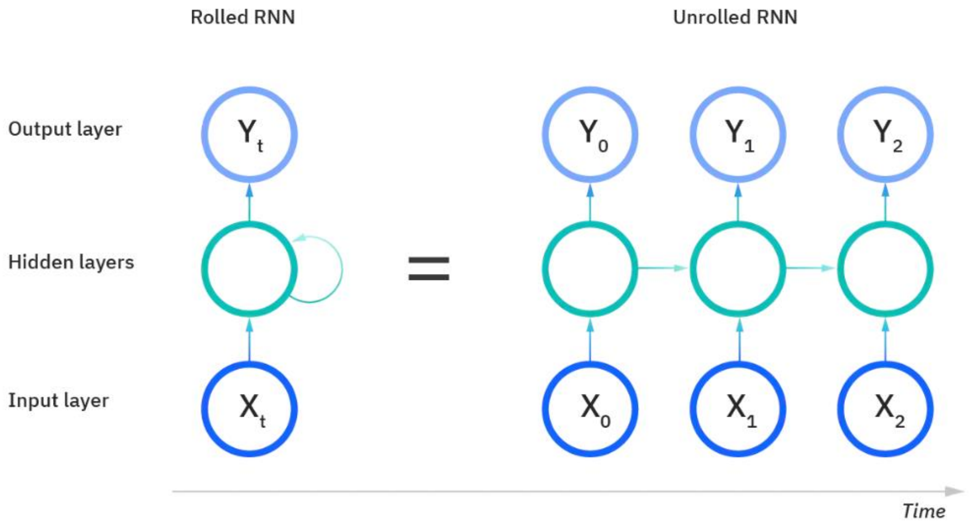

Figure 5: Structure of a RNN when unrolled. (Taken from IBM Cloud Education (2020b))

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 with 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 representation, showing that it becomes a feedforward neural network 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 (Rumelhart et al., 1985; Werbos, 1990) 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. (Jurafsky & Martin, 2021, pp. 186–198)

Long Short-Term Memory

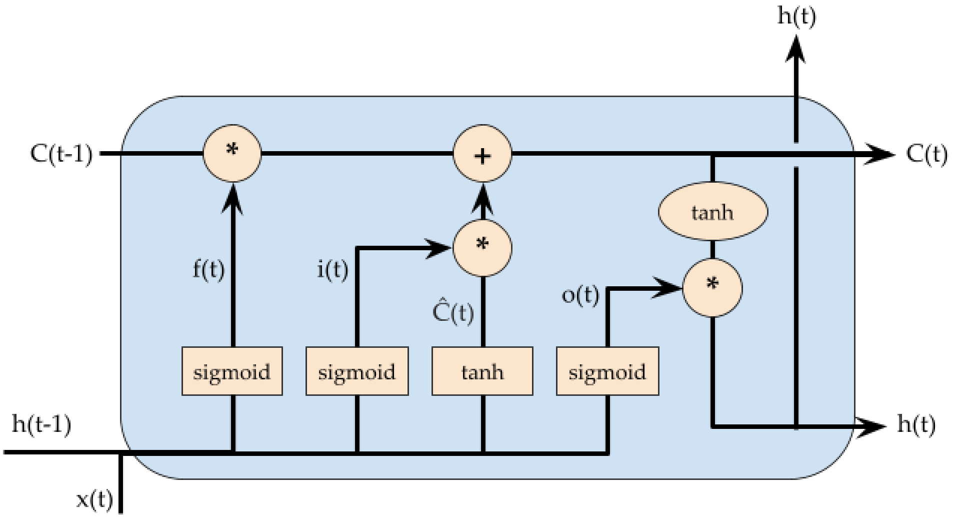

Figure 6: Structure of a single LSTM cell. (Based on Oinkina (2015))

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, shown in Figure 6. In addition to the normal hidden state h, a LSTM cell also maintains a cell state C 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 i_t, the forget gate f_t, and the output gate o_t. 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:

$$f_t = \sigma\,(W_f \cdot [h_{t-1}, x_t] + b_f)$$
$$i_t = \sigma\,(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\hat{C}_t = tanh\,(W_C \cdot [h_{t-1}, x_t] + b_C)$$
$$C_t = f_t * C_{t-1} + i_t * \hat{C}_t$$
$$o_t = \sigma\,(W_o \cdot [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * tanh(C_t)$$

where x_t denotes the input vector at time step t. $\hat{C}$ can be seen as a supportive gate that computes how much to write to the cell state C. W_f, W_i, W_C, W_o and b_f, b_i, b_C, b_o 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 σ. 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. (Jurafsky & Martin, 2021, pp. 196–200)

2.2.4 Encoder-decoder Architecture

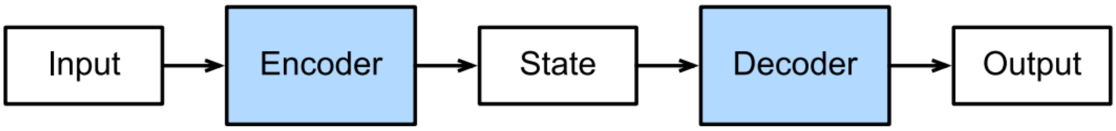

Figure 7: Encoder-decoder architecture. (Taken from Zhang et al. (2021, p. 378))

Encoder-decoder is a specific neural network architecture that was proposed by Sutskever et al. (2014) 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).

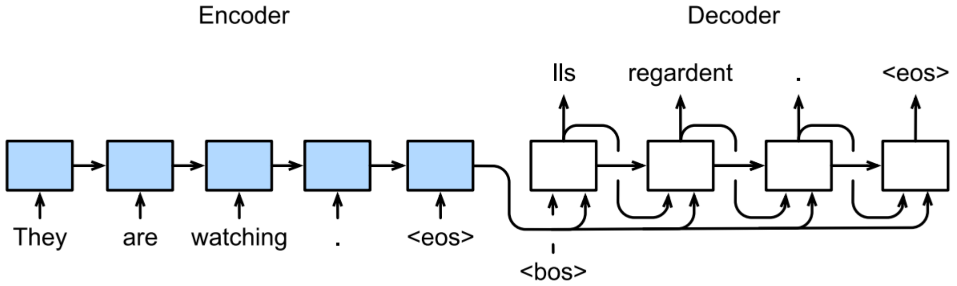

Figure 8: Machine translation represented as a sequence-to-sequence learning problem with a RNN encoder and a RNN decoder. (Taken from Zhang et al. (2021, p. 380))

The architecture consists of two major components, illustrated in Figure 7. 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 (denoted as “State” in Figure 7) 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 RNN architecture, especially LSTM, 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 they use information from previous steps in order to output the hidden state and predicted word, respectively (see Figure 8). While the default encoder-decoder architecture works fine for short input sequences, it struggles with longer ones, because it is difficult for the encoder to compress all the contextual information of a long sequence into a single fixed-size vector, which thus motivates optimization by means of *attention*. (Zhang et al., 2021, Chapter 9.7)

2.2.5 Attention

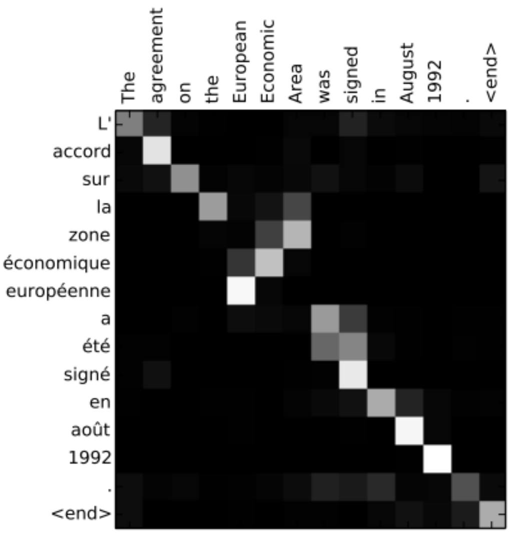

Figure 9: Attention visualized in practical use with machine translation. X-axis and y-axis correspond to the words in the source sentence (English) and generated translation (French), respectively. Pixels indicate the focus of attention in grayscale. (Taken from Bahdanau et al. (2014, p. 6))

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 9 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 version of attention is self-attention. Self-attention has been proposed in several papers, where it is sometimes also 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 "understanding" 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.

2.2.6 Transformer Architecture

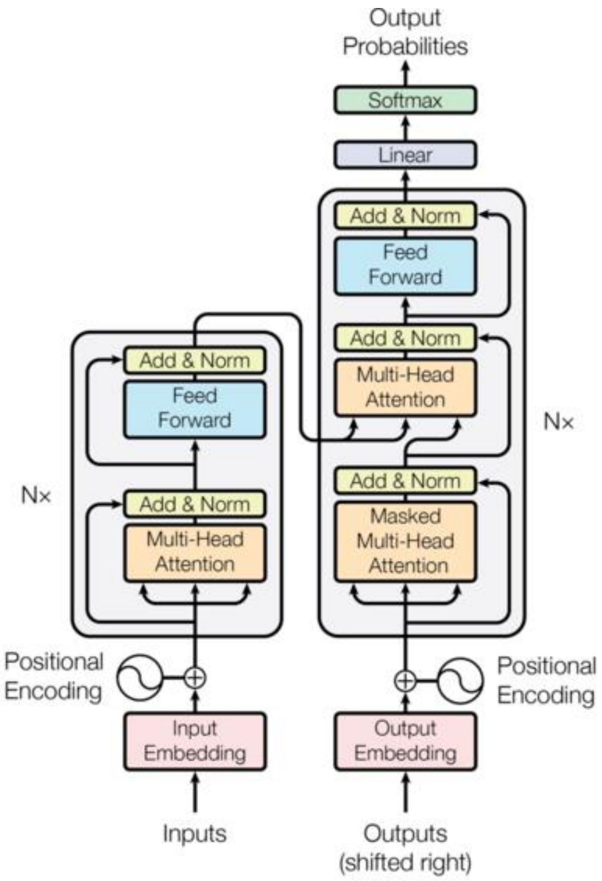

Figure 10: Transformer architecture. (Taken from Vaswani et al. (2017, p. 3))

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 (Vaswani et al., 2017).

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 10, 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 each decoder accepts the output of the last encoder of the encoder stack and uses 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 10, 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, first 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 a normalized 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:

$$Attention(Q,K,V) = softmax(\frac{Q \times K^T}{\sqrt{d_k}})V$$

where Q, K and V denote the query, key, and value matrix, respectively, and d_k denotes the key dimension. The general flow of information is visualized again in Figure 11. (Alammar, 2018a; Rush, 2018; Vaswani et al., 2017)

Multi-head self-attention

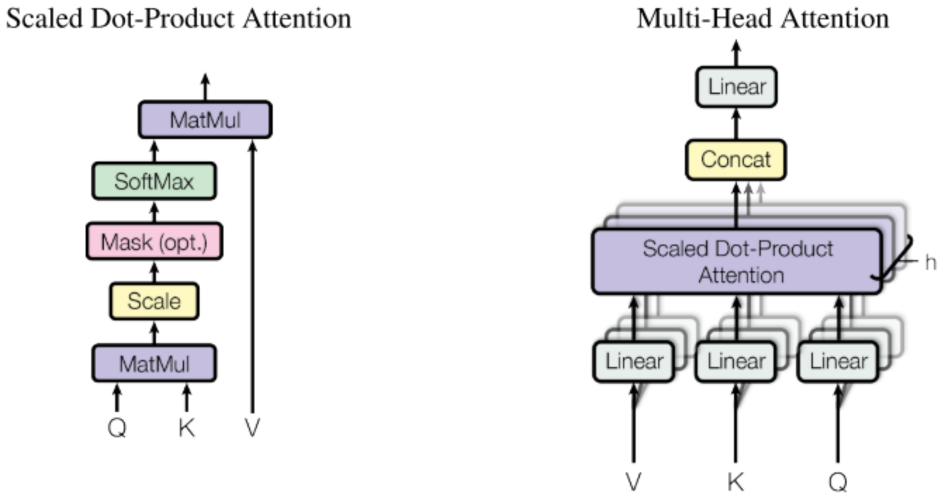

Figure 11: Comparison of scaled dot-product attention (left) and multi-head scaled dot-product attention (right). (Taken from Vaswani et al. (2017, p. 4))

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, and 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 11. (Alammar, 2018a; Rush, 2018; Vaswani et al., 2017)

2.2.7 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 or 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)

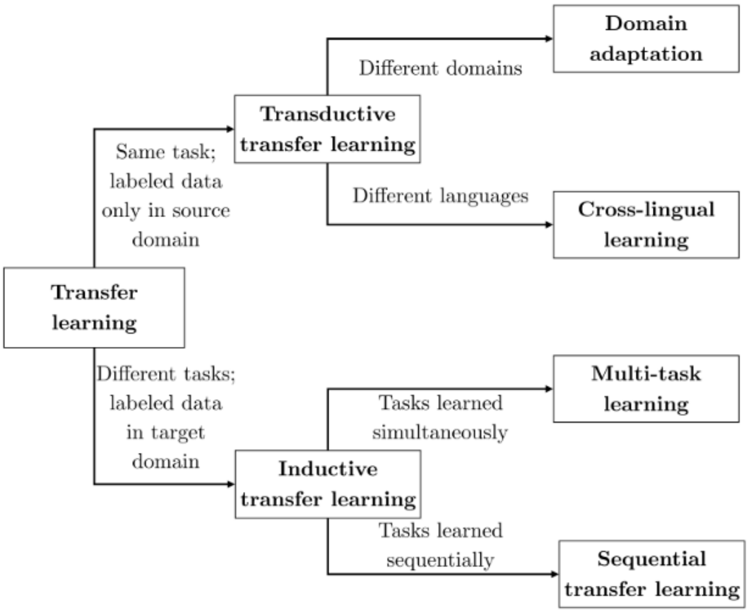
Transfer learning
Same task; labeled data only in source domain
Transductive transfer learning
Different domains
Domain adaptation
Different languages
Cross-lingual learning
Different tasks; labeled data in target domain
Inductive transfer learning
Tasks learned simultaneously
Multi-task learning
Tasks learned sequentially
Sequential transfer learning

Figure 12: Transfer learning taxonomy. (Taken from Pan and Yang (2010) in the abridged version of Ruder (2019))

In the work of Pan and Yang (2010), different settings for such knowledge transfer are identified and categorized. Their proposed taxonomy is adopted in this thesis and can be seen in the simplified version of Ruder (2019) in Figure 12. In the following, the two transfer learning scenarios relevant to this thesis are explained, namely *sequential transfer learning* and *domain adaptation*.

Sequential transfer learning

The scenario of sequential transfer learning occurs when the source task differs from the target task and the goal is "to transfer information from the model trained on the source task to improve performance of the target model" (Ruder, 2019, p. 63). Note that the tasks in this setting are learned sequentially, 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 "adaptation to many target tasks is necessary" (Ruder, 2019, p. 63).

Generally speaking, this kind of transfer learning consists of two phases, i.e., *pretraining* and *adaptation*. 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 generally only has to be done once. (Ruder, 2019, pp. 65–66)

The adaptation 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 is 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

Domain adaptation is the appropriate technique when the distribution of the data seen during training is different from the distribution of the data from the target task. Domain adaptation aims to learn better representations for a specific target domain, unlike sequential transfer learning, which tends to aim to learn more generally useful representations. Furthermore, domain adaptation assumes that both, the source, and the target domain, 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, the researchers 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 2.2.3). 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, which was finetuned on only 100 examples, achieved the same performance as a model that was trained from scratch on 100× the number of examples.

2.3 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 pre-training 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 state-of-the-art performance on eleven different NLP tasks (Devlin et al., 2018) and it 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 this 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 BERT's lack of domain specific knowledge, which forms the basis for the methodology chosen in this work.

2.3.1 Inner Workings

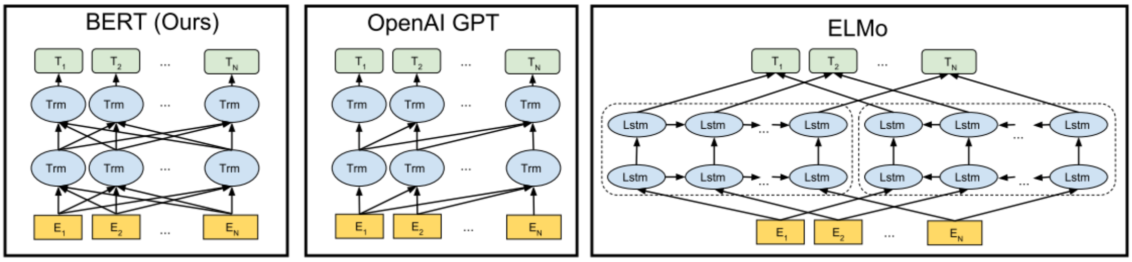

Figure 13: Overview of different contextual pretraining methods. BERT is deeply bidirectional, GPT is unidirectional, and ELMo is shallowly bidirectional. (Taken from Devlin et al. (2018, Appendix A))

BERT's architecture is simply a stack of transformer-encoders (see section 2.2.6). Originally, BERT came in two different versions, BERTBASE and BERTLARGE, which differ in the number of layers (i.e., transformer-encoder blocks) L, their hidden size H, their number of self-attention heads A and their total number of parameters P, with $L = 12$, $H = 768$, $A = 12$, $P = 110M$ for BERTBASE and $L = 24$, $H = 1024$, $A = 16$, $P = 340M$ for BERTLARGE. What makes BERT so powerful in comparison to previous models is that it is "the first deeply bidirectional, unsupervised language representation" (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 Figure 13 and section 2.2.1). In order to learn these language 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, Section 3.1)

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 the transfer learning section above.

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, Section 3.1)

It is to mention that research has shown that NSP is not mandatory as it does not improve downstream task performance of BERT (Lample & Conneau, 2019; Liu et al., 2019), and hence researchers often only use the MLM task to (further) pretrain their own BERT models (Caselli et al., 2020; Martin et al., 2020; Pellegrini et al., 2021).

Input and Output Representations

BERT does not input the embeddings of whole words but rather uses WordPiece embeddings (Wu et al., 2016) 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., $embedding \rightarrow em + \#\#bed + \#\#ding$. Note that all subtokens 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. *[CLS]* is added, so that the final hidden state that corresponds to it can be used "as the aggregate sequence representation for classification tasks" (Devlin et al., 2018, Section 3). *[SEP]* is added to differentiate sentences within the input sequence, which is necessary for the NSP pretraining task, but also for downstream tasks such as question answering. The final representation of a token results from the sum of its token embedding, segment embedding (encodes information about what sentence the token is in) and positional embedding (encodes information about the position of the token in the input sequence). (Devlin et al., 2018)

2.3.2 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. Chen et al. (2019) applied BERT for intent classification and slot filling and achieved significant performance improvements for those tasks on several benchmark datasets compared to traditional attention-based RNNs. Vakulenko et al. (2021) propose a new conversational question answering architecture which achieves new state of the art performance on the TREC Cast 2019 passage retrieval dataset. They used BERT in their architecture for passage re-ranking and answer span extraction. Answer span extraction is also the task that BERT is applied for in the BERTserini chatbot proposed by Yang, W. et al. (2019). The BERTserini architecture first retrieves relevant articles from a Wikipedia corpus given a specific query, and a BERT model that was finetuned on a big question answering corpus then extracts the appropriate text span that contains the answer. Voskarides et al. (2020) applied BERT for query resolution, which is the task of enriching the current turn query with context from the conversational history. In their paper, they formulate it as a binary task where BERT predicts for each term of the conversational history, whether it should be added to the current turn or not. Other tasks in conversational AI for which BERT was used include dialogue breakdown detection (Sugiyama, 2021), response selection (Han et al., 2021; Whang et al., 2020), dialogue state tracking (Chao & Lane, 2019) or topic prediction in recommender systems (Zhou et al., 2020). In terms of the cooking domain, Schwabl (2021) and also Frummet et al. (2021) showed that BERT-based models are beneficial when it comes to classifying information needs that occur during cooking, which was done using the Cookversational Search dataset (see section 2.4.2). Schwabl (2021) furthermore claims that a domain specific BERT model for the cooking domain would probably yield even better classification results.

2.3.3 BERT for Specific Domains

Since BERT was only pretrained on plain text data from the general domain, its performance is limited due to the lack of domain specific knowledge (Gururangan et al., 2020; Rietzler et al., 2019). Adapting BERT for a specific domain is thus a common approach and there is a lot of research showing that this generally leads to performance improvements on the downstream tasks of the adapted domain (see the cited research below), and it is even suggested by the Google Research team (Google Research, 2018). The two main methods that have been established in literature for obtaining a domain specific BERT model are *pretraining on domain specific data from scratch* and *additional pretraining on domain specific data*, which is sometimes also referred to as *post-pretraining* (Liu et al., 2021).

In case of pretraining from scratch, a large domain specific corpus is used instead of the general-domain Wikipedia and book data to train the BERT model from scratch. The most popular example for this approach is SciBERT (Beltagy et al., 2019), a BERT model for the scientific domain which was pretrained on a huge number (1.14M) of scientific articles. The researchers of SciBERT additionally used a custom WordPiece vocabulary that was specifically created for the science domain, which lead to an average increase of 0.6 F1 score. SciBERT achieves state-of-the-art performance in several tasks in the fields of biomedicine, computer science and multi-domain science, and thus shows the effectiveness of the pretraining from scratch approach. However, the downside of this is the huge amount of domain specific data required, which is mostly not available for other domains, as well as the high computational resources that are necessary to perform the training of these huge models in a reasonable time. To put that in perspective, Tim Dettmers, a famous researcher in the field of AI, estimates that "[f]or a standard 4 GPU desktop with RTX 2080 Ti […], one can expect to replicate BERT large in 68 days and BERT base in 34 days" (Dettmers, 2018, Conclusion).

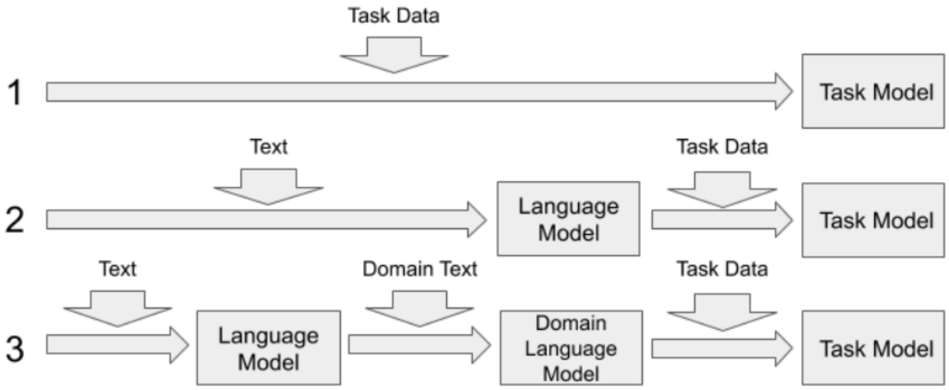

Figure 14: Comparison of traditional machine learning training scheme (1), traditional BERT training scheme (2), and DAPT (3). (Konle & Jannidis, 2020, p. 249)

A good alternative is additional pretraining on domain specific data, either via domain adaptive pretraining (DAPT), task-adaptive pretraining (TAPT), or a combination of both, and is proposed in several research papers as it generally increases task performance of target domain tasks (Gururangan et al., 2020; Rietzler et al., 2019). In case of TAPT, a pretrained model is used as a starting point and its learned language representations are then adapted to the target domain through further pretraining directly on task relevant data, namely the unlabelled task corpus. This is helpful in that the data of the target task sometimes belongs to a very specific area within a more general but specific domain. TAPT has been proven beneficial in terms of downstream task performance in several studies (e.g., Gururangan et al., 2020; Konle & Jannidis, 2020; Lee et al., 2021; Sun et al., 2019). However, TAPT has the disadvantage that the model is adjusted to this very specific target task data and thus might not generalize well on the target domain itself.

DAPT is similar to TAPT, except that the additional pretraining data does not originate from the target task corpus, but from a domain specific corpus in general. A comparison of DAPT to traditional training schemes is given in Figure 14. Just like TAPT, DAPT was shown to be beneficial for downstream tasks of the target domain (e.g., Araci, 2019; Caselli et al., 2020; Gururangan et al., 2020; Konle & Jannidis, 2020; Lee et al., 2020).

Moreover, Gururangan et al. (2020) show consistent improvements in their model's performance when the DAPT domain is more distant from the source domain, indicating that there is a higher potential for DAPT the bigger the data shift between the source and target domain. Regarding the amount of data used for DAPT, this varies a lot in literature, ranging from around 1M words (Sung et al., 2019) up to 13.5B words (Lee et al., 2020). As is generally the case in machine learning, larger amounts of data also tend to lead to better downstream task performance with DAPT (Lee et al., 2020; Liu et al., 2019), and even though it may not always be the case, it is still considered the best option when the target task data is scarce (Zhu et al., 2021). Usually, DAPT is more expensive than TAPT due to more training data, but still less expensive than pretraining from scratch, which makes it the most commonly used approach for creating a domain specific model.

The mentioned techniques have led to the publication of a wide variety of domain-specific BERT models, including BioBERT (Lee et al., 2020) for the biomedical domain, FinBERT (Araci, 2019) for the financial domain, HateBERT (Caselli et al., 2020) for hate speech, and ClinicalBERT (Alsentzer et al., 2019) for the clinical domain. Pellegrini et al. (2021) proposed FoodBERT, a BERT model for the food domain. The researchers of FoodBERT followed the DAPT approach and used recipe instructions from the Recipe1M+ dataset (Marin et al., 2019) as their additional pretraining data. Furthermore, they extended the default BERT vocabulary with domain specific vocabularies as this can have a positive effect on model performance (Tai et al., 2020), similar to the above-mentioned complete rewrite of the SciBERT vocabulary. Pellegrini et al. (2021) merely applied FoodBERT for the specific task of context-free food substitute recommendation, where it performed significantly better than the default BERT model. Further cooking specific BERT models could not be found in the literature even after a thorough search.

2.4 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 datasets that are somehow relevant for conversational AI and thus could be used for training and evaluating CookBERT are considered. Available datasets that meet these criteria and are therefore utilized in this thesis are presented in the next sections.

2.4.1 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 (Radford et al., 2019) 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.

2.4.2 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 the Cookversational Search dataset by Frummet et al. (2021).

Cookversational Search is the resulting dataset of the work Frummet et al. (2021), in which the information needs that arise during cooking were examined, and was already mentioned in section 2.1.3. 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 cooking specific 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 researchers' experiments, 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.

2.4.3 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 the DoQA dataset by Campos et al. (2020).

DoQA (Campos et al., 2020) is a dataset for the task of question answering (in the sense of answer span extraction) and contains a total of 10917 question-answer pairs from 2437 dialogues for the three domains cooking, travelling and movies. With 7320 question-answer pairs, the largest proportion is given for the cooking domain, which is advantageous for this thesis. The dialogues were created via Wizard of Oz method with crowdsourcing, where the crowd workers, which were divided into users and experts, had to ask questions about given posts of Frequently Asked Questions (FAQ) websites 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. An excerpt of DoQA is given in Table 2.

2.4.4 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 with the five different tagging schemes used by Stojanov et al. (2021).

FoodBase (Popovski, Seljak, & Eftimov, 2019) is a corpus for annotated food entities, available in a curated and uncurated version. For both versions, food entities were automatically annotated from cooking recipes with FoodIE (Popovski, Kochev et al., 2019), a rule based named entity tagger. They differentiate in that the curated version was manually reviewed by experts afterwards to remove false positives and add false negatives. 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*[1]. It should also be noted that individual food entities were 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 corpus[1], FoodOn (Dooley et al., 2018), OntoFood[2], 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 schemes/tasks for each entity, which were partly taken from FoodOntoMap and partly constructed themselves. According to the researchers, 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 the *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 cho-

[1] https://www.english-corpora.org/hansard/ (Retrieved on March 5, 2022)

[2] https://bioportal.bioontology.org/ontologies/OF (Retrieved on March 22, 2022)

sen 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 tagging schemes to the commonly used IOB (inside, outside, and beginning) tagging format, proposed by Ramshaw and Marcus (1999). Stojanov et al. (2021) made their adapted version of the FoodBase corpus publicly available[3]. Table 3 shows the respective annotations for each of the five tagging schemes for a given sample sentence.

2.5 Summary and Key Differentiators

CAs have now become ubiquitous and can be found in a broad range of contexts (section 2.1.1). The kitchen also provides a fertile context, but research in this regard is rather sparse (see section 2.1.3), even though conversational AI has generally made strong progress due to rapid advances in the area of deep learning (see section 2.2). The deep learning model BERT (see section 2.3) in particular stands out from these recent developments due to its new properties and its generally excellent performance in NLP and thus also in CAs related tasks (see section 2.3.2). As BERT lacks domain specific knowledge, domain adaptation has proven to be beneficial, with DAPT being a good trade-off of between required time/computational resources and improvements in performance (see section 2.3.3).

Based on these advances and discoveries, CookBERT, a domain specific BERT model for the cooking domain, is developed in this thesis. This is done with the DAPT approach, as well as the addition of domain specific vocabulary to BERT's base vocabulary. Compared to the existing FoodBERT model for the cooking domain, CookBERT uses a bigger amount of data for DAPT, as this has proven beneficial in the past (see section

[3] https://github.com/ds4food/FoodNer (Retrieved on March 5, 2022)

2.3.3). The RecipeNLG corpus (see section 2.4.1) is used for this purpose, as it is the biggest dataset for the cooking domain and contains almost double the data that FoodBERT was further pretrained on. Moreover, CookBERT is evaluated for several CA relevant tasks, including information need classification, food entity tagging, and question answering, since the goal of this thesis is to provide a sophisticated BERT model that can be applied for tasks of cooking specific CA. The central research question of this thesis arising from this is **how does cooking domain adaptation affect BERT's performance on CA relevant tasks.**

3 Methodology

This chapter covers the methodology used in order to answer the research question, which is strongly inspired by previous research covered in chapter 2. First, the data that was used for DAPT as well as necessary pre-processing steps are elucidated in section 3.1. To get an impression of the potential that is to be expected by adapting BERT for the cooking domain, a simple domain similarity analysis of the general source domain and the specific cooking domain is presented in section 3.2. Section 3.3 covers the addition of cooking specific vocabulary to BERT's vocabulary. Section 3.4 gives a brief overview of the tools and development environment used for the domain adaptation of BERT. Information on the actual DAPT is covered in section 3.5. Finally, section 3.6 covers the experimental setup for evaluating the created CookBERT model.

3.1 Preparing the Data for Domain Adaptive Pretraining

Compared to the FoodBERT model by Pellegrini et al. (2021), which was further pretrained on Recipe1M+ instructions, more data was used for CookBERT, as this tends to increase the models task performance (see section 2.3.3). The data for DAPT of CookBERT originates from the RecipeNLG dataset (see section 2.4.1), more precisely, only the recipe instructions. By default, the instructions for a recipe in this dataset are divided into single processing steps. However, each recipe can be regarded as a self-contained document and thus all the instructions of a recipe were concatenated. Additionally, lowercasing was applied to all text data because the uncased version of $\text{BERT}_{\text{BASE}}$ model was used as the starting point for CookBERT, as described in section 3.5. Each recipe was then written into a separate line in a text file to ensure that each recipe could be handled as a distinct sequence when performing DAPT, without including the context of another recipe. This results in a total of almost exactly 1GB of pure cooking specific text data.

3.2 Analyzing Domain Similarity

Prior to the actual DAPT, a simple domain analysis of the cooking specific data to be used for creating CookBERT and the original BERTBASE training corpus was carried out. Out of interest, the data that was used for further pretraining FoodBERT was also examined. The approach for the analysis is adopted from Gururangan et al. (2020), which quantifies domain similarity based on the vocabulary overlap of the corpora. As BERT's original pretraining data is not publicly available, a Wikipedia dump (Merity et al., 2016) and randomly sampled books from the Homemade BookCorpus (Kobayashi, 2018) were used to reconstruct a similar corpus. From RecipeNLG and Recipe1M+, the respective recipe instructions were used as corpus data. For each of the three corpora, the vocabulary, consisting of unigrams (after lowercasing and removal of stopwords and punctuation) was then created.

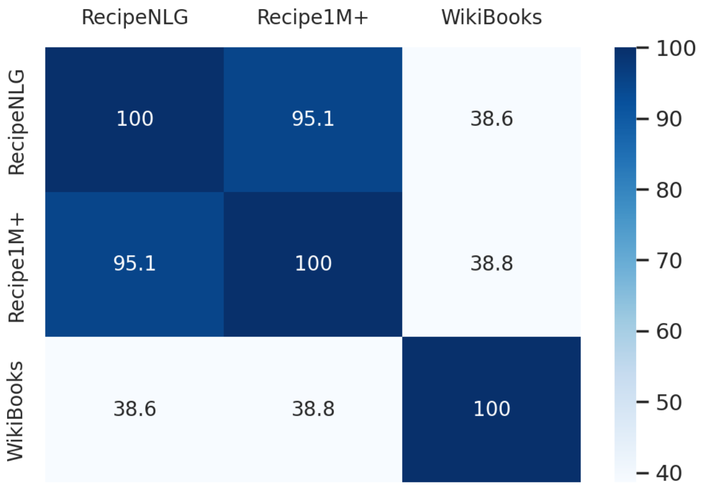

Figure 15: Vocabulary overlap (in %) between the training corpora of BERTBASE (WikiBooks), FoodBERT (Recipe1M+), and CookBERT (RecipeNLG), based on the 10,000 most frequent unigrams.

The vocabulary overlap between the corpora was then determined based on the 10,000 most frequent unigrams of each domain and is illustrated in Figure 15. 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 WikiBooks and the two cooking corpora is quite small, emphasizing the data distribution 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, because "the more dissimilar the domain, the higher the potential for DAPT" (Gururangan et al., 2020, p. 3).

3.3 Domain Vocabulary Insertion

The influence of out-of-vocabulary (OOV) words was proven to have negative influence on the performance of NLP models (Daumé Iii & Jagarlamudi, 2011; Dong & Huang, 2018). Even though BERT deals quite well with OOV words by splitting them up into smaller subtokens (see section 2.3.1), creating a custom vocabulary or adding domain specific words to the existing BERT vocabulary can still improve task performance (see section 2.3.3). As in the work of Pellegrini et al. (2021), this thesis also considers the addition of vocabulary useful, since many common cooking specific words are not included in BERT's vocabulary and are therefore split into "non-representative" subtokens, e.g.:

$$
\begin{gathered}
baguette \rightarrow bag + \#\#uet + \#\#te \\
preheat \rightarrow pre + \#\#hea + \#\#t \\
eggplant \rightarrow egg + \#\#pl + \#\#ant \\
zucchini \rightarrow zu + \#\#chi + \#\#ni \\
caramelized \rightarrow cara + \#\#mel + \#\#ized
\end{gathered}
$$

While Pellegrini et al. (2021) use an external list of cooking ingredients and add every ingredient from this list (multi-word ingredients were combined to one a single token with an underscore) that occurs at least ten times in their pretraining corpus to their FoodBERT vocabulary, this is considered suboptimal in this thesis, as the weights of these new vocabularies are initialized from scratch and such low occurrence frequencies are arguably not sufficient for learning adequate representations. Therefore, to enhance CookBERT's vocabulary, only those words from the RecipeNLG vocabulary created in section 3.2 that occur at least 1000 times in the dataset and were not already included in the BERT$_{\text{BASE\_UNCASED}}$ vocabulary were added so that adequate representations can be learned. Additionally, unlike FoodBERT, the added vocabularies do not only correspond to ingredients, but also to other cooking-related terms such as actions (e.g., "preheat", “overbake”) or kitchen utensils (e.g., “skillet”, “saucepan”, “whisk”). To obtain CookBERT’s final vocabulary, a total of 1229 cooking specific words were added to the BERT$_{\text{BASE\_UNCASED}}$ vocabulary, resulting in a new total size of 31,751 tokens.

3.4 Tools and Environment

Google Colaboratory[4], short Colab, was used as the development environment for DAPT and the experiments in the following sections. It is a free of charge product from Google Research. It is based on Jupyter[5] and allows to manipulate and execute Jupyter notebooks in a browser. The biggest advantage of Colab is the freely provided GPU, which makes it particularly attractive for machine learning and data analysis problems. In addition, Colab enables easy integration of Google Drive[6], which was used in this thesis to store the Jupyter-Notebooks and the relatively memory-intensive CookBERT checkpoints that are saved during DAPT. While the free version already offers extensive functionality, the Google Colab Pro version was used for this thesis, which provides faster GPUs, more RAM, and longer notebook runtimes (up to 24 hours continuously).

Huggingface Transformer Library (Wolf et al., 2019) is arguably the most popular python library when it comes to working with pretrained language representation models. It provides a large selection of different pretrained models and allows you to easily customize them to your own needs. It also provides scripts to finetune the models for specific downstream tasks. The thesis made use of these models provided, as well as the finetuning scripts for the respective evaluation tasks.

3.5 Domain-Adaptive Pretraining

The BERT$_{\text{BASE\_UNCASED}}$ model from Huggingface Library was used as the starting point for CookBERT. Although BERT$_{\text{LARGE}}$ is assumed to yield better results, the base version of BERT was chosen for two reasons. First, computational resources for this thesis are constraint and training BERT$_{\text{LARGE}}$ would be significantly more expensive. Second, the results are more comparable to those of FoodBERT since it is also based on BERT$_{\text{BASE}}$. In terms of the *uncased* version, no distinction between uppercase and lowercased words is

[4] https://colab.research.google.com/ (Retrieved on March 17, 2022)

[5] https://jupyter.org/ (Retrieved on March 17, 2022)

[6] https://www.google.com/drive/ (Retrieved on March 17, 2022)

made by the model. While this results in the name "Bill" and the dollar "bill" being considered the same, it also ensures that "Baguette" and "baguette" are not considered different just because one is placed at the beginning of a sentence and the other one in the middle. The uncased version is used for CookBERT, as the second case is arguably more common in the cooking domain, and due to the self-attention mechanism, BERT should be able to distinguish the first case, nonetheless. Next to DAPT data size and the vocabulary added, this is another differentiator to FoodBERT, which is based on the cased version.

To obtain CookBERT, $\text{BERT}_{\text{BASE\_UNCASED}}$ was trained according to DAPT for three additional epochs on the MLM task on the RecipeNLG instructions prepared in section 3.1, with 5% of the data serving as validation data. The script for the MLM task from Huggingface Library was used for this. Training was performed with a learning rate of 2e-5 as suggested by the Google Research team (Google Research, 2018), a batch size of 16 with two gradient accumulation steps, and a maximum sequence length of 256. In addition, it was ensured that each sequence only contains complete recipe instructions, which corresponds to lines in the prepared RecipeNLG instructions corpus (see section 3.1). Validation was performed every 1000 steps. Checkpoints were also saved at the same intervals, to prevent the training from having to be restarted from scratch if it was aborted. Otherwise, BERT's default training parameters were used.

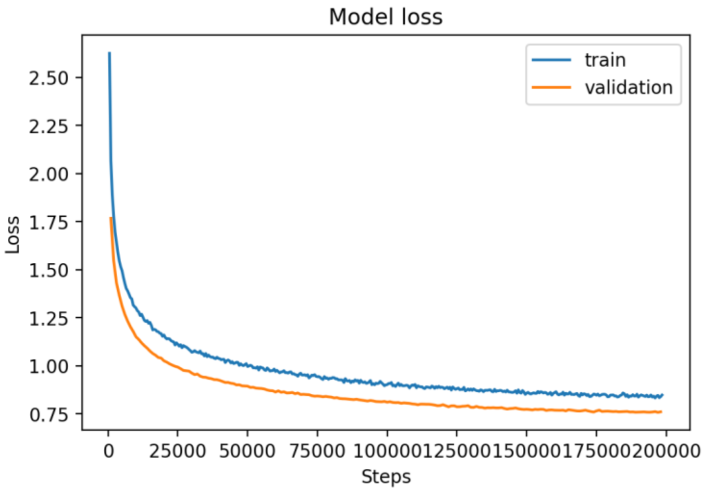

Figure 16: Model loss for DAPT on RecipeNLG instructions. Note that validation loss is lower as dropout layers are only activated in training and not in validation. Furthermore, training loss is calculated during each epoch and validation loss after each epoch.

DAPT was performed on a single NVIDIA Tesla P100 GPU provided by Google Colab Pro and took approximately five complete days to complete. CookBERT's learning curve is shown in Figure 16 and indicates that even though the model is slowly converging after almost 200,000 steps, it would probably still continue to learn if it was trained for additional steps.

Input	**Model**	**Top 5 predictions for [MASK] token (in sorted order)**
"Do I have to [MASK] the apple?"	CookBERT	peel, slice, use, dice, chop
	$\text{BERT}_{\text{BASE\_UNCASED}}$	eat, take, have, touch, get
"[MASK] the water."	CookBERT	boil, heat, add, scald, chill
	$\text{BERT}_{\text{BASE\_UNCASED}}$	in, drink, under, into, on
"Cut the [MASK] into small pieces."	CookBERT	chicken, cheese, fruit, cabbage, sausage
	$\text{BERT}_{\text{BASE\_UNCASED}}$	wood, paper, leaves, meat, bark

Table 4: Comparison of MLM predictions.

As a result of DAPT, CookBERT emerges, which is now more strongly focused on the cooking domain compared to the base model. This can be well illustrated by the examples in Table 4. While both models, CookBERT and $\text{BERT}_{\text{BASE\_UNCASED}}$ make reasonable word predictions, the ones of $\text{BERT}_{\text{BASE\_UNCASED}}$ are more generic and the ones of CookBERT are strongly related to cooking specific vocabulary.

3.6 Experimental Setup

CookBERT is evaluated using three CA relevant tasks, including information need classification, food entity tagging, and question answering. In addition, $\text{BERT}_{\text{BASE\_UNCASED}}$ and FoodBERT were applied for the same tasks in order to be able to compare and rank CookBERT's performance. $\text{BERT}_{\text{BASE\_UNCASED}}$ was chosen, as it is the model used for initializing CookBERT and thus allows to investigate the effect that the domain adaptation methodology used in this thesis has on task performance. FoodBERT was chosen as the other baseline model, since it is also a cooking specific BERT model, but was created with a slightly different methodology compared to CookBERT. This remainder of this chapter covers the setup for performing the experiments.

3.6.1 Information Need Classification

The first evaluation task is about the classification of information needs that arise during cooking with the Cookversational Search dataset, which was presented in section 2.4.2. The focus in this thesis is on the classification of the level one information needs, which includes eleven different types of information needs and is thus a multi-class classification problem. Furthermore, the classification performance is to be evaluated for two different conditions that were proposed by Frummet et al. (2021). In case of the *no context* condition, the underlying information need has to be identified based on a single user utterance, without any additional contextual information. With the *one previous utterance* condition, on the other hand, the preceding user utterance is prepended to the current utterance as additional context information.

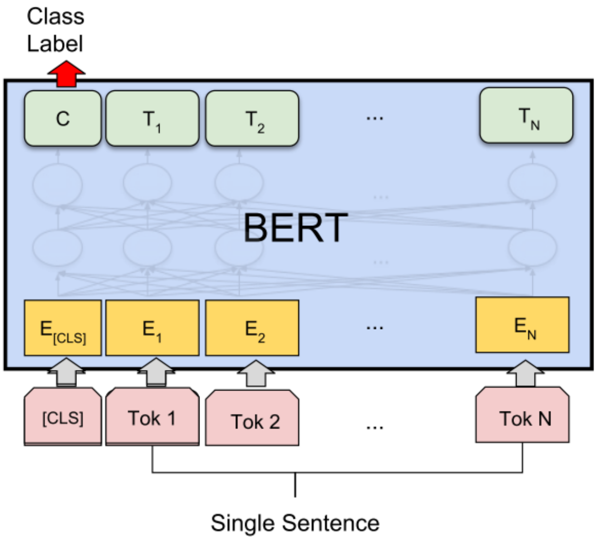

Figure 17: BERT for text classification. Ei denotes the input embedding and Ti the contextual representation of token i. C denotes the contextual representation of the *[CLS]* token. (Taken from Devlin et al. (2018, Appendix B))

In order to apply BERT for multi-class classification, a softmax classifier is added on top of the BERT model, which takes the final hidden state of the *[CLS]* token and predicts the probability of each possible label (see Figure 17; Sun et al., 2019). The label with the highest probability is the one that is assigned to the input sequence. To adjust the three models accordingly, the finetuning script for sequence classification that is provided by the Huggingface Library was used.

All three models were evaluated for both task conditions using the 10-fold cross validation method, with identical folds for each model within a condition. Since classes of the Cookversational Search dataset are highly imbalanced, stratified sampling was applied and the class weights were adjusted accordingly. The finetuning parameters were identical for all models. Finetuning was performed for four epochs with a batch size of 16 with two gradient accumulation steps, and a lower learning rate of 2e-5, as this is suggested by Sun et al. (2019) in order to avoid catastrophic forgetting. To evaluate classification performance, the metrics precision, recall and F-measure, which is defined as the harmonic mean between them, were used. Accuracy was not considered because it is not a reliable metric in this case due to the high imbalance of the dataset. The following hypotheses were made:

H1.1: CookBERT's general performance on the classification of cooking specific information needs from the Cookversational Search dataset is significantly higher than for $\text{BERT}_{\text{BASE\_UNCASED}}$.

H1.2: CookBERT's general performance on the classification of cooking specific information needs from the Cookversational Search dataset is significantly higher than for FoodBERT.

H1.3: CookBERT performs significantly better on the *one previous turn* condition than on the *no context* one.

Additional hypotheses were formulated for each condition *c*:

H1.4: CookBERT performs significantly better than $\text{BERT}_{\text{BASE\_UNCASED}}$ for classifying cooking specific information needs from the Cookversational Search dataset under condition *c*.

H1.5: CookBERT performs significantly better than FoodBERT for classifying cooking specific information needs from the Cookversational Search dataset under condition *c*.

3.6.2 Food Entity Tagging

The second evaluation task is food entity tagging and was carried out using the curated version of the FoodBase corpus, as well as the labels provided by Stojanov et al. (2021) for five different tagging schemes (see section 2.4.4).

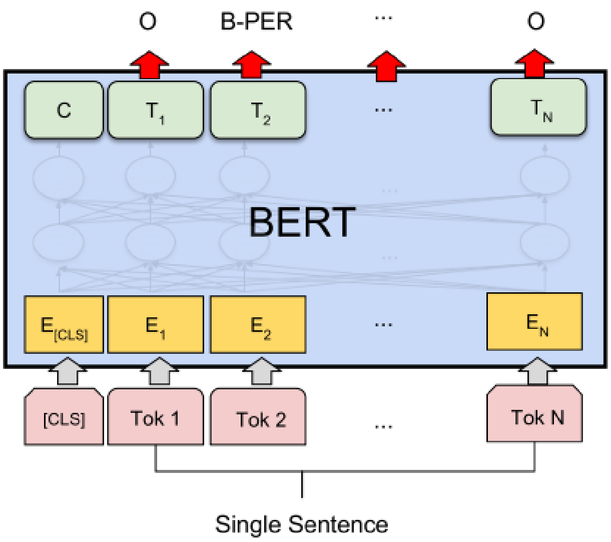

Figure 18: BERT for entity tagging. (Taken from Devlin et al. (2018, Appendix B))

The adaptation of BERT for entity tagging is similar as for sequence classification. The difference is that a token-level softmax classifier is added on top of BERT. This takes the last hidden states for each token of the sequence and then predicts the corresponding tag (see Figure 18; Devlin et al., 2018). To adjust the three models that are to be evaluated accordingly, the finetuning script for token classification provided by the Huggingface Library was used.

All three models were evaluated for the five different tagging tasks using the 10-fold cross validation method, with identical folds for each model within a tagging task. Since Stojanov et al. (2021) showed that many epochs (up to 100, but with linear learning rate reduction) are needed to finetune BERT for their tagging task, a higher number of epochs was chosen in this thesis as well. However, due to computational and time constraints, the models were still only finetuned for 15 epochs, even though additional epochs would probably yield better results. In addition, a learning rate of 2e-5 and a batch size of 16 with two gradient accumulation steps were chosen. Finetuning parameters were identical for all models. As for the classification task, the metrics precision, recall and F-measure were chosen for evaluation. Accuracy was not considered reliable here either since most of the tags are assigned with the outside tag and the accuracy would therefore be disproportionately high for all models and tagging schemes. For each of the five tagging schemes *s*, the following hypotheses were formulated:

> **H2.1**: CookBERT performs significantly better than BERT$_{\text{BASE\_UNCASED}}$ for tagging food entities from the FoodBase corpus given tagging scheme *s*.

H2.2: CookBERT performs significantly better than FoodBERT for tagging food entities from the FoodBase corpus given tagging scheme s.

3.6.3 Question Answering

The last evaluation task is question answering in the sense of answer span extraction. For this, the cooking subset of the DoQA dataset was used (see section 2.4.3). Unanswerable questions were removed, resulting in a total of 5266 question-answer pairs.

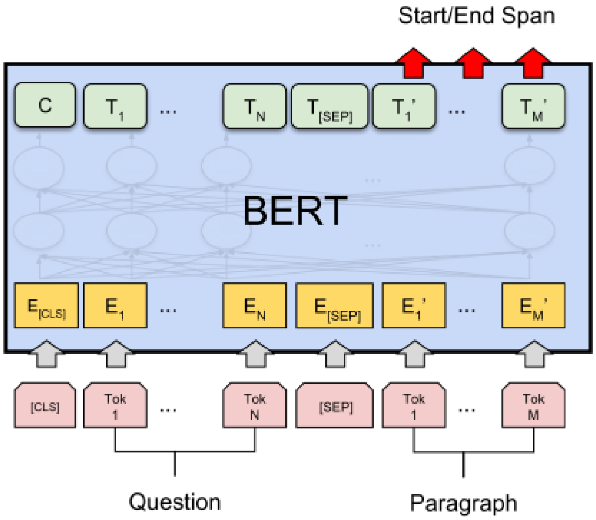

Figure 19: BERT for question answering. (Taken from Devlin et al. (2018, Appendix B))

To apply BERT for question answering, a token-level softmax classifier is added on top of it. The classifier has two different weight vectors, so that the start and the end of the answer span can be predicted (see Figure 19). The softmax classifier takes the final hidden state of each token from the paragraph and creates a probability distribution over all words. The word with the highest probability then corresponds to the start or end of the answer span, respectively. (McCormick, 2020)

Again, the appropriate script from Huggingface Library was used for adapting and finetuning the three BERT models, and the performance was evaluated via 10-fold cross validation. The parameters chosen for finetuning were identical for all models and correspond to the ones proposed in the original BERT paper (Devlin et al., 2018). This means, the models are finetuned for three epochs with a learning rate of 5e-5 and a batch size of 32. Exact match and F-measure are two common evaluation metrics for question answering and are therefore chosen in this thesis. The aptly named exact match metric describes the percentage of samples for which the extracted answer exactly matches the ground truth. As soon as the prediction is off by a single character, the sample is scored with zero. In return, the F-measure is more lenient. It is defined as the harmonic mean between precision and recall, with precision being the proportion of the number of shared words to the total number of words in the prediction and recall being the proportion of the number of shared words to the total number of words in the ground truth. The hypothesis regarding this task are as follows:

H3.1: CookBERT performs significantly better than BERT$_{\text{BASE\_UNCASED}}$ for question answering on the DoQA cooking dataset.

H3.2: CookBERT performs significantly better than FoodBERT for question answering on the DoQA cooking dataset.

4 Evaluation

4.1 Information Need Classification

Model	**Condition**	**Precision**	**Recall**	**F-Measure**	**95%-CI**
BERT$_{\text{BASE\_UNCASED}}$	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 5: Information need classification experiment results after 10-fold cross validation grouped by model and condition.

The results for the precision, recall and F-measure for the information need classification task are listed in Table 5. Note that results are macro-averaged over all classes. The results show that CookBERT performs best for both conditions, followed by BERT$_{\text{BASE\_UNCASED}}$, which in turn 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 performances for the *no context* condition, no significant differences were found (F = 1.58, p = .21). In case of the *one previous turn* condition, CookBERT performs significantly better than FoodBERT (p

< .001), but the findings between CookBERT and BERT$_{\text{BASE\_UNCASED}}$ (p = .062) and BERT$_{\text{BASE\_UNCASED}}$ and FoodBERT (p = .18) 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$_{\text{BASE\_UNCASED}}$ (p = .034) and FoodBERT (p < .001). No significant difference was found between FoodBERT and BERT$_{\text{BASE\_UNCASED}}$ (p = .27). The overall performance over both conditions is 52.38% for CookBERT, 45.77% for BERT$_{\text{BASE\_UNCASED}}$ 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 *one previous turn* condition (M = 54.05%) than for the *no context* one (M = 50.72%), these findings are not significant (F = 0.82, p = .37).

4.2 Food Entity Tagging

Model	Tagging-Task	Precision	Recall	F-Measure	95%-CI
BERT$_{\text{BASE\_UNCASED}}$	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 6: Food entity tagging experiment results after 10-fold cross validation grouped by model and tagging task.

The results for the precision, recall and F-measure for the food entity tagging task are listed in Table 6. It shows the order of the best-performing models is the same across all tagging tasks: CookBERT achieves superior performance on all tasks, followed by BERT$_{\text{BASE\_UNCASED}}$, and FoodBERT consistently performs 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*, *FoodOn*, *Hansard-parent* and *SNOMED CT*, with p < .001 in each case. In comparison to BERT$_{\text{BASE\_UNCASED}}$, CookBERT performs significantly better on the three tasks *Food-classification* (p < .001), *FoodOn* (p = .002) and *SNOMED CT* (p = .002), but this is not the case for the *Hansard-parent* task (p = .067). Similar to CookBERT, BERT$_{\text{BASE\_UNCASED}}$ performs significantly better than FoodBERT on the four tasks *Food-classification*, *FoodOn*, *Hansard-parent* and *SNOMED CT*, where p < .001 applies in each case. *Hansard-closest* is the only task where no significant performance differences between any of the three models could be found (F = 1.31, p = .27).

The overall performance for food entity tagging on the FoodBase corpus, given as the macro-averaged F-measure over all tagging tasks, is 80.33% for CookBERT, 77.65% for BERT$_{\text{BASE\_UNCASED}}$, and 67.87% for FoodBERT.

4.3 Question Answering

Model	**Exact match**	**F-Measure**	**95%-CI**
BERT$_{\text{BASE\_UNCASED}}$	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 7: 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 7. Between the three models, BERT$_{\text{BASE\_UNCASED}}$ 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 BERT$_{\text{BASE\_UNCASED}}$ performance (p = .072). However, both CookBERT and BERT$_{\text{BASE\_UNCASED}}$ perform significantly better than FoodBERT, with p < .001 in both cases.

5 Discussion

Before discussing the meaning and relevance of the results and answering the research question, the hypotheses formulated in section 3.6 are first taken up again. When looking at the performance of CookBERT for the classification of cooking specific information needs, a clear benefit of the domain adaptation can be seen, since CookBERT achieves better results than the initial BERT$_{\text{BASE\_UNCASED}}$ for both conditions. Although these performance improvements are not significant when viewed individually and thus **H1.4** is not supported for either of the task conditions, the general performance improvement in the classification task is significant according to expectations and **H1.1** can therefore be accepted. In comparison to the also domain specific FoodBERT, CookBERT's overall performance for this task is significantly better as well, which means **H1.2** can be accepted. Additionally, **H1.5** can be accepted for the *one previous utterance* condition but is in return not supported for the *no context* one. While **H1.3** is not supported, it is still assumed that CookBERT's performance in classifying certain information needs benefits from additional context information from previous utterances, which was also previously shown by Frummet et al. (2021). However, as it is not part of this thesis, no further investigations towards this were attempted. Cooking domain adaptation is also beneficial for the food entity tagging task. CookBERT outperforms BERT$_{\text{BASE\_UNCASED}}$ in all of the five tagging tasks. For the three tagging schemes *Food-classification*, *FoodOn* and *SNOMED CT*, these findings are significant and **H2.1** can be accepted, but it is not supported for the tagging schemes *Hansard-parent* and *Hansard-closest*. CookBERT also outperforms FoodBERT in all tasks and **H2.2** can be accepted for the *Food-classification*, *FoodOn*, *SNOMED CT* as well as the *Hansard-parent* tagging scheme. Only the finding for *Hansard-closest* is not significant, and thus the hypothesis for this tagging scheme is not supported. Question answering is the only task where the domain adaptation did not yield performance improvements, and CookBERT even performed worse than its base model, but this finding is not significant. This means **H3.1** is not supported. In return, **H3.2** is supported, as CookBERT performs significantly better than FoodBERT, nonetheless.

Based on these findings, the underlying research question, **how cooking domain adaptation affects BERT's performance on CA relevant tasks,** can be answered as follows. Domain adaptation as carried out in this thesis has proven to be a viable and efficient approach to bring BERT closer to the cooking domain with relatively little time and computational effort. However, the impact on task performance is task dependent and does not always automatically result in a significant increase. For the classification of cooking specific information needs, as well as for most food entity tagging tasks, the adaptation led to significant performance boosts. Based on the findings of Tai et al. (2020), Beltagy et al. (2019), and Gururangan et al. (2020), it is assumed that both further pretraining on domain specific data and the vocabulary enhancement led to better language representations learned by CookBERT, which thus responsible for these improvements. However, the actual impact of the individual aspects of the domain adaptation method used is not examined in this work. In case of the question answering task, CookBERT was not able to achieve better results. This could be due to the unnaturalness of the DAPT data, which may have been adopted during DAPT, thus harming CookBERT's language representations. This aspect is explained in more detail in the following limitations section. Note that the generally low performance of the three evaluated models for the question answering task can be explained by the high difficulty of the dataset, as "questions in DoQA require long and complex answers" (Campos et al., 2020, Section 4). All in all, the findings regarding the effects of domain adaptation align with those in the literature: **domain adaptation can significantly improve task performance, but this does not always have to be the case** (Zhu et al., 2021). Significant performance deterioration could not be determined on any of the evaluated tasks in this thesis. Therefore, with CookBERT, a BERT model is presented, which should definitely be considered for conversational agents in the context of cooking.

Compared to the only other known cooking specific BERT based model, FoodBERT, CookBERT performs better in every experimental setting tested, with most findings being significant. Although the actual reasons for this are not known, it is assumed that CookBERT learns better language representations for the cooking domain due to the combination of the individual methodologically decisions taken, including the bigger amount of data for DAPT, the different approach for the vocabulary enhancement, and the choice of BERT$_{\text{BASE\_UNCASED}}$ as the starting point instead of the cased version. What is surprising is that BERT$_{\text{BASE\_UNCASED}}$ performs significantly better than FoodBERT in question answering as well as in four out of five food entity tagging tasks. It could be that uncased models perform better for these evaluation tasks in general, but further research needs to be done to substantiate this.

6 Limitations

This section reflects on the limitations of this thesis. First of all, the data used for DAPT 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, recipe instructions are not very natural, since they are mostly formulated as imperative, the syntax is often incorrect due to omitted articles and pronouns, and words are often replaced by abbreviations. The sentence "add egg and 3 tbls butter to batter" provides a good example for this. This unnaturalness could have a negative impact on CookBERT's linguistic competence as it might be adopted during DAPT. In the previous discussion section, this is also listed as a possible reason why CookBERT performs worse in question answering than the $\text{BERT}_{\text{BASE\_UNCASED}}$ version. This limitation is difficult to remedy because only few datasets exist that provide natural cooking data, and most of them are rather small-scale. So, the best approach would be to collect a huge amount of natural, cooking specific data yourself, for example from subtitles of cooking shows (which was also suggested from Schwabl (2021, pp. 82–83), from cooking podcast transcripts, or from general cookbooks that do not only contain plain 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.

Finally, computational and time related constraints are the reason that only the BERT base model was adapted for the cooking domain. However, it would be interesting to see if the approach that was used in this thesis also yields promising results for other language representation models. It is assumed that the adaptation of newer and more optimized models to the cooking domain, such as RoBERTa (Liu et al., 2019) or XLNet (Yang, Z. et al., 2019), leads to further performance improvements.

7 Conclusion

This thesis introduces CookBERT, a domain specific BERT model for the cooking domain. The model was created via domain adaptive pretraining on recipe instructions of the large RecipeNLG corpus, as well as enhancing BERT's vocabulary with cooking specific words. Since the goal behind CookBERT was the creation of a sophisticated BERT model that can be applied for different kitchen CA relevant tasks, the research question "how does cooking domain adaptation of BERT affect its performance on CA relevant tasks" was set up. To answer this question, CookBERT was evaluated on several such tasks, including the classification of cooking information needs, the tagging of food entities, and question answering. The findings indicate that the effects of domain adaptation as done in this thesis are task dependent, meaning that it can significantly increase task performance, but this does not always have to be the case. Significant improvements in performance could be determined for the information need classification and the food entity tagging task, but not for question answering. In addition, CookBERT performs significantly better in almost all experimental settings compared to the only other known cooking domain specific BERT model, that is FoodBERT. Furthermore, limitations of this thesis were highlighted. Future work can take on these to try and train an even more sophisticated cooking specific model. Future work can also use CookBERT and apply it to their own CA relevant tasks that need to be solved. For example, in the case of the Cookversational Search dataset by Frummet et al. (2021), CookBERT could be applied for query rewriting to enrich a user utterance with information from previous ones, which the authors claim is beneficial for classifying information needs.

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/. Retrieved on 03/27/2022.

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/. Retrieved on 03/27/2022.

Alsentzer, E., Murphy, J. R., Boag, W., Weng, W.-H., Di Jin, Naumann, T., & McDermott, M. B. A. (2019, April 6). *Publicly Available Clinical BERT Embeddings*. http://arxiv.org/pdf/1904.03323v3

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.

Araci, D. (2019, August 27). *FinBERT: Financial Sentiment Analysis with Pre-trained Language Models*. http://arxiv.org/pdf/1908.10063v1

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

Beltagy, I., Lo Kyle, & Cohan, A. (2019). SciBERT: A Pretrained Language Model for Scientific Text. http://arxiv.org/pdf/1903.10676v3

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

Caselli, T., Basile, V., Mitrović, J., & Granitzer, M. (2020, October 23). *HateBERT: Retraining BERT for Abusive Language Detection in English*. http://arxiv.org/pdf/2010.12472v2

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

Daumé Iii, H., & Jagarlamudi, J. (2011, June). Domain adaptation for machien translation by mining unseen words. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 407–412.

Dettmers, T. (2018, October 17). *TPUs vs GPUs for Transformers (BERT).* https://timdettmers.com/2018/10/17/tpus-vs-gpus-for-transformers-bert/#:~:text=Conclusion,11%20days%20using%208%2Dbit. Retrieved on 03/27/2022.

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. Retrieved on 03/27/2022.

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

Dong, J., & Huang, J. (2018). Enhance word representation for out-of-vocabulary on ubunt dialogue corpus. *ArXiv Preprint ArXiv:1802.02614*.

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

Google Research. (2018). *BERT - Github*. https://github.com/google-research/bert. Retrieved on 03/27/2022.

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

Han, J., Hong, T., Kim, B., Ko, Y., & Seo, J. (2021, June). Fine-grained Post-training for Improving Retrieval-based Dialogue Systems. *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1549–1558. https://doi.org/10.18653/v1/2021.naacl-main.122

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/. Retrieved on 03/27/2022.

IBM Cloud Education. (2020a, August 17). *Neural Networks*. https://www.ibm.com/cloud/learn/neural-networks. Retrieved on 03/27/2022.

IBM Cloud Education. (2020b, September 14). *Recurrent Neural Networks*. https://www.ibm.com/cloud/learn/recurrent-neural-networks. Retrieved on 03/27/2022.

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/

Khatri, C., Hedayatnia, B., Venkatesh, A., Nunn, J., Pan, Y., Liu, Q., Song, H., Gottardi, A., Kwatra, S., Pancholi, S., Cheng, M., Chen, Q., Stubel, L., Gopalakrishnan, K., Bland, K., Gabriel, R., Mandal, A., Hakkani-Tur, D., Hwang, G., . . . Prasad, R. (2018, December 27). *Advancing the State of the Art in Open Domain Dialog Systems through the Alexa Prize*. http://arxiv.org/pdf/1812.10757v1

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 2020: Workshop on Computational Humanities Research*.

Lample, G., & Conneau, A. (2019, January 22). *Cross-lingual Language Model Pretraining*. http://arxiv.org/pdf/1901.07291v1

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

Lee, J., Kim, J., & Kang, P. (2021, July 22). *Back-Translated Task Adaptive Pretraining: Improving Accuracy and Robustness on Text Classification*. http://arxiv.org/pdf/2107.10474v1

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

Liu, Y., Ott, M., Goyal, N., Du Jingfei, Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019, July 26). *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. http://arxiv.org/pdf/1907.11692v1

Liu, Z., Lin, W., Shi, Y., & Zhao, J. (2021). A Robustly Optimized BERT Pre-training Approach with Post-training. In S. Li, M. Sun, Y. Liu, H. Wu, L. Kang, W. Che, S. He, & G. Rao (Eds.), *Lecture Notes in Computer Science. Chinese Computational Linguistics* (Vol. 12869, pp. 471–484). Springer International Publishing. https://doi.org/10.1007/978-3-030-84186-7_31

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

Martin, L., Muller, B., Suárez, P. J. O., Dupont, Y., Romary, L., La Clergerie, É. V. d., Seddah, D., & Sagot, B. (2020). CamemBERT: a Tasty French Language Model, 7203–7219. https://doi.org/10.18653/v1/2020.acl-main.645

McCormick, C. (2020, March 10). *Question Answering with a Fine-Tuned BERT*. https://mccormickml.com/2020/03/10/question-answering-with-a-fine-tuned-BERT/. Retrieved on 03/27/2022.

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/. Retrieved on 03/27/2022.

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.*

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI Blog*(1(8)), 9.

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

Rietzler, A., Stabinger, S., Opitz, P., & Engl, S. (2019, August 30). *Adapt or Get Left Behind: Domain Adaptation through BERT Language Model Finetuning for Aspect-Target Sentiment Classification*. http://arxiv.org/pdf/1908.11860v2

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. Retrieved on 03/27/2022.

Schwabl, P. (2021). *Classifying user information needs in cooking dialogues – an empirical performance evaluation of transformer networks* [Masterarbeit, Universität Regensburg]. DataCite.

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

Sugiyama, H. (2021). Dialogue breakdown detection using BERT with traditional dialogue features. *Increasing Naturalness and Flexibility in Spoken Dialogue Interaction*, 419–427.

Sun, C., Qiu, X., Xu, Y., & Huang, X. (2019, May 14). *How to Fine-Tune BERT for Text Classification?* http://arxiv.org/pdf/1905.05583v3

Sung, C., Dhamecha, T., Saha, S., Ma, T., Reddy, V., & Arora, R. (2019, November). Pre-Training BERT on Domain Resources for Short Answer Grading. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 6071–6075. https://doi.org/10.18653/v1/D19-1628

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

Tai, W., Kung, H. T., Dong, X., Comiter, M., & Kuo, C.-F. (2020, November). exBERT: Extending Pre-trained Models with Domain-specific Vocabulary Under Constrained Training Resources. *Findings of the Association for Computational Linguistics: EMNLP 2020*, 1433–1439. https://doi.org/10.18653/v1/2020.findings-emnlp.129

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.

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, *78*(10), 1550–1560.

Whang, T., Lee, D., Lee, C., Yang, K., Oh, D., & Lim, H. (2020, October 25). An Effective Domain Adaptive Post-Training Method for BERT in Response Selection. In *Interspeech 2020* (pp. 1585–1589). ISCA. https://doi.org/10.21437/Interspeech.2020-2153

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

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., Platen, P. v., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., . . . Rush, A. M. (2019, October 9). *HuggingFace's Transformers: State-of-the-art Natural Language Processing*. http://arxiv.org/pdf/1910.03771v5

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

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., & Le V, Q. (2019, June 19). *XLNet: Generalized Autoregressive Pretraining for Language Understanding*. http://arxiv.org/pdf/1906.08237v2

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

Zhou, K., Zhou, Y., Zhao, W. X., Wang, X., & Wen, J.-R. (2020, October 8). *Towards Topic-Guided Conversational Recommender System*. http://arxiv.org/pdf/2010.04125v2

Zhu, Q., Gu, Y., Luo, L., Li, B., Li, C., Peng, W., Huang, M., & Zhu, X. (2021). When does Further Pre-training MLM Help? An Empirical Study on Task-Oriented Dialog Pre-training. *Proceedings of the Second Workshop on Insights from Negative Results in NLP*, 54–61. https://aclanthology.org/2021.insights-1.9

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.