UNIVERSITY
OF OSLO

Master's thesis

Language Models and World Knowledge

Injecting structured information using masked language modeling and adapters

Sondre Wold

Informatics: Language Technology
60 study points

Department of Informatics
Faculty of Mathematics and Natural Sciences

Abstract

Combining structured information with language models is a standing problem in NLP. Building on previous work, we study how lightweight neural networks, known as adapters, can be used to inject information from a knowledge graph into two popular pre-trained language models based on the transformer architecture. The adapters are trained using the masked language modeling objective over extracted triples from ConceptNet, a knowledge graph that captures a range of world knowledge and commonsense concepts and relations. Experiments on three popular NLP benchmarks believed to require world knowledge and commonsense reasoning abilities show that the adapter injection does not increase performance on these tasks. However, probing experiments indicate that the injected models are better at recovering factual information seen during training, and that this can be achieved by introducing a small amount of additional parameters to the overall model. Ablation studies show that the injected knowledge is distributed equally among the layers in the underlying model. Furthermore, using the AdapterFusion framework, we propose and perform initial testing of a two-step learning algorithm that partitions ConceptNet by predicate type and trains a set of disjoint adapters that are later combined using an attention mechanism. For reproducibility, we present a reproduction of the most related previous work and release our code.[1]

[1] https://github.com/SondreWold/Thesis_code

Contents

List of Figures

List of Tables

Chapter 1

Introduction

It should be noncontroversial to state that understanding of language relies on access to experiences of the world. We use an array of mundane, multi-modal and factual information in order to navigate the ambiguities that arise in everyday conversation. Inference based on such knowledge is non-trivial, even for humans. It is as difficult to make the correct inference as it is to not make one at all. When faced with pieces of language, either through speech or written text, we can't help ourselves: We make assumptions that go far beyond what is explicitly stated, some more unconscious than others. Taking any statement at face value, without eliciting any connotations, seems impossible, almost inhuman.

As modern pieces of language technology primarily consume unstructured text using machine learning, that is pure linguistic form (Bender and Koller, 2020), how can a machine, which has no native assumptive mechanism, understand or make inferences based on language in a similar fashion? For example, how can it infer that only sentence **b** follows from sentence **a** in the example below, and not **c**? There are no linguistic queues that reveal that treks in Jotunheimen National Park are a subset of the treks in all of Norway but not in Sweden.

Recent advancements in Natural Language Processing (NLP) nevertheless increasingly succeed at making these types of inferences. The explanation? Given a large enough corpus, Jotunheimen will appear more frequently in close proximity with Norway than Sweden. This makes it less probable that statements mentioning Jotunheimen is related to statements mentioning Sweden, compared to any statement mentioning Norway. What it does not, however, is to make it true.

(1.1) a) Jotunheimen is a great place for trekking.
b) Norway has places which are great for trekking.
c) Sweden has places which are great for trekking.

This poses problems for deployment of pieces of language technology into domains where the truth of what is produced matter. If the training material for a language model include frequent mentions of Swedish tourists to Jotunheimen, the example above quickly becomes more challenging. Current models do not have any native mechanism for encoding and enforcing the knowledge that seems necessary, namely that Jotunheimen **is in** Norway. At the same time, there exist multiple sources that attempt to capture and structure this type of factual knowledge. Such sources range from domain specific knowledge graphs for medical information (Shi et al., 2017), more general ontologies like Yago or ConceptNet (Speer, Chin and Havasi, 2017; Suchanek, Kasneci and Weikum, 2007) to lexico-semantic networks like WordNet (Miller, 1995).

This work investigates how language models based on the transformer architecture (Vaswani et al., 2017) can leverage the knowledge encoded in such resources. Following the work of Lauscher et al. (2020), we try to achieve this by extracting triples of factual information from ConceptNet using a random traversal algorithm and translating them into natural language. The triples are used as the training data for lighweight neural networks set on the masked language modeling objective. We inject the trained networks into the large pre-trained language models BERT (Devlin et al., 2019) and RoBERTa (Y. Liu et al., 2019). We fine-tune and evaluate the injected models on popular NLP tasks believed to require world knowledge. The lightweight neural networks, often referred to as adapters, are inserted into the root language model without changing the original parameters. Consequently, they make for a highly parameter-efficient technique for transfer learning. What we hope to achieve is to demonstrate that adapters are a viable technique for combining structured information and language models that are trained on unstructured data.

Our experiments are based on the following research questions:

(1.2) **RQ1** To what extent can adapters be used to inject knowledge into pre-trained language models?

- (a) Can we reproduce the results achieved by Lauscher et al. (2020)?
- (b) How is the injected knowledge distributed? Can we prune adapters and achieve even better efficiency?
- (c) Are the injected models able to reproduce the factual information seen during adapter training?

RQ2 Can we use adapters to inject knowledge from ConceptNet in order to increase the performance on popular NLP benchmarks believed to require the same kind of knowledge?

- (a) Can we increase the performance of a base BERT model?
- (b) Can we increase the performance of a base RoBERTa model?

RQ3 Can we deploy the newly introduced AdapterFusion (Pfeiffer, Kamath et al., 2021) framework on ConceptNet?

- (a) Can we partition the ConceptNet knowledge graph by predicate type, as done with biomedical data in Meng et al. (2021)?

(b) Can we use attention over multiple adapters in order to increase the recall of factual information in a zero-shot setting over cloze-style statements?

1.1 Overview

Chapter 2 presents the problem of deriving world knowledge from text and how such information can be encoded in knowledge graphs. Furthermore, it reviews previous research on knowledge representation and the combination of structured knowledge with language models. It also provides a brief introduction to neural networks and how they are used and in NLP, with a more in-depth focus on recent architectures and the adapter technique.

Chapter 3 presents a reproduction study of work done by Lauscher et al. (2020). The results of this reproduction motivates the experiments that follow.

Chapter 4 contains information on the experimental part of this thesis. Adapter modules trained over the ConceptNet (Speer, Chin and Havasi, 2017) knowledge graph are injected into two popular language models and evaluated on three downstream NLP tasks believed to require world knowledge and commonsense reasoning capabilities. Furthermore, the chapter provides an in-depth error analysis using probing experiments to showcase the effectiveness of the proposed injection method.

Chapter 5 expand on the findings from chapter 4 by experimenting with new types of adapter setups. By pruning the adapter modules from different transformer layers, we hope to shed light on where and how factual knowledge is encoded into large pre-trained language models. Using the AdapterFusion framework (Pfeiffer, Kamath et al., 2021), we also create an adapter ensemble like model and evaluate its effectiveness over a subset of the LAMA probe (Petroni et al., 2019).

Chapter 6 summarizes the work put forth in this thesis and presents suggestions for future research.

Chapter 2

Background

The first section of this chapter introduces the issue of extracting and representing world knowledge and commonsense information for language modeling purposes. Although this work focuses on reasoning that requires these types of capabilities in particular, recent work on different types of knowledge injection, like medical domain expertise, is of high relevance. Section 2.2 and 2.3 introduce neural networks and how they are used for NLP, with an emphasis on modern architectures and the adapter framework — the method used for knowledge injection in this work. Section 2.4 presents previous work on the combination of external knowledge with language models. Section 2.5 discusses the problem of evaluating the injected models and presents three datasets believed to assess world knowledge and commonsense reasoning capabilities. Together, these sections provide the theoretical foundation for the experimental part that follows in chapter 3, 4 and 5.

2.1 World knowledge in text

Often, the interpretation of an utterance must be understood using information external to the utterance itself. An interpretation of the sentence *it is cold today* is a function not only of the meaning to be found in these four words and their compound, but also of the subject's sensory perception of temperature. That is, the act of understanding language requires more than knowledge of the meaning of words. With this particular example, we can interpret the sentence successfully (we get what is conveyed by the speaker), but if we are unable to feel whether or not it is in fact cold today, understanding is limited. This type of knowledge is often referred to as world knowledge or common sense. Our understanding of language relies on such knowledge that we have accumulated over time through multi-modal interaction with the world and by deductions from implicit observations and facts.

In the case of NLP, we do not focus much on multi-modal information such as perception of temperature. How would we even represent the feeling of temperature textually? However, a lot of world knowledge can be represented using language. If we consider the example from the introduction (see 1.1), the knowledge that Jotunheimen is a mountain range in Norway and not in Sweden can be conveyed as a statement on the form *J is in N* and *J is not in S*. Given this information, our understanding of the statement *Jotunheimen is in Sweden* is more meaningful than it would have been without it. Even though the statement has a valid semantic interpretation, our understanding of it as false depends on external knowledge.

In a language model, one sense of a word has a higher probability in a sequence of words than another based on the frequency of co-existence between the sequence and this word sense in the training material. Thus, if an instance of world knowledge never co-exist in a text, that Jotunheimen is in Norway, there is no way for language models to encode this knowledge. Even though we have instances that mention both Jotunheimen and Norway, the fact that the former *is in* the latter is not made explicit.

Language models are nonetheless able to generate pieces of language with valid semantic interpretations. But even if we explain their ability to do so by their capability for understanding meaning of words, embedded as a probability into vector space, the pieces of language they generate need not to be easily understood.

This begs the question: can world knowledge be extracted from text? The ability to expresses something that is both meaningful and true seems crucial to any notion of human analogous understanding of language. The models of today are deployed as filtering mechanisms, dialogue systems and translators. If there is no way of extracting world knowledge from the training material, this greatly limits such systems.

It is difficult to determine whether or not language models understand facts of the world or only facts of the words of our language. In order to illustrate the difference, we ask the reader to consider these two sentences from Hagoort et al. (2004):

(2.1) a) The present queen of England is divorced

b) The favorite palace of the present queen of England is divorced

Sentence **a** is meaningful; the semantic content is coherent. It is only by knowing that the queen of England is in fact married, a fact about the world, that the sentence can be determined false. Sentence **b** is different, however. There are no valid semantic interpretations available to us, not because the sentence is ungrammatical, but because the predicate *is-divorced* typically takes something animate as its argument. A neural language model can through the use of contextualised embeddings (see section 2.2 for an explanation) learn that this predicate is typically followed by active proper nouns or pronouns, which increases the probability that the particular word denotes a human, and thus generate sentences with a valid semantic interpretation. What it can not, though, is to infer the truth value of either sentence or why *queen* is an valid argument to the predicate but not *palace*. Such reasoning would require world knowledge, not just knowledge about the facts of our language.

2.1.1 Knowledge graphs

Now that we have some notion of what world knowledge could look like in text, how do we go about extracting and representing it? Attempts at structuring facts about the world into both general and more domain specific systems go way back. In the field of philosophy, questions on how entities are grouped into categories are often referred to as the topic manner of ontology — a tradition of thought that predates most others. The term ontology is also used in the fields of knowledge representation and semantic technologies, but there the concern is not only on the formal definition of the groupings but also on the actual implementation of them. In these fields we also find the terms "knowledge bases" and "knowledge graphs".

As all these terms have their own specific connotations depending on which field of study they are employed in, we choose to refer to all attempts at structuring some related state-of-affair in the world into hierarchical systems as ontologies. If the ontology is represented as a graph intended for computational use, we refer to is a knowledge graph. More formally, we can say that our use of the term knowledge graph refers to a world model that meets these criteria:

- it must be a collection of symbols that stands for some state-of-the-affair in the world
- it must organize these symbols in a graph structure
- it must connect these symbols together using relations that provide some semantic interpretation
- preferably, the knowledge graph must also be mappable to other graphs within the same domain.

The knowledge graphs relevant for this work are often constructed by automated solutions, driven by heuristics, and are thus examples of how it is possible to extract and represent world knowledge from text. Below is an overview of some existing graphs, both for domain specific knowledge and for more general world knowledge and commonsense information, with an emphasis on ConceptNet (Speer, Chin and Havasi, 2017), which is the graph we use in this work.

Domain specific graphs

The primary benefit of a domain specific knowledge graph is that it enables inferences from text using knowledge contained in a set of concepts that is narrower and less polysemous. For example, if the domain is artificial intelligence, the concept of "model" has another meaning than it does in everyday conversation and will most likely always point to the same concept in the ontology.

Below are some examples of domain specific graphs:

- The Unified Cybersecurity Ontology (UCO), intended to support information integration and cyber situational awareness (Syed et al., 2016).
- IDEON, for modeling, analyzing, and managing processes within the systems engineering enterprise (A. Madni, C. Madni and Lin, 1998).
- The Systematized Nomenclature of Medicine—Clinical Terms, known as SNOMED CT, a comprehensive medical terminology used for standardizing the storage, retrieval, and exchange of electronic health data (Donnelly et al., 2006).

WordNet

The lexical database WordNet links word classes to sets of synonyms that are in turn linked through semantic relations that determine word definitions (Miller, 1995). It is perhaps the best known structured resource for word level semantic information. The motivation for WordNet was to provide a resource for computational applications that cater to the way machines consume information, not humans. The original distribution of WordNet included more than 118,000 different word forms and more than 90,000 word senses, of which 17% of the total were polysemous.

There have been numerous works using WordNet for NLP applications, as well as multiple extensions, like WordNet::Similarity for semantic similarities between concepts (Pedersen, Patwardhan, Michelizzi et al., 2004) and WordNet-Affect for lexical representation of affective knowledge (Strapparava, Valitutti et al., 2004).

ConceptNet

ConceptNet[1] is an online knowledge graph designed to represent the general knowledge involved in understanding language (Speer, Chin and Havasi, 2017) and is what we use for our experiments. Compared to WordNet, ConceptNet provides semantic relations for concepts, not words. The current iteration, 5.5, is optimized to be used with modern NLP techniques. The original release in H. Liu and P. Singh (2004) was intended as a representation of the crowd-sourced knowledge project Open Mind Common Sense.

The knowledge in ConceptNet is coded as triplets consisting of two concepts and one relation. When combined, they make up an assertion on the form: (*subject*, *relation*, *object*), where both the subject and the object is a concept. For example, the assertion that dogs have tails can be represented by the triple *(dog, HasA, tail)*. Hence, every assertion in ConceptNet can be seen as a subset on the form $Assertion \subset C \times L \times C$ where C is the set of all concepts and L is the set of all relations. Relations include *RelatedTo*, *CapableOf* and *InstanceOf*. A list of all the relations in ConceptNet can be found in Figure A.1 in the appendix and an illustration of an excerpt can

[1]https://conceptnet.io/

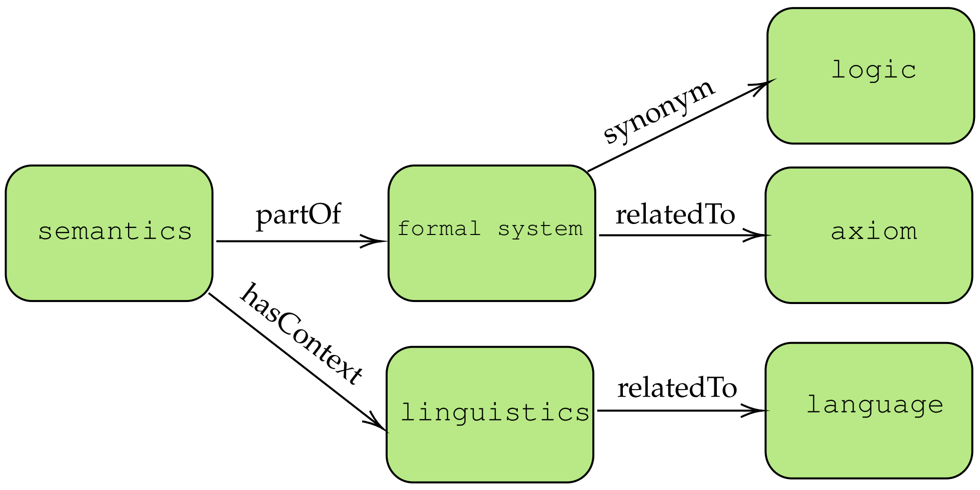


Figure 2.1: An illustration of some concepts found in ConceptNet and their relation

be seen in Figure 2.1. In total, these relations express over 13 million links between concepts and thus provide an attempt at structuring the type of commonsense knowledge that does not necessarily exist explicitly in text. Later in this work we also use the term "predicate" to refer to the same thing as relation.

Before we discuss how knowledge graphs such as ConceptNet can be combined with pre-trained language models in order to better their commonsense reasoning capabilities, we give a brief introduction to neural networks and how they are used in NLP. Furthermore, we introduce the architecture and workings of recent language models, together with a more in-depth explanation of adapters — which is the method used for knowledge injection in this work.

2.2 Modeling

If you want to classify something in a text, how do you transform and condition a model on the data? The following section introduces the basic architecture of a neural network, together with a short presentation of the general architecture behind pre-trained language models, such as ELMo (Peters, Neumann, Iyyer et al., 2018), BERT (Devlin et al., 2019), GPT-3 (Brown et al., 2020) and Switch Transformer (Fedus, Zoph and Shazeer, 2022). This section gives a brief introduction of the concepts needed for the succeeding section on adapters.

2.2.1 Neural networks

Although the theory behind artificial neural networks is old, their usage within NLP is fairly recent. Due to advancements in computer hardware and the publication of large datasets, they have been in widespread use for the last decade.

The explanations that follow are inspired by those given in Jurafsky and Martin (2021) [2], and Goldberg (2017).

2.2.2 Feed-forward networks

Neural networks are centered around the idea of the *neuron* as a computational unit. The neuron takes some real valued number as input, performs some computation, and returns a new real number as output. This computation typically involves multiplying the input with a set of weights and adding a bias term. Thus, the neuron can be formally defined as:

$$z = b + \sum w_i x_i \tag{2.2}$$

Or, in shorthand, using the dot product:

$$z = wx + b \tag{2.3}$$

Where w is the weights matrix, x the input matrix and b the bias term.

A neural network then, is the collection of such neurons into connected layers so that the output of one layer functions as the input to the next. The first layer, known as the input layer, takes the input vectors, transforms them and passes them down the network. The last layer, the output layer, provides the predictions of the network. All other layers are known as *hidden layers*. Together, these layers perform a linear transformation:

$$NN(x) = xW + b \tag{2.4}$$

$$x \in \mathbb{R}^{d_{in}}, W \in \mathbb{R}^{d_{in} \times d_{out}}, b \in \mathbb{R}^{d_{out}}$$

If all neurons in one layer are connected to all the neurons in the next layer, we call the network *fully connected*, and if all the connections goes forward in the network, we call it a *feed-forward network*. An illustration of a such a network can be seen in Figure 2.2.

Non-linear activation

In order for the model to learn information inherent in the input data, we must apply some form of non-linear transformation. Without it, we achieve nothing by scaling from a single computational unit to a full network. We add this non-linear transformation by applying what is referred to as an

[2] https://web.stanford.edu/~jurafsky/slp3/

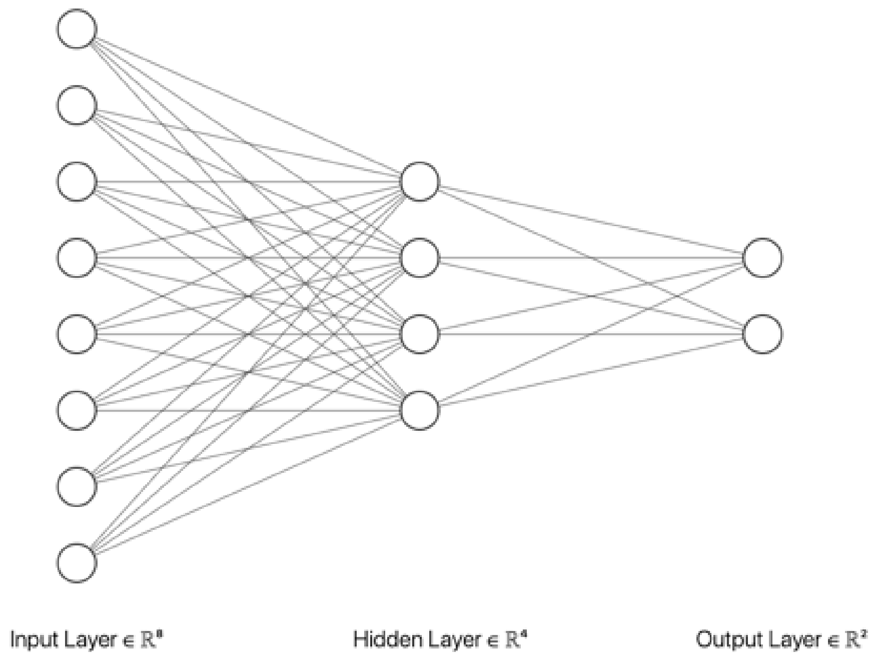


Figure 2.2: A fully connected feed-forward network with one hidden layer

activation function. A popular variation of such a function is the Rectified Linear Unit, *ReLU* for short:

$$f(x) = max(0, x) \tag{2.5}$$

Other examples of non-linear activation functions are the *sigmoid* and the *tanh*. When we have linear transformations followed by such activations in each layer, we can create networks that approximate complex functions – making predictions on hitherto unseen data possible. An example of such a network, with one hidden layer and g as an activation function, can be written formally as:

$$NN_1(x) = g(xW + b) \tag{2.6}$$

Or with two hidden layers and two activation functions, g^1 and g^2:

$$NN_2(x) = (g^2(g^1(xW^1 + b^1)W^2 + b^2))W^3 \tag{2.7}$$

2.2.3 Modern neural architectures

Albeit sufficient for many applications, the simple feed-forward architecture must often yield for more advanced models. A common architecture in

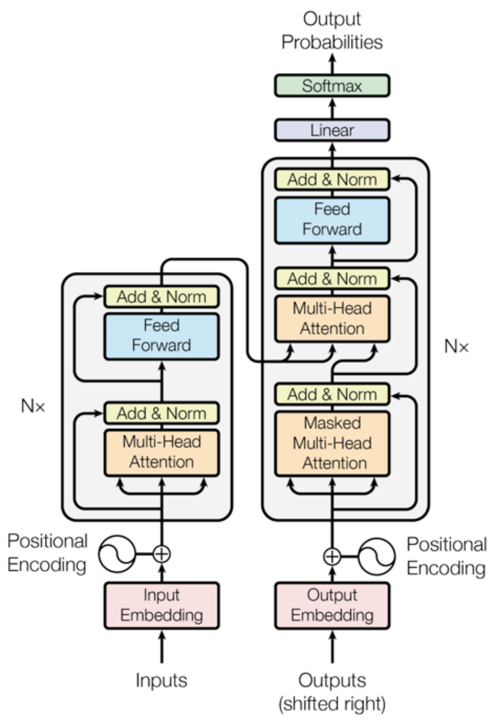


Figure 2.3: The original transformer architecture proposed in Vaswani et al. (2017)

NLP is the Recurrent Neural Network, *RNN*, with its variations Long Short Term Memory network (LSTM) (Hochreiter and Schmidhuber, 1997) and its bidirectional relatives BiLSTM (Schuster and Paliwal, 1997) and GRU (Cho et al., 2014). Another architecture class, mostly used for image recognition tasks but also with some applications in NLP, is the Convolution Neural Network, *CNN* (LeCun et al., 1999).

A third option, which has seen massive uptake in recent years, is the transformer architecture, which will be the focus for the remaining part of this section, and also the most relevant architecture for this work.

The transformer

First introduced in Vaswani et al. (2017), the transformer architecture has become the dominant flavor of artificial neural networks in NLP. The development has in large been motivated by challenges inherent in the architecture of its most used predecessor, the RNN.

The RNN generates a sequence of hidden states as a function of previous states, resulting in a recurrence across iterations that make up the models internal memory. This enables more context sensitive modeling than a plain feed forward network; the output of a layer is no longer only a function of its current state but also the output of previous states. Although effective in its predictions, the RNN is not efficient to train. Furthermore, if the input sequences become too long we encounter the problem of vanishing gradients which will prevent the weights in the network from changing enough to encode new information.

In order to model dependencies across the whole sequence without regard to length and the position of word tokens, we can use an attentive mechanism. Conceptually, this makes it possible to spread information across sequences by concentrating on what is relevant in a training instance while ignoring the rest. The transformer architecture relies entirely on such mechanisms, making any recurrence and convolutions redundant (Vaswani et al., 2017).

After the input sequence has been embedded into vector space by assigning a numerical value for each token and given a positional weight, we pass the vectors through a special type of attention layer, called self-attention. This mechanism will encode how important each word is in relation to other words in the same sentence. This makes it possible for the model to access other words in an input sentence as it encodes a single token, providing a notion of context. Formally, the attention is calculated as a scaled dot product between the vector representing the current query, often a single word, Q, and the hidden state in the encoder block, K, which will hold information on all previous words, acting as a memory. This product is normalized to 1 using a softmax function before being multiplied by V, which is often the same tensor as K (Vaswani et al., 2017):

$$Attention(Q, K, V) = softmax(\frac{QK^\top}{\sqrt{n}})V \tag{2.8}$$

In order for the model to see information from different parts of the input representations, the attention mechanism is calculated multiple times for different projections of the keys (K), values (V) and queries (Q) – all in parallel. This Multi-Head attention calculation is used in three different ways in the transformer, as can bee seen in Figure 2.3.

2.2.4 Language models

How can we use architectures like feed-forward neural networks and the transformer to work with language? If we are to solve tasks that involve the generation of phrases, for example a dialogue system, we want our system to respond to some context. The way to approach this problem has been and still is to create a probability distribution over sequences of words. This way, we can calculate what the most probable next word is to some textual context.

Although the general approach to modeling language has stayed somewhat the same, the particular techniques for achieving it have evolved a lot. The early *N-gram* modeling technique, for example, estimates the conditional probability of the next word given the previous N words, while the more recent WORD2VEC framework (Mikolov et al., 2013) embeds

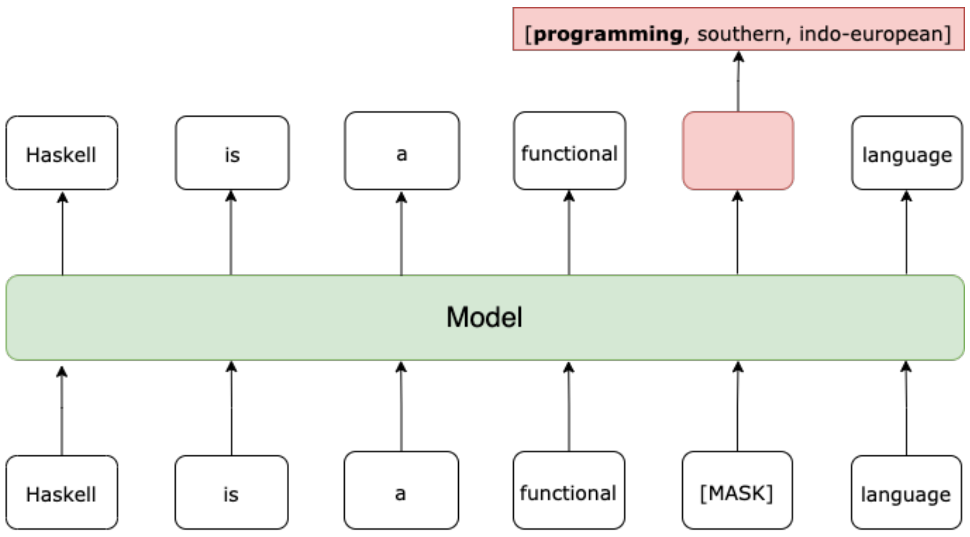


Figure 2.4: An illustration of the masked language modeling objective.

each word into vector space by training a neural network on the task of predicting a word given its context, or vice versa.

A method that has achieved great results in recent years, most noticeably as a training regimen for large language models, is the masked language modeling objective (MLM). Popularized in Devlin et al. (2019), this method masks a percentage of tokens in a sequence and trains a model on the task of recovering the words beneath the masks. It is primarily the MLM objective that is relevant for the models used in this work.

The great advantage of MLM over other language modeling objectives is that it enables the model to be bidirectional, focusing on the both right and left contexts. After training, the resulting network contain representations of words commonly refereed to as contextualized embeddings. Figure 2.4 illustrates this procedure. The most probable word for the masked token is calculated by applying the softmax function to the scores calculated by the network for all possible words, which will be equal to length of the models vocabulary:

(2.9)
$$\sigma(Z)_i = \frac{e^{Z_i}}{\sum_{j=1}^{K} e^{z_j}},$$

where Z is the output of the model, e^{Z_i} the exponential function applied to each element of Z and K is the number of words in our vocabulary. The denominator is the normalization that ensures that all the probabilities add up to 1. The token with the highest probability is chosen as the correct word for the mask, and the weights in the model's network are adjusted according to the loss calculated between the guessed and actual token.

The exact workings of masked language modeling is not relevant for this work. However, the adapters used for knowledge injected use MLM as their objective when training over the extracted triples from ConceptNet. As MLM is also the training objective during the models initial pre-training (see next section), this have the added benefit of not introducing a new objective during the knowledge injection phase, relying instead on known patterns.

2.2.5 Pre-trained language models

At the time of writing, the most popular model in NLP is BERT (Devlin et al., 2019). Trained on the objective of masked language modeling (MLM) and next sentence prediction (NSP), this framework has become the dominant flavor of what is refereed to as pre-trained language models. Models like BERT enable anyone working on NLP to leverage a model trained on large datasets, such as Wikipedia and CommonCrawl, without having access to the required resources themselves. The pre-trained models can be downloaded from publicly available API's and then further trained on the specific problem of the end user. This approach is known as fine-tuning.

Architecturally, BERT uses more layers, more attention heads and larger dimensions for the feedforward-networks in each layer than the original transformer from Vaswani et al. (2017). It as also bidirectional, making the representations dependent on both left and right contexts. The base version released in Devlin et al. (2019) has approximately 110 million tuneable parameters.

In addition to BERT, we also use the popular RoBERTa (Y. Liu et al., 2019) model for our adapter injection. While the masking of tokens in BERT only happens once during data preprocessing, before being fed as input to the model, the pre-training regimen used in Y. Liu et al. (2019) duplicate the training data ten times so that each sequence is masked in ten different ways — a method known as dynamic masking. Furthermore, RoBERTa also drops the loss values for the NSP tasks, only relying on dynamic masking for modeling long range dependencies.

In order to use pre-trained language models on a downstream task, we have to add a classification layer, often referred to as the *head*, on top of the model. This layer reduces the output of the language model to the required dimensions of the specific problem at hand. For masked language modeling, the classification layer will be a linear transformation from the size of the last layer in the language model, M, to the length of the models vocabulary N. For $\text{BERT}_{\text{BASE}}$, this will be $M = 768$ to $N = 30522$ (Devlin et al., 2019); and for binary sentiment analysis with the same model it would be from $M = 768$ to $N = 2$.

Interpretability. Although pre-trained language models achieve unmatched performance on a range of NLP tasks, it is not so easy to interpret what they actually learn during training. Understanding why a model made or did not make a particular prediction is difficult when the overall capacity of the model is distributed over 110 million parameters, as in BERT, or the 175 billion parameters in GPT-3 (Brown et al., 2020). This has spiked a research interest into the interpretability of such language models.

One such study, by Rogers, Kovaleva and Rumshisky (2020), systematize and presents what over 150 different publications using BERT has taught us about how transformer-based architectures represent knowledge and what information they learn. Much of the surveyed work uses a technique known as *probing* to assess whether or not BERT actually learns certain linguistic features or not. Using probing, work by Tenney, Das and Pavlick (2019) suggest that BERT actually learns parts of the traditional NLP-pipeline, and even so in-order — starting with POS tagging, parsing, NER and semantic roles, and finally with co-reference. We rely on this way of probing in the evaluation of our models in chapter 4 and 5, where we try to isolate factual knowledge.

As BERT is in short a stack of layers, one can also probe individual layers in order to quantify where specific types of knowledge are distributed or most prominent. For example, multiple studies suggest that lower layers have the most information on linear word order, that middle layers focus on encoding syntactic information and that the last layers are the most task-specific (Rogers, Kovaleva and Rumshisky, 2020). As pointed out earlier, we use masked language modeling as the training objective for our adapter modules. This suggests that whatever information is acquired during adapter training should be most prominent in the top layers of the injected model. Chapter 5 prunes different layers in order to investigate whether or not our models align with this intuition.

It must also be pointed out that recent studies, such as work by Sinha et al. (2021), have shown pre-trained language models to be insensitive to word order. We argue that this challenges the claim that knowledge, at least the human analogous type, could be represented in a way that makes it reasonable to talk about layer-specific locations.

2.3 Adapters

The training approach used today, where we fine-tune a pre-trained model on every downstream task that we want to solve, is inefficient. Adapters mitigate this by allowing for efficient parameterization and sharing (Houlsby et al., 2019). Usually, this is done by introducing lightweight neural networks into transformer-based models and only adjusting the weights of this network on the downstream task. Thus, the weights of the original network are untouched, whilst the new adapter layers, which have few parameters, are fine-tuned and can be extracted and shared between models.

Although fairly new, this approach have proven effective. For a review of how adapters are currently being used in NLP, see section 2.4. The following subsections outline the technicalities of the most popular adapter architectures and some of their problems.

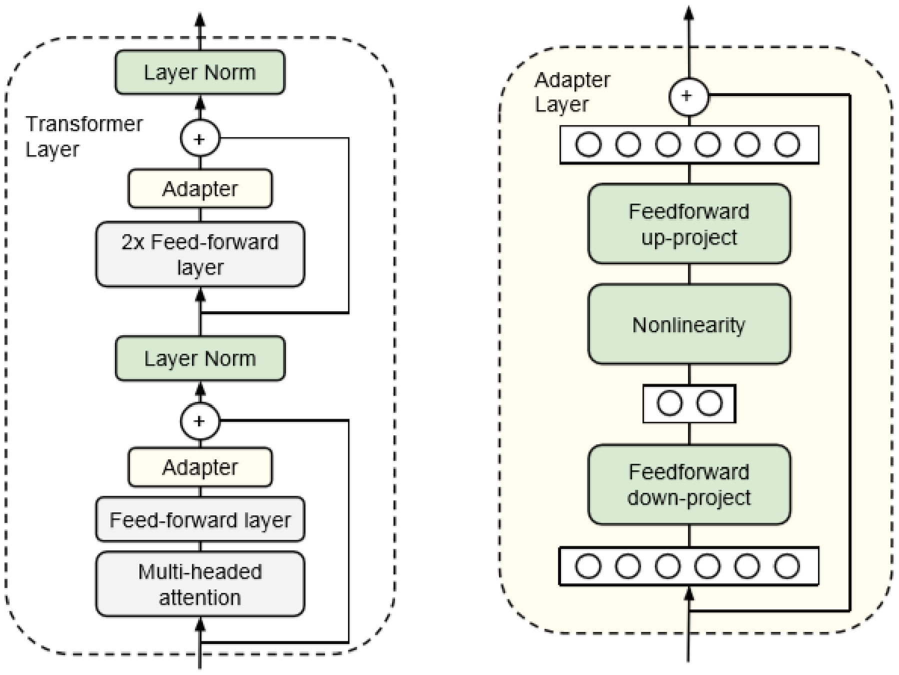


Figure 2.5: The single-task adapter architecture introduced by Houlsby et al. (2019).

2.3.1 Single-task adapters

In a single-task adapter, for each of the N downstream tasks that we want to train an adapter for, a pre-trained model like BERT is initialized with parameters θ_0. In addition, a set of new and randomly initialized adapter parameters ϕ_n is introduced (Pfeiffer, Kamath et al., 2021).

The right side of Figure 2.5 shows the general architecture of each of these adapters. ϕ_n is distributed across two feed-forward networks, denoted as the up and down project in the figure. After passing through the multi-headed attention layer in the original model, the input vectors enter the first of these two networks, which has a low-dimensionality in order to keep the total parameter size of the adapters low. After this bottleneck, the data is transformed with a nonlinear function before being projected back into the original dimensionality using the second network. The nonlinear transformation is typically done with the gaussian error linear unit (GELU) which weights inputs by their percentile in the standard gaussion distribution function, rather than by their sign as with ReLU in equation 2.5 (Hendrycks and Gimpel, 2016). Some adapter implementations also use the SWISH function. Figure 2.6 illustrates their difference over a small interval.

In practice, as done by Houlsby et al. (2019) and Lauscher et al. (2020), each adapter module is injected into each transformer layer twice, as

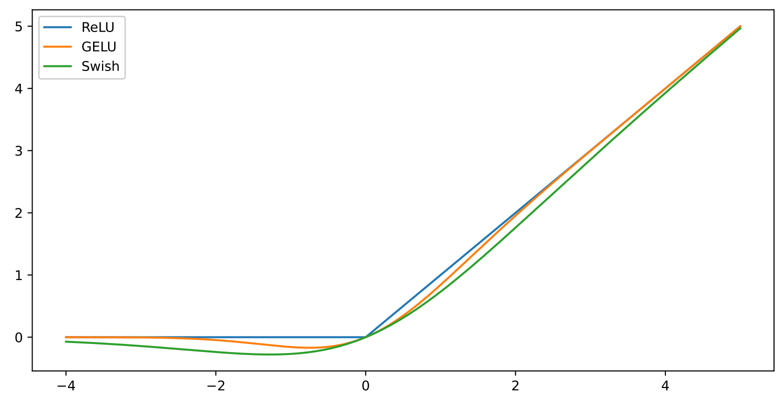


Figure 2.6: A comparison of popular activation functions for adapters

illustrated by the left side of Figure 2.5. For each adapter, if we let W_d and W_u be the parameters of the down-projection and up-projection layers respectively, together with bias terms b_d and b_u and a nonlinear activation function, f, we can define each adapter as a function of its input vector X:

$$Adapter(X) = X + f\left(XW_d + b_d\right)W_u + b_u \tag{2.10}$$

Catastrophic forgetting

A common issue with transfer learning in general is that weights learned from the new task interfere with the contextual information gained in the initial large scale training of the model. This phenomenon is known as catastrophic forgetting.

This problem is pressing in numerous applications of transfer learning, such as Peters, Neumann, Logan et al. (2019) and Z. Zhang et al. (2019). As a method of transfer learning, the same goes for adapters. The solution could either be to apply a normalization technique that dampens this effect, or, as proposed in R. Wang et al. (2021) to ensure that all original parameters are kept static at all times by only adjusting the introduced weights, ϕ_n — which is the technique we apt for in chapter 4 and 5.

2.3.2 AdapterFusion

One problem of the single-task approach is that the distinct weights of the adapters prevent the classification layer from utilizing different sources of information. Pfeiffer, Kamath et al. (2021) propose a two stage algorithm that enables sharing of information between adapters trained on different knowledge tasks, known as AdapterFusion.

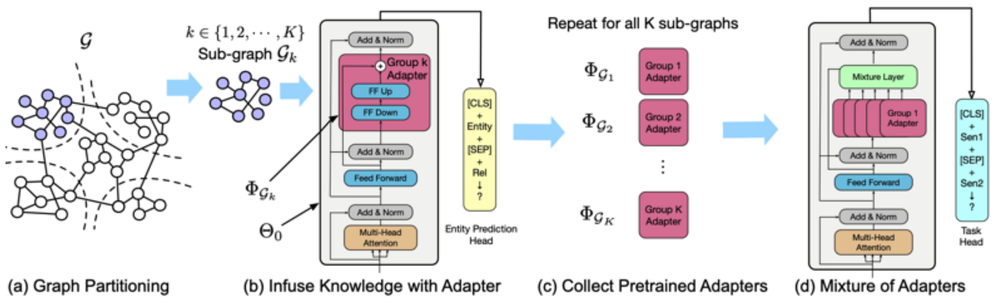


Figure 2.7: The usage of the AdapterFusion technique in Meng et al. (2021).

Here, for a set of N downstream tasks, we train an adapter for every $n \in N$. This first stage of the algorithm generates a set of adapters, S. In the second stage we freeze the weights of the original pre-trained model, $\theta_{original}$ and the weights of each adapter, ϕ_S. Upon fine-tuning on our goal downstream task, which can be a member of N or not, we then introduce a new set of parameters γ which through an attention mechanism learns to combine the knowledge encoded in each of the adapters in S in order to solve the final task. This approach address the issues of catastrophic forgetting and parameter sharing between tasks. One can imagine the AdapterFusion approach as having access to a panel of domain experts, all with disjoint expertise, and then being tasked with learning when and how much you should listen any one expert at a given point in time, that is for each task. For example, if we are to solve an inference task and believe that knowledge of named entities and sentiment in text might provide added benefits we can train one adapter on a Named Entity Recogniton (NER) dataset and one one a sentiment analysis set. When fine-tuning on our inference task we add an AdapterFusion layer that learns which of the two adapters are most helpful for each sample in that task.

Figure 2.7 illustrates how Meng et al. (2021) applies this to the medical domain by training a single-task adapter on distinct partitions of a medical knowledge graph.

2.3.3 Performance

The usage of a bottleneck in the adapter modules is a trade-off between performance and parameter efficiency. In order for adapters to be useful at all, this balance must be favorable.

The single-task adapter used in Houlsby et al. (2019) attain within 0.4% difference of the performance of a standard full fine-tuning approach on the GLUE benchmark, only tuning 3.6% of the total parameters. This gives improvement both in terms of space complexity and the time needed to fine-tune the model. The performance of the K-adapter model in R. Wang et al. (2021) even surpasses that of the original model with full fine-tuning.

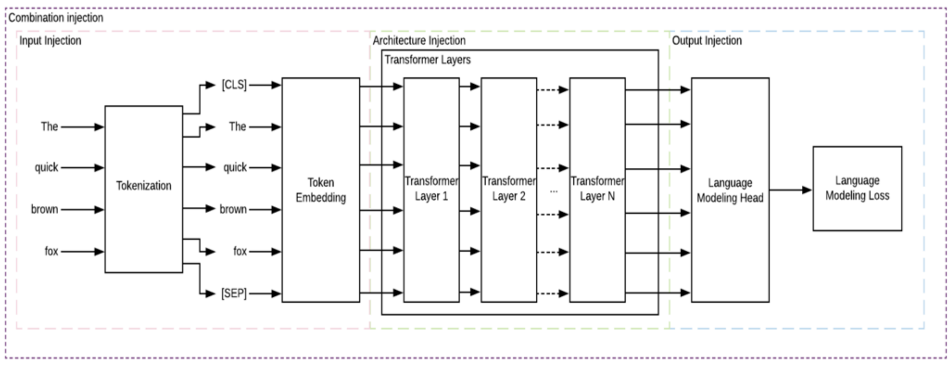


Figure 2.8: The difference between the three primary types of knowledge injection. Figure from Colon-Hernandez et al. (2021).

For an extensive analysis on the efficiency of adapters, see Rücklé et al. (2021).

2.4 Previous work

As we have now seen both how world knowledge can be represented structurally, for example as a graph, and how modern neural networks can create language representations from unstructured text, we now turn to how previous work have attempted to combine the two. The following section discusses both how adapters in particular have been used for knowledge injection, but also other related attempts at the combination of structured information with language models.

At one point in time, the usage of structured information, such as ontologies, knowledge bases and graphs, were upheld by many in the NLP communtiy to be a vital component to any NLP system, as illustrated by this quote from Mahesh, Nirenburg et al. (1995):

> In the field of natural language processing there is now a consensus that all NLP systems that seek to represent and manipulate meanings of texts need an ontology.

Now, the opposite seems to be the consensus. It is the modeling of the usage of language, not manipulation of concepts, that functions as the theoretical premise for much of what NLP has achieved the last decade or so. Representation learning based on observed language use, for example through tasks like masked language modeling, has proven to be an effective starting point for a range of downstream NLP tasks (Devlin et al., 2019).

However, there has been a recent rise in attempts at combining information from external resources with state-of-the-art neural language models, such as the transformer (Vaswani et al., 2017). This has in large been motivated by the observation that such models tend to "hallucinate knowledge", that is to generate imprecise constructs (Colon-Hernandez et al., 2021), and have difficulty recovering factual knowledge (Logan et al., 2019). Following Colon-Hernandez et al. (2021), we separate previous attempts into three categories (see Figure 2.8 for a graphical visualization):

- Input-focused injection: modifying the training data by introducing structured knowledge.
- Architecture-focused injection: altering the model architecture, e.g: injecting a feed-forward layer trained on a knowledge graph into the transformer layers (such as adapters).
- Output-focused injection: modifying the output of a model and/or introduce a custom loss function.

Although fairly recent, the literature on the different types of injection is already considerable. For a review of approaches in all three categories, we refer the reader to the survey by Colon-Hernandez et al. (2021), and continue with a more thorough review of architecture-focused injection attempts, of which many uses the same general approach: to insert variations of neural networks, trained on a knowledge resource through some learning objective, into some parts of a language model.

The adapter technique as we have presented it thus far is one example of an architecture-focused injection and stems from work by Rebuffi, Bilen and Vedaldi (2017) on the parametrization of residual networks. It was originally conceived of as a method of preserving domain specificity when faced with the rise of universal families of neural networks. The approach was initially developed for image recognition and later introduced in NLP by Houlsby et al. (2019).

In contrast to the classic full fine-tuning approach, where all the parameters of the pre-trained model are adjusted with regard to the downstream task, Houlsby et al. (2019) injects variations of the residual adapter modules from Rebuffi, Bilen and Vedaldi (2017) into the architecture of BERT. This introduces a new set of trainable parameters to an already large parameter space. However, it is only the new set of parameters that gets to fine-tune on the downstream task. Thus, per task, this approach only adds and adjusts 3.6% of the 110 million scalar values in the base BERT model, while the standard fine-tuning approach adjusts all of them. Despite this, the adapter injected method performs comparable to the baseline on the GLUE benchmark (A. Wang, A. Singh et al., 2018), attaining within 0.4% of the performance of full fine-tuning.

Whereas Houlsby et al. (2019) introduce adapters into each transformer layer, all trained on one task, R. Wang et al. (2021) inject different types of knowledge into separate adapter models: one that captures factual knowledge obtained from aligned text triplets on Wikipedia and Wikidata,

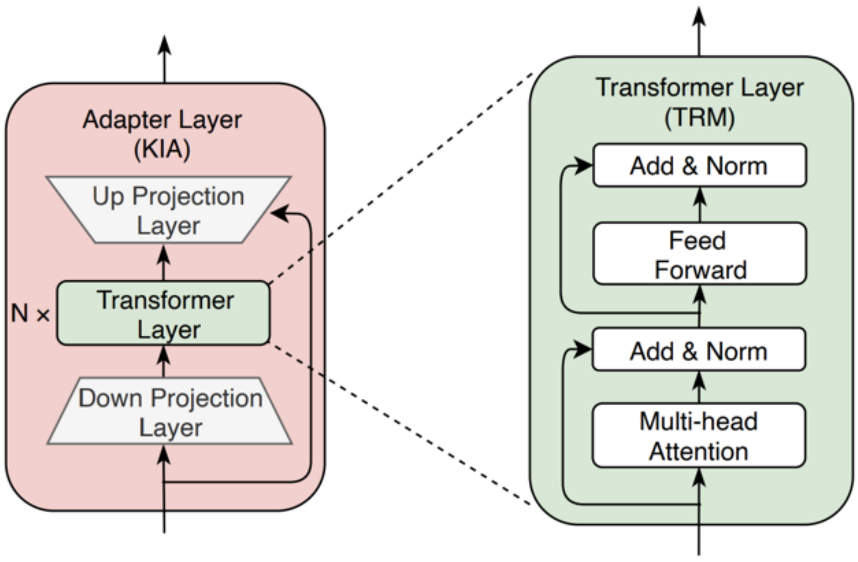


Figure 2.9: The adapter architecture proposed in R. Wang et al. (2021).

and one that captures linguistic knowledge obtained via running an off-the-shelf dependency parser over a subset of the Book Corpus (Zhu et al., 2015). The adapters are combined with a base ROBERTA model and the resulting architecture, named K-ADAPTER, demonstrate increased performance on three knowledge-driven downstream tasks.

The novelty of this approach is that the knowledge-specific adapters work as outside plug-ins and are not injected inside the layers of the original. In practice, this means that the input of a given adapter module is the output of the current layer in the pre-trained model, concatenated with the output of the former adapter layer. Figure 2.9 shows the adapter architecture used for the K-ADAPTER model, as compared to the architecture by Houlsby et al. (2019) in Figure 2.5. We note that we find the explanation of the specific workings of the adapters in this work to be somewhat unclear. Furthermore, as the approach is only tested with ROBERTA as the underlying language model, the evidence for its effectiveness is not as convincing as those achieved by the architecture in Houlsby et al. (2019).

Instead of using adapters, it is also possible to integrate the structured information from a knowledge graph in other ways. Peters, Neumann, Logan et al. (2019) propose a general method named *Knowledge Attention and Recontextualization* (KAR), where they use an entity linker to retrieve relevant entity embeddings (a vector representation of an entity in a graph) from WordNet, among others, in order to form knowledge enhanced entity-span representations of text. The wordpiece embeddings that the root language model produces by default is recontextualized by substituting the multi-headed self-attention mechanism (see section 2.2.3) with a multi-headed attention between these wordpiece embeddings and the knowledge enhanced entity span vector returned by the entity linker. KAR is injected in between two layers in the middle of the stack in BERT.

One of the novelties of this approach is also that the entity linker is learned using self-supervision on unlabelled data. The overall model, named KNOWBERT, consequently learns to otimize the loss of the underlying BERT language model, but also the entity linker:

(2.11)
$$\ell_{KnowBert} = \ell_{bert} + \ell_{EntityLinker}$$

The evaluation of KNOWBERT, using a perplexity measure and a factual recall probe on Cloze-style statements, shows that the injection of knowledge graphs to BERT, such as WordNet, significantly improves the model's ability to recall facts. It also shows improvement in performance for relationship extraction, entity typing and word sense disambiguation.

It is also possible to use architecture-focused injection, such as adapters, for more domain specific tasks. By partitioning large knowledge graphs in the biomedical domain into smaller sub-graphs, which are then fed into a set of respective adapter modules, Meng et al. (2021) apply mixture layers that combine the knowledge from all the adapters using attention (see section 2.2 for a more detailed explanation of this framework). In short, the usage of attention mechanisms enables the model to learn which adapter to extract knowledge from. Since the sub-graphs used as training data are disjoint sets, one can imagine having a panel of experts, with no overlap in knowledge, to draw from. The adapter modules and the mixture layers are instantiated using the ADAPTERHUB API and the recently proposed AdapterFusion architecture (Pfeiffer, Kamath et al., 2021; Pfeiffer, Rücklé et al., 2020). It is this work that inspires the experiments we present in chapter 5, as outlined in **RQ3**.

The most relevant work for this thesis is that by Lauscher et al. (2020). Using the adapter technique introduced in Houlsby et al. (2019), they inject BERT with knowledge extracted from ConceptNet (Speer, Chin and Havasi, 2017). The injected information, which is on a `subject-predicate-object` format, is extracted using a random walk procedure and translated into natural language. For example, the assertion triple (`Christianity, InstanceOf, Religion`) is transformed into the statement: *Christianity is an instance of religion*. The adapters are trained on these statements using the masked language modeling and injected into every transformer layer in BERT. The original parameters in BERT are kept static during the training of the adapter.

The overall model is fine-tuned and evaluated on tasks from the GLUE benchmark (A. Wang, Pruksachatkun et al., 2019; A. Wang, A. Singh et al., 2018). Although the model is only comparable to the base model on most tasks, it shows some improvement on the diagnostic set (see section 2.5.1)

Model/Work	Knowledge source	Injection style	Objective
GREASELM (X. Zhang et al., 2022)	ConceptNet (Speer, Chin and Havasi, 2017)	Graph neural network integration	Question answering
MOP (Meng et al., 2021)	UMLS (Bodenreider, 2004)	Intrinsic adapters	Biomedical question answering, document classification and natural language inference.
K-ADAPTER (R. Wang et al., 2021)	Wikidata and Wikipedia	Extrinsic adapters	Entity typing, question answering and relation classification
Lauscher et al. (2020)	ConceptNet (Speer, Chin and Havasi, 2017)	Intrinsic adapters	NLU tasks
BERT-MK (He et al., 2020)	UMLS (Bodenreider, 2004)	Aggregator blocks	Relation classification and entity typing
SENSEBERT (Levine et al., 2020)	Wordnet (Miller, 1995)	Entity linking	Word sense disambiguation
KNOW-BERT (Peters, Neumann, Logan et al., 2019)	WordNet (Miller, 1995) and Wikipedia	Entity linking	Relation classification and entity typing
ERNIE (Z. Zhang et al., 2019)	Wikidata	Entity linking	Relation classification and entity typing

Table 2.1: A tabular summary of previous work at combing external knowledge with pre-trained language models

— where some fine-grained categories require world knowledge in order to be classified correctly, the same type of information believed encoded into ConceptNet.

Although there exists additional work that are of high relevance to the approach we employ later in this thesis, we limit our review of these by presenting a summary of the already presented work together with other relevant publications on the combination of structured knowledge and pre-trained language models. This summary, see Table 2.1, distinguish previous work by what knowledge source they try to integrate, their injection style and overall objective. We separate the use of adapters into intrinsic and extrinsic approaches depending on whether or not the adapters are injected directly into the model's layers or residing outside. We label approaches using some form of mapping between entities in text and entities in the knowledge source as "Entity linking" — regardless of how the mapping is injected into the underlying model.

2.5 Evaluation

When we combine external resources with pre-trained language models, the added information will only make up some percentage of the total model size. As mentioned, the adapter injection technique used by Houlsby et al. (2019) increases the total model size with as little as 3.6%. This poses one key problem with regard to evaluation: How can we measure that the injection of external information was successful?

As this work attempts to inject world knowledge and commonsense factual information, we will follow Lauscher et al. (2020), Meng et al. (2021), R. Wang et al. (2021) and Gajbhiye, Moubayed and Bradley (2021) in using the task Natural Language Inference (NLI) as an assessment method. The diagnostic set from GLUE (A. Wang, A. Singh et al., 2018), which relies on the NLI format, provides fine-grained categories for the evaluation of world knowledge and commonsense reasoning capabilities. Furthermore, we also evaluate our models on the tasks of grounded commonsense inference and question answering — both believed to target the same type of knowledge.

The following sections present these tasks — focusing on NLI — together with relevant datasets, challenges and a discussion on their potential for evaluating our proposed method of knowledge injection.

2.5.1 Natural language inference

Consider the following sentence pair:

(2.12) I was born in March.
My birthday is on Christmas day.

Does the latter follow from or contradict the former? Perhaps neither? A system that can answer this question, for any two textual fragments, solves the task of Natural Language Inference (NLI). Today, NLI is often a multinomial classification task where models predict the relationship between two sentences by choosing a label from the well-defined set $\{Entailement, Contradiction, Neutral\}$. However, the task is also commonly referred to as RTE, *Recognizing Textual Entailment*, where the challenge is a binary decision problem, typically predicting whether or not there is an entailing relationship between the two sentences. The variations and formal definitions are plentiful, but they all share the same root mission: to determine whether a natural language hypothesis h can justifiably be inferred from a natural language premise p (MacCartney and Manning, 2009, p. IV).

It is important to note that NLI is different from other inference tasks, as it is not strictly targeted at deriving formal conclusions from premises. Sometimes, upon determining the relationship between the two sentences, we do not deduce formally, but rather, in an informal sense. This is what makes it interesting as an evaluation method for our knowledge injected models. The following two examples, inspired by MacCartney and Manning (2009), illustrate the difference between these two types of reasoning:

(2.13) p Several students at the University of Oslo, when asked by a journalist, said that they would like to have less homework.
h Some students asked by the journalist reported wanting less homework on their mid-term class evaluations.

(2.14) p All books have at least one page.
h My book has at least one page.

The hypothesis in 2.13 is not a logical implication of the premise, as what one reports on an official evaluation is not logically determined by what one has previously said to a journalist. The students can boast about their lack of need of homework to the journalist whilst reporting something seemingly contradictory on the evaluation. Hence, the sentences can have the same truth value at the same time without creating a logical contradiction. Informally, however, the hypothesis is an entailment of the premise, as it is reasonable to believe that the students are somewhat coherent and honest. In the domain of NLI this is often sufficient for counting as an example of an entailing relation.

In this particular sense, then, the relationship between premise and hypothesis is judged on the basis of pragmatics. Neither semantics nor syntax alone can determine the relationship. Consequently, what is required to solve the task is a combination of lexical inference and world knowledge. As the world knowledge of any language model (see section 2.2.4) is limited to what is conferred implicitly through its training data – constructing a limited context on which meaning can be extracted – determining this balance must be a pragmatic choice. There are no linguistic phenomena in the premise that one can apply to the classical rules of inference in order to arrive at the hypothesis. As pragmatics are somewhat ambiguous by nature, annotation efforts for NLI become especially challenging, making the process of creating solid datasets to train and validate models on difficult.

This way of approaching inference follows what is outlined by MacCartney and Manning (2009). But of course, not all inferences are done in this informal sense. Most premise/hypothesis pairs are similar to the one in example 2.14. If we accept that entities in the premise and hypothesis are co-referent, in this case that the books in p are the same books as in h, we can use formal rules of inference to conclude that there is an entailing relationship between the two sentences. We can express this using a universal quantifier. If $P(x)$ is the predicate "x has more than one page", then:

(2.15)
$$\begin{array}{l} \forall x P(x) \\ x \\ \hline \therefore P(x) \end{array}$$

In the popular Pascal RTE challenge sets, starting with Dagan, Glickman and Magnini (2005), and later in the MultiNLI (Williams, Nangia and S. Bowman, 2018), the criterion upon which the nature of a relation is decided was presented to the annotators as the following: for a relation to be labeled as *Entailment*, the hypothesis must state something that is definitely correct about the situation or event in the premise. Their definition is closer to the requirement illustrated in example 2.15 than the informal variants described in MacCartney and Manning (2009). Yet, what counts as "definitely correct" does not seem to be the same as logically sound.

To conclude: depending on the dataset, both 2.13 and 2.14 might be annotated as examples of entailment. It is this mix of occasional informal reasoning, semantics and variability of linguistic expression, not only strict evaluation of syntactical transformations using rules of inference, that makes NLI stand out as an inference task (MacCartney and Manning, 2009) and, for the same reasons, why NLI is a suitable task for evaluating the effectiveness of our adapter-injected models.

Datasets for NLI

The widespread interest for NLI began with the FraCas test suite and the Pascal RTE Challenges. Both the FraCas (Cooper et al., 1996) and the RTE sets, which came out in iterations, starting with Dagan, Glickman and Magnini (2005), aimed to provide a generic evaluation framework for testing the inferential capabilities of NLP systems.

Since then, extensive work has been done for monolingual NLI – especially for English. There are more NLI datasets than what is plausible to introduce here; however, the two most commonly used datasets today are the SNLI (S. R. Bowman, Angeli et al., 2015) and MultiNLI (Williams, Nangia and S. Bowman, 2018). This section briefly presents these two datasets, as well as some influential contributions utilizing the NLI format to benchmark language models.

Premise	**Gold label**	**Hypothesis**
A man inspects the uniform of a figure in some East Asian country.	*contradiction* C C C C C	The man is sleeping
A smiling costumed woman is holding an umbrella.	*neutral* N N C E N	A happy woman in a fairy costume holds an umbrella.
A soccer game with multiple males playing.	*entailment* E E E E E	Some men are playing a sport.

Table 2.2: Sample sentences from SNLI (S. R. Bowman, Angeli et al., 2015).The gold label from all five annotators are illustrated with a single letter denoting the annotation decision, together with the final gold label.

SNLI. The Stanford Natural Language Inference (SNLI) corpus, made available in S. R. Bowman, Angeli et al. (2015), contains 570k human-written sentence pairs that are all manually labeled with one of the three relationship labels: *contradiction*, *entailment* and *neutral*. The annotations were crowd-sourced by providing annotators with automatically extracted image captions and then asking them to create three alternate captions: one that is definitively a true description of the photo, one that might be true and one that is definitively false. Examples of such sentences can be seen in Table 2.2.

10% of the dataset was also validated by asking sets of five annotators to choose a single gold label for the same sentence pair. Examples of such inter-annotator agreement can also be seen in Table 2.2. The neutral example in Table 2.2 illustrates that annotators often disagree. In this case, the hypothesis is clearly not logically deducible from the premise, but the two are not semantically unrelated either, which might explain why this particular sample has been annotated with all three gold labels by different annotators.

MultiNli. A problem with the SNLI was its limited origin of text. This motivated the need for the Multi-Genre Natural Language Inference corpus, MNLI, introduced in Williams, Nangia and S. Bowman (2018). This corpus has a more diverse range of media types, including sentence pairs from face-to-face conversations, fiction, letters and telephone transcripts. The annotations were created in a similar fashion as the SNLI , asking crowd-sourced workers to generate hypotheses based on provided premises. We use the MNLI as one of the evaluation datasets for our reproduction study in the next chapter.

The new media types brought with them new linguistic phenomena that drastically increased the difficulty of the problem. Baseline models

Premise	Predicted label	Hypothesis
Boats in daily use lie within feet of the fashionable bars and restaurants.	*Entailment*	There are boats close to bars and restaurants
restaurants and use feet of fashionable lie the in Boats within bars daily .	*Entailment*	bars restaurants are There and to close boats .

Table 2.3: An original sentence pair from MNLI and a random permutation of the same pair, both receiving the same predicted label in Sinha et al., 2021

performed about 15% better on the SNLI than on the MNLI when trained on both sets (Williams, Nangia and S. Bowman, 2018). Currently, the highest performing model on the MNLI is ERNIE from Sun et al. (2021), achieving 92.3% accuracy. It is worth mentioning here that recent work by Sinha et al., 2021 shows that a permutation scheme of the word ordering in the sentence pairs lifts the accuracy up to 98.7%, i.e there is evidently statistical irregularities in the MNLI that can explain some of the performance. Table 2.3 shows an example of such a permutation. Challenges such as this will be discussed further in section 2.5.1 and in the error analysis of our experiments in chapter 4.

MNLI provides a partition for training, development and testing, with 392,7k, 20k and 20k samples respectively. The development and test sets are further separated into matched and mismatched sub sets, depending on whether or not the samples were in- or cross-domain.

The Glue benchmarks. In order to better measure the development in NLP, A. Wang, A. Singh et al. (2018) introduced GLUE – a benchmark platform collecting multiple common NLP tasks into a single test suite with a single overall performance score. Although we never evaluate our models on the entire suite, we make recurring references to it throughout this work. After only a year, the tasks in GLUE – of which the MNLI test sets were a part – proved too easy, motivating the creation of a harder benchmark, consequently labeled SUPERGLUE (A. Wang, Pruksachatkun et al., 2019). The pure NLI task, as it has been described here, was removed from SUPERGLUE, but it persists in an auxiliary diagnostic set which all model submitters are required to evaluate their models on. This diagnostic set measures the accuracy of a NLI classifier on fine-grained categories, like specific linguistic phenomena, world knowledge and commonsense. The set was introduced in GLUE, but despite the high performance on the benchmark itself it remained hard to solve. The diagnostic set was therefore kept unchanged in SUPERGLUE. This indicates that the NLI format might

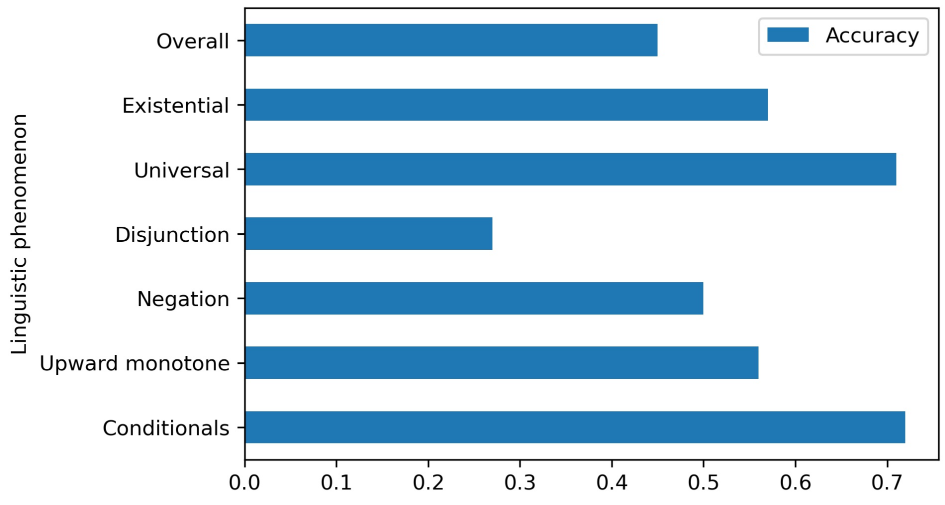


Figure 2.10: An extract of an example evaluation of BERT fine-tuned on the MultiNLI agianst the diagnostic set of GLUE

be suited for more general evaluation, not only as an isolated downstream task. This will be more thoroughly discussed in the next section.

NLI as a diagnostic tool for NLU

Where the NLP models of the past relied on feature engineering and pre-defined rules, contemporary models often rely on pre-trained encoders to represent the meaning of the target text in vector space, making interpretability harder. Answers to questions like: Does my model handle negation and co-reference? How does different linguistic quantifiers affect performance? Why did it fail on this subset? are not easily available.

The most popular way of assessing these models have consequently been to create benchmarks, like GLUE and SUPERGLUE (A. Wang, Pruksachatkun et al., 2019; A. Wang, A. Singh et al., 2018). As mentioned, these test suites - with their publicly available leaderboards – evaluates how different models perform on a variety of NLP tasks, such as question answering, semantic similarity and NLI.

One might argue that this has moved the focus from assessment of language capabilities over to pure performance – symbolized by an accuracy score ranking which university groups and large corporations alike fight to reign. To mediate this, the aforementioned diagnostic set assesses how well a model handles different linguistic phenomena, not just an overall accuracy. By running a set of hand-crafted examples up against an NLI classifier, the goal of the diagnostic phase is to assess sentence understanding, not displayed performance. The sentences are manually annotated with the linguistic phenomena that are in play. Thus, NLI becomes not just another benchmark but also an important analysis tool.

In a survey paper looking at the overview of different approaches for evaluating and understanding the reasoning capacities of current NLP systems, Poliak (2020) concludes that the NLP community should use such diagnostic tools more rigorously. This call from Poliak emphasizes that it is not only models trained specifically on NLI training data that should be diagnosed on such sets: All contemporary language models could benefit from such an analysis. This can easily be done by using an auxiliary NLI classifier on top of the model at hand to run against a diagnostic set. This method can be used to evaluate text generation systems (Falke et al., 2019), machine translators (Poliak, Belinkov et al., 2018), models for predicting discourse markers (Kim et al., 2019) and surely many more. That is to say, there is something about the inference task that makes it especially apt for evaluating language understanding capabilities, and thus also the world knowledge and commonsense reasoning abilities that we try to inject into our models using adapter modules.

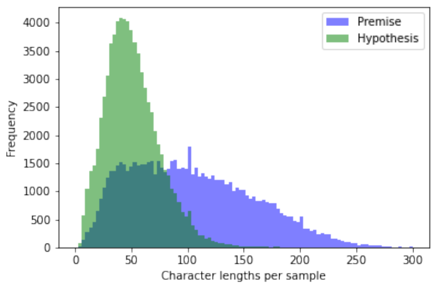
Figure 2.10 illustrate how one language model, BERT (Devlin et al., 2019), performs on one fine-grained cateogry from the GLUE diagnostic set. The model is first fine-tuned on 40% of the MNLI, with a learning rate of $4e-5$ and batch size 16. On the original MNLI test set, the model achieves an overall accuracy of $.\tilde{8}0$. This is a stark contrast to the overall achieved accuracy of .46 on the diagnostic set. As the figure illustrates, the model infer well from some linguistic phenomena, such as universal quantifiers, much like what is communicated in example 2.15, but handles disjunction poorly – only predicting the correct entailing relationship in .28 of the examples. For a full analysis, using a three-class generalization of the Matthews correlation coefficient for more accurate evaluation, see the reproduction study in chapter 3.

This goes to show that high overall performance on traditional held out test sets, which has been put forth as examples of definitive progress, might shade challenges in deep learning models, and that we must be careful about how we use the term *understanding* (Bender and Koller, 2020). This observation is key to how we reason about our experiments in chapter 4.

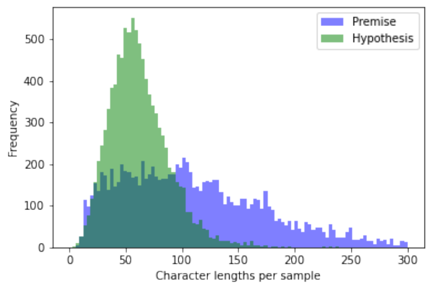
Challenges in NLI

As with any other machine learning task, one might question whether or not the models we train acquire the knowledge we intend or if their performance is influenced by confounding variables and statistical irregularities – which often manifest itself as some form of linguistic pattern.

For the task of NLI, where inference is based on assessing the relationship between a premise and a hypothesis, one might think that ignoring the premise and only conditioning a model on the hypothesis is a futile effort. However, hypothesis-only baselines, such as the one by Poliak, Naradowsky et al. (2018), are able to outperform a majority-class classifier. This suggests that the popular datasets, such as the SNLI and MNLI, contain linguistic biases that create statistical irregularities.



(a) XNLI test set



(b) MultiNLI dev mismatched set

Figure 2.11: The character lengths of the premises and hypotheses in XNLI and MNLI

This behavior might be explained by specific wording patterns, such as a more frequent use of negation terms for the label *contradiction* or general vagueness. There is also a difference in sentence length for the premises and hypothesis in popular datasets such as MNLI (Williams, Nangia and S. Bowman, 2018) and XNLI (Conneau et al., 2018) – as illustrated in Figure 2.11 (Gururangan et al., 2018; Poliak, Naradowsky et al., 2018). Other examples of annotation artifacts include the tendency of entailed hypotheses to contain gender-neutral references to people and the usage of purpose clauses in neutral samples (Nie et al., 2020).

The existence of such annotation artifacts could be explained by the procedure used to create the SNLI and MNLI. The crowd-sourced annotators unintentionally leave linguistic clues that make it possible for hypothesis-only baselines to perform much better than expected (Gururangan et al., 2018). These claims are also supported in Glockner, Shwartz and Goldberg (2018), where a small, simple dataset intended to capture various lexical knowledge breaks many state-of-the-art NLU systems, illustrating that such models fail to generalize, and even more so: fail to make simple inferences.

2.5.2 Grounded commonsense inference

A limitation of the aforementioned NLI datasets is their primary focus on linguistic entailment, as seen in example 2.14. Inference from text in day-to-day life, on the other hand, necessitates understanding about everyday situations, such as the physical properties of objects. In order to cater to this and better evaluate a language models abilities, work by Zellers, Bisk et al. (2018) introduce the the task of grounded commonsense inference — an attempt at unifying NLI and commonsense reasoning.

Datasets

Swag. The most popular resource for this task, also introduced in Zellers, Bisk et al. (2018), is SWAG — consisting of 113k multiple choice questions about a variety of grounded situations. The samples in SWAG originate from pairs of temporally adjacent video captions, where the first element serves as the context for a follow-up scenario. The samples were selected using adversarial filtering, a novel approach introduced with the dataset. Details on this filtering mechanism can be found in Zellers, Bisk et al. (2018).

For our purposes, it suffice to provide a definition on what it means for a dataset to be adversarial. As defined in Zellers, Bisk et al. (2018, p.3), "we say that an adversarial dataset for a model f is one on which f will not generalize, even if evaluated on test data from the same distribution". The motivation of such a process, of course, is that the procedure will help remove annotation artifacts which in turn will make the dataset more challenging for state-of-the-art systems. Although the effect of adversarial filtering as a means of addressing such artifacts has been questioned (S. R. Bowman and Dahl, 2021), the results from Zellers, Bisk et al. (2018) at least indicate that top NLI models struggle more on SWAG than on normal NLI datasets when compared to humans.

The format of the task itself is based on four follow-up scenarios to some context, three of which are unlikely to happen or are counterfactual. The objective for the language model is to choose the correct/most likely follow-up. For example:

(2.16) A girl is going across a set of monkey bars. She

- jumps up across the monkey bars.
- struggles onto the monkey bars to grab her head.
- **gets to the end and stands on a wooden plank.**
- jumps up and does a back flip.

(2.17) People are walking next to the camels leading them. A building

- is shown riding the camels.
- **is shown in the background.**
- with a rifle is leading them.
- is then shown for several clips.

HellaSwag. Examples like the one in 2.5.2 are not easy. However, the introduction of pre-trained language models like BERT pushed the results up to near human-level performance on SWAG. Building on the same ideas, work by Zellers, Holtzman et al. (2019) uses the same adversarial filtering mechanism to create an even more challenging dataset for the task of grounded commonsense inference. This dataset, named HELLASWAG, provides valuable insight into the commonsense abilities of pre-trained language models.

The conclusion from Zellers, Holtzman et al. (2019) is that such models do not demonstrate robust commonsense reasoning, but rather rely on dataset-specific biases. This might partly explain why the models that predate the architecture used in BERT struggled on SWAG, while current systems are above human-level performance. Yes, they do have more commonsense knowledge as a result of their pre-training, but not enough to account for such a large improvement. Despite the adversarial filtering, they nevertheless exploit artifacts, as we also discussed for NLI (see section 2.5.1).

Challenges and use case

Despite these flaws, we use SWAG as one of our evaluation datasets in chapter 4. We do this so that we can compare our knowledge injected models with non-injected language models that use the traditional fine-tuning approach, such as a base instance of BERT or RoBERTa. What we hope to achieve with this comparison, is some insight into whether or not the injection of commonsense information, in our case from ConceptNet (Speer, Chin and Havasi, 2017), improves performance on the grounded commonsense inference task, or if the injection just interfere with the distributional knowledge acquired during the base models initial pre-training — which might be more important for performance if these models are indeed *rapid surface learners*, as indicated by Zellers, Holtzman et al. (2019). Consequently, motivated by **RQ2**, we are not interested in performance competitive with the top-performing systems — the current state-of-the-art result is at 91.71%, but whether or not they perform on par with non-injected models of the same architecture type.

2.5.3 Question answering

When answering a question, we frequently make use of knowledge external to the question itself. This need for commonsense reasoning and world knowledge makes question answering suitable for evaluating these capabilities in language models. This has motivated the creation of multiple question answering datasets, many of which are targeted at some specific domain knowledge, for example within science (Clark et al., 2018; Tang et al., 2020). Some, as the two presented below, target commonsense reasoning specifically, making them especially apt for the evaluation of the models presented in this work.

Datasets

CommonsenseQA. One of the issues with previous question answering sets that target commonsense reasoning and world knowledge is the overall sample size. The Choice of Plausible Alternatives (COPA) (Roemmele, Bejan and Gordon, 2011), for example, contains only 1000 question answer

Predicate type	Question	%
`atlocation`	Where would I not want a fox? **A.** hen house, B. england, C. mountains, D. ...	47.3
`causes`	What is the hopeful result of going to see a play? **A**. being entertained, B. meet, C. sit, D. ...	17.3
`hasProperty`	What is a reason to pay your television bill? **A**. legal, B. obsolete, C. entertaining , D. ...	1.2

Table 2.4: An excerpt of some of the questions from Talmor et al. (2019), together with the predicate for the subgraph used to source the questions, and the frequency of that type in the dataset. The correct answer is marked in bold text.

pairs, split evenly between train and test partitions. To meet the need for large scale set targeted at commonsense, work by Talmor et al. (2019) introduced the COMMONSENSEQA resource — a dataset consisting of 12,247 commonsense questions collected through the use of crowdsourcing. Table 2.4 shows some of these questions.

The dataset is based on ConceptNet (Speer, Chin and Havasi, 2017), which is also the source material for the knowledge injected into the models presented in this work. As explained in section 2.1.1, ConceptNet contains *concepts*, such as `Car`, `Stockholm` and `Book`, and *predicates* that can tie concepts together, such as `causes`, `isA` and `atLocation`. The authors extracted subgraphs based on predicate type, each containing one source concept and three target concepts. Crowdsourcing workers were then asked to write three questions per subgraph (one for each target concept).

Challenges and use case

As COMMONSENSEQA is constructed from the same knowledge source as the data we inject our models with in chapter 4, one should expect to see some performance gain on this benchmark, as outlined in **RQ2**. Table 2.5 shows four general question formats and the commonsense knowledge category that Talmor et al. (2019) deem necessary in order to answer the associated question correctly, from a human perspective. However, as is the issue with SWAG, it is not clear at all whether or not those categories of commonsense knowledge are actually required in order to choose the

Category	Definition	%
`Spatial`	Concept A appears near Concept B	41
`Cause & Effect`	Concept A causes Concept B	23
`Has parts`	Concept A contains Concept B as one of its parts	23
`social`	It is a social convention that Concept A 15 correlates with Concept B	15

Table 2.5: Commonsense knowledge categories from Talmor et al. (2019), their definition and their frequency in the dataset.

correct answers. It might be the case that statistical irregularities and artifacts prove more useful for solving the task from a computational perspective.

Despite this criticism, we choose to evaluate our models on COMMONSENSEQA. As our injected models are compared to non-injected models of the same root architectures, BERT and ROBERTA, we argue that our models will exploit the same artefacts, if any, and that the injected knowledge should nonetheless increase the performance since many of the same concepts will be seen both during adapter training and fine-tuning and evaluation on COMMONSENSEQA.

2.6 Summary

This chapter presents the problem of world knowledge and commonsense in text. By using structures like knowledge graphs, we can represent the relation between concepts, as is done in ConceptNet, and create a hierarchical structure. Doing this allows language models based on neural networks to encode structured information using learning objectives like masked language modeling. Instead of fine-tuning entire models on this task, we can create light-weight neural networks, called adapters, and use those to model the information and inject them into larger pre-trained models. In order to evaluate whether or not this information increases the world knowledge and commonsense reasoning abilities of the injected models, we can evaluate their performance on popular NLP datasets belived to require the same knowledge, such as SWAG (Zellers, Bisk et al., 2018) and COMMONSENSEQA (Talmor et al., 2019).

Combined, such a pipeline explores the more general problem of combining structured information with neural networks trained in an unsupervised or self-supervised manner, using adapters as the intermediate structure. The next chapter studies the feasibility of this approach by reproducing and analyzing related work by Lauscher et al. (2020).

Chapter 3

A Reproduction Study of Retrograph

This chapter reproduces a previous attempt at adapter based knowledge injection by Lauscher et al. (2020). The first section presents the key components of the study, name Retrograph [1], while the second section present experimental results together with an error analysis. The last section discusses the results and possible directions for future research. By doing this, we hope to shed light on the effectiveness of adapters as a method for combining structured information with language models, as outlined in **RQ1** (see point 1.2 in chapter 1).

3.1 Overview

The aim of the Retrograph study is to investigate one approach to complementing the distributional knowledge in BERT (Devlin et al., 2019). Specifically, the authors extract information from the ConceptNet (Speer, Chin and Havasi, 2017) knowledge graph and the Open Mind Common Sense corpus (P. Singh et al., 2002) and inject it into a pre-trained BERT model through the use of single-task adapters. The adapters are trained using the masked language modeling objective and inserted into each encoder layer twice, as in Houlsby et al. (2019).

Since the information in ConceptNet is stored as triples, the authors use a random walk procedure to extract and translate these triples into natural language. Using this procedure, the triples `(alcholism, causes, stigma)`, `(stigma, hasContext, christianity)`, `(christianity, partOf, religion)` are transformed to the sentences: *alcoholism causes stigma. stigma is used in the context of christianity. christianity is part of religion.* For more details on ConceptNet, see section 2.1.1.

The models are evaluated against the full GLUE (A. Wang, A. Singh et al., 2018) benchmark and compared to the standard uncased base version of BERT, which as 12 transformer layers and a hidden layer size of 768. The authors find the overall result of their approach on these tasks to be

[1] https://github.com/Wluper/Retrograph

inconclusive. However, they outperform BERT with a substantial amount on some categories in the diagnostic set (see section 2.5.1). These categories include samples on the NLI format that are hand-crafted to demonstrate world knowledge and common sense; the same kind of information believed to be captured by ConceptNet.

3.1.1 Random walk

Retrograph uses the weighted random walk algorithm from NODE2VEC (Grover and Leskovec, 2016) in order to extract the `subject--predicate--object` triples from ConceptNet,. The pseudocode from the original publication on this algorithm is presented below. The alias method refers to a popular way of sampling from a discrete probability distribution.[2]

Algorithm 1 The random walk procedure from Lauscher et al. (2020)

```
1: procedure NODE2VECWALK(Graph G′ = (V, E, $\pi$), Start node $u$, Length $l$)
2:     Inititalize walk to [u]
3:     for walk_iter = 1 to $l$ do
4:         curr = walk[-1]
5:         $V_{curr} = GetNeighbors(curr, G')$
6:         $s = AliasSample(V_{curr}, \pi)$
7:         Append $s$ to $walk$
       return walk
```

In Retrograph, for each node, the authors do two walks with a length of 15. In total, this gives 2,268,485 walks, inducing a corpus of 34,560,307 synthetic sentences.

3.1.2 Adapters

Using the architecture from Houlsby et al. (2019), the authors introduce an adapter module in every transformer layer in BERT. Each adapter consists of two feed-forward neural networks and a residual connection between these, as described in section 2.3.1. They set the dimensionality of the bottleneck down-projection to 64, ensuring that the additional parameters added to the transformer is kept low. The non-linear activation in the adapter module is done with GELU (Hendrycks and Gimpel, 2016).

The weights of the adapters are initialized at random and then adjusted using the Adam optimizer (Kingma and Ba, 2015). The natural language sentences from the random walk are fed into a base BERT model set on the masked language modeling objective, as in the original pre-training of the model. Only the weights in the adapters are adjusted, keeping the distributional information from pre-training intact. See chapter 2.3 for a more detailed explanation and illustrations of the architecture.

[2] https://lips.cs.princeton.edu/the-alias-method-efficient-sampling-with-many-discrete-outcomes/

3.2 Reproduction

3.2.1 Experimental setup

All the reported experiments are run on NVIDIA P100 graphics cards, using the code provided by the authors. For replicability, we do not train the adapters from scratch, relying instead on already trained adapters downloaded directly from a s3 bucket provided by the authors. The pre-trained BERT model is also downloaded directly from this folder. We also note that the this model is not an instance of the now popular `transformers` library (Wolf et al., 2019), but rather based on the original implementation from Devlin et al. (2019).

We report results for three models: $\text{BERT}_{\text{BASE}}$, CN-100K and CN-150K. The numbers 100k and 150k refer to the total number of update steps that the adapters are allowed to complete during training. That is, during adapter training the duration of the process is determined by update steps and not by the more common epoch measure. All models are uncased, as the extracted corpus does not contain any capital letters. It is primarily the CN-100K model that are of interest in this reproduction, as it is the CN-100K that the authors report as achieving best results on the most relevant sections of the evaluation.

The models are evaluated against four datasets: the whole diagnostic set from GLUE (A. Wang, A. Singh et al., 2018), the test sets from MNLI (Williams, Nangia and S. Bowman, 2018) and two fine-grained subcategories of the diagnostic set annotated for world knowledge and common sense. All models in this reproduction fine-tune on the entire training set of the MNLI before evaluation, consisting of close to 400k samples, as this is the downstream format of the diagnostic set as well. Hence, the whole training procedure has two steps: 1) to train only the adapter parameters on the extracted triples from ConceptNet and 2) to fine-tune all parameters on the downstream task.

As in the original Retrograph paper, the downstream system is fine-tuned using a grid search in the following space of hyperparameters: learning rate $\in \{2e-5, 3e-5\}$, epochs $\in \{3, 4\}$, with a batch size of 16 and a maximum length of 128 tokens for input sequences.

The test set of the MNLI is split into a matched and mismatched partition, indicating whether or not the samples are cross-domain, as the sentences are sampled from various media types. The evaluation against the test sets are done on the official web page of the GLUE benchmark[3] as the gold labels are kept private. All other scores are calculated by us. Known discrepancies compared to the original Retrograph study:

- The downstream fine-tuning script from the Retrograph code base does not set any seed.
- The original study uses a hyperparameter grid search but does not report which setting caused the best results. For us, a learning rate of *3e-5* for three epochs performs best for all models.

[3] https://gluebenchmark.com/

Model	MNLI-m	MNLI-mm
$\text{BERT}_{\text{BASE}}$ (Lauscher et al., 2020)	84.6	83.4
CN-100K	84.0	82.8

Table 3.1: Accuracy scores on test sets of MNLI. *m* and *mm* refers to the matched and mismatched sections of the test set respectively.

Model	Mean & std.deviation	Lauscher et al. (2020)
$\text{BERT}_{\text{BASE}}$	$\mu = 36.2, \sigma = 0.009$	34.2
CN-100K	$\mu = 35.4, \sigma = 0.006$	37.8

Table 3.2: Mean MCC scores over three runs on the full diagnostic set from GLUE ($N = 1104$) and the single reported score from Retrograph.

- We had to remove a call to the tensorflow method `dataset.repeat()` as it caused a never ending loop.

NOTE: Since the sample sizes are so small, the fact that the original code does not set a seed for the fine-tuning process is problematic and greatly limits what can be said about the results in comparison to the original paper. To mediate this effect, we run the best configuration of hyperparameters three times and report the mean and standard deviation for CN-100K and $\text{BERT}_{\text{BASE}}$ in addition to the best performance score.

3.2.2 Matthews correlation coefficient

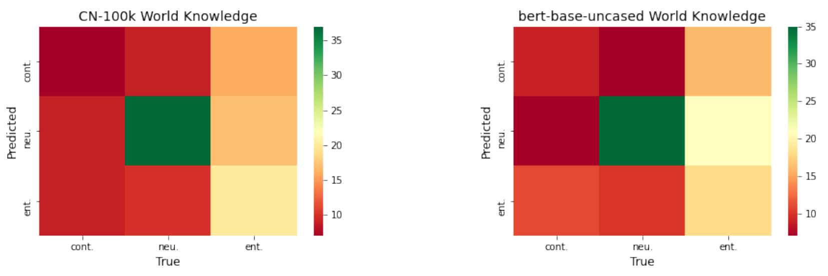
Due to unbalance in the label distribution, the results on the diagnostic set are commonly measured with a three-way generalization of Matthews correlation coefficient, MCC. This measure takes into account true or false positives and negatives even if the classes are unbalanced. The MCC returns results in the range [-1,1], where 1 denotes a perfect classification. For the binary case, the MCC is defined as:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(RN+FP)(TN+FN)}} \tag{3.1}$$

The results in this paper are calculated using the MCC implementation from `scikit-learn`.[4] For a more detailed explanation of the MCC, see Jurman, Riccadonna and Furlanello (2012). The scores for the MNLI are normal accuracy points.

Model	**Mean & std.deviation**	**Lauscher et al. (2020)**
BERT$_{\text{BASE}}$	$\mu = 16.4, \sigma = 0.1210$	10.3
CN-100k	$\mu = 17.8, \sigma = 0.0155$	25.6

Table 3.3: Mean MCC scores over three runs on the world knowledge category from the diagnostic set in GLUE and the single reported score from Retrograph, $N = 134$



(a) CN-100K (b) BERT$_{\text{BASE}}$

Figure 3.1: A confusion matrix on world knowledge samples for CN-100K and BERT$_{\text{BASE}}$

3.2.3 Results

Lauscher et al. (2020) conclude that the overall performance on GLUE are inconclusive. However, as they receive a difference in performance of up to 20 MCC points compared to a non-injected BERT on some categories on the diagnostic set, they conclude that the knowledge injection using adapters as the intermediate structure between the underlying language model and ConceptNet is effective.

As the main contenders are CN-100K and BERT$_{\text{BASE}}$, we argue that the higher the difference are between these two, in favor of the former, the more likely it is that the adapter injection has an effect. Furthermore, we stress that this difference must also be considered in relation to the sample size of the test categories.

Table 3.2 shows that our experiments are not consistent with the findings reported in the Retrograph paper on the full diagnostic set. None of the models reach the target score — most likely due to the issue of no seeds — and the difference between CN-100K and BERT$_{\text{BASE}}$ is less than in the original paper. On the full MNLI set, see Table 3.1, the results are more consistent. This is probably due to the size of the test sets, of over 9000 in both the matched and mismatched partitions.

For samples demonstrating world knowledge, Table 3.3 reveals that

[4]sklearn.metrics.matthews_corrcoef

Model	Mean & std.deviation	Lauscher et al. (2020)
$\text{BERT}_{\text{BASE}}$	$\mu = 27.0, \sigma = 0.0021$	29.0
CN-100k	$\mu = 30.2, \sigma = 0.0238$	24.4

Table 3.4: Mean MCC scores over three runs on the common sense category from the diagnostic set in GLUE, and the single reported score from Retrograph, $N = 150$

the difference in performance between the mean scores of CN-100K and $\text{BERT}_{\text{BASE}}$ is 1.4 MCC points. In comparison, Lauscher et al. (2020) reports a difference of 15.6 on the same test. With a set size of 134, this means that the highest performing version of the CN-100K classifies 11 instances correctly that the best version of $\text{BERT}_{\text{BASE}}$ does not. Conversely, $\text{BERT}_{\text{BASE}}$ classifies 9 correctly that CN-100K does not. See Table 3.5 for some specific examples. Hence, CN-100K does not perform consistently better at this category. If we compare the confusion matrix for each model on this category, see Figure 3.1, we see that "contradiction" is the most difficult label to classify for both, and that in general, the two models classify quite similarly. This is further supported by the low standard deviations for both set of experiments.

For the commonsense category, we get the inverse difference compared to Retrograph: CN-100K performs better than $\text{BERT}_{\text{BASE}}$, albeit not by much. This is the opposite of the effect reported in Lauscher et al. (2020).

3.2.4 Discussion

The experiments outlined in this reproduction have all failed to replicate the results in Lauscher et al. (2020). For the large MNLI dataset the difference between our experiments and the reported results in the original paper is not that large, but for the diagnostic sets, where the sample size is considerably lower, we get something quite different. One would expect this behavior due to the fact that these experiments did not run on the same seed as Retrograph, making the fine-tuning non-deterministic. As mentioned, we reason that this effect scales with the sample size, which might help explain why the difference between our experiments and theirs differ more for the smaller categories. That being said, the mean MCC score over three runs should still give some indication on the models performance in a way that is statistically valid.

On the world knowledge category, the difference between CN-100K and $\text{BERT}_{\text{BASE}}$ is minimal, or if there is an effect, we argue that it cannot be attributed to the successful injection of knowledge from ConceptNet alone , as the sample size of the test sets are too low (N=134). Furthermore, the number of samples that one model classify correctly and the other does not is close to equal: the number of instances CN-100K classify correctly but $\text{BERT}_{\text{BASE}}$ does not is similar to the number of cases $\text{BERT}_{\text{BASE}}$

ID	Premise	Hypothesis	Gold label	CN100k	bert-base-uncased
960	The reaction was strongly endothermic	The reaction media got very hot.	entailment	entailment	neutral
593	The reaction was strongly exothermic.	The reaction media got very cold	cont.	cont.	neutral
671	Fun fact, that guy in the Ireland jacket is on Saturday Night Live now..	Fun fact, that guy in the Ireland jacket is on SNL now.	entailment	entailment	cont.

s

Table 3.5: A comparison of three out of the eleven total samples in the world knowledge category that the CN100K model classified correctly but $\text{BERT}_{\text{BASE}}$ did not.

classify correctly but CN-100K does not. As the MCC score on the world knowledge samples in particular are so low to begin with (around 0.17), this points towards a somewhat arbitrary difference in classification between the models. The same goes for the commonsense category.

To conclude: The effect of adapter-based knowledge injection is still inconclusive and should be explored further. The discrepancy between the results outlined here and the ones by Lauscher et al. (2020) provides evidence towards the inconclusiveness of the extent to which the relations from ConceptNet had any significant effect of the classification capabilities of BERT. We argue that the discrepancy is also too large to be explained solely by the unequal seeds between the experiments. As the code and models were identical, this points to problems in either the adapter injection method itself, or the methods of evaluation.

These findings motivate more extensive experimentation with different combinations of pre-trained language models and adapters, as well as a deeper analysis of how the relations from ConceptNet influence the underlying language model when faced with other datasets. The next chapter attempts to meet some of these challenges.

Chapter 4

Injecting Knowledge Using Adapters

Building on the work of Lauscher et al. (2020) and R. Wang et al. (2021), this chapter studies the problem of injecting common sense and world knowledge into large pre-trained language models. We evaluate different approaches to solving this on a range of downstream NLP tasks believed to require commonsense reasoning abilities and world knowledge. Furthermore, we conduct probing experiments in order to explore to which extent our models are able to reproduce facts seen during training. The methods in this chapter all rely on adapter modules as they are presented in section 2.3.

4.1 Adapters as an injection method

As presented in section 2.3, adapters (Houlsby et al., 2019) mitigate the need for full scale fine-tuning by allowing for efficient parameterization. This is done by introducing lightweight neural networks into larger models and only adjusting the weights of this smaller network on the downstream task. Adapters can then be shared by only transferring the weights of the lightweight adapter instead of the entire language model, for example with the AdapterHub network (Pfeiffer, Rücklé et al., 2020).

Hence, adapters are first and foremost an efficient technique for transfer learning. Using frameworks, such as AdapterHub, we can easily see if adding an adapter module trained on dependency parsing yields any performance gain when combined into an inference pipeline.

In this work, however, we do not study the use of adapters in the conventional sense; we study the case where adapters are not trained on any traditional downstream task, but rather used as a tool for knowledge representation. By encoding structured information from knowledge graphs into adapter modules and then injecting them into language models, we hope to determine if adapters are a viable solution to the long-standing problem of combining structured information and models that are primarily trained in an unsupervised way. Although still operating within the domain of transfer learning, we argue that adapters are a somewhat unexplored technique to this end.

4.1.1 Experiments

4.1.2 Datasets

Below is a short presentation of the datasets used to evaluate the models in this chapter. A more thorough explanation of all resources can be found in section 2.5.

GLUE Diagnostic The diagnostic set from GLUE (A. Wang, A. Singh et al., 2018) is a manually curated NLI dataset, with examples annotated with the type of knowledge believed to be required in order to classify the correct entailment relation between two sentences. For a more detailed explanation of the NLI task, see section 2.5.1.

SWAG This benchmark, introduced in Zellers, Bisk et al. (2018), evaluates language models on the task of grounded commonsense inference. Using a multiple choice format, the task is to judge from a range of options which scenario is the most likely follow-up from a provided grounded scenario. For example, from the scenario "*On stage, a woman takes a seat at the piano. She*" the models must choose between the follow-ups: (*"sits on a bench as her sister plays with the doll.", "smiles with someone as the music plays.", "is in the crowd, watching the dancers.",* ***"nervously sets her fingers on the keys."***). The dataset was created using adversarial filtering. For a more detailed explanation of the task, see section 2.5.2.

Commonsense QA Introduced in Talmor et al. (2019), the COMMONSENSE QA contains 12,247 multiple choice questions that are sourced from ConceptNet (Speer, Chin and Havasi, 2017). The questions draw upon world knowledge external to the particular context of the question. In order for the model to choose the correct answer, it must therefore use existing knowledge. For example, for the question *I'm crossing the river, my feet are wet but my body is dry, where am I?* the models must choose between the following set of possible options, of which only one is correct: (*waterfall, bridge,* ***valley***, *bank, island*). For a more detailed explanation of the task, see section 2.5.3.

4.1.3 Experimental setup

All experiments ran on a high performance computing resource made available by the University of Oslo. The models were implemented using the PyTorch version of the Huggingface Transformers library (Wolf et al., 2019). Unless another implementation is specifically specified, the adapters were implemented using the *adapter-transformer* library (Pfeiffer, Rücklé et al., 2020). The models trained and were evaluated on a single NVIDIA A100 GPU node.

4.1.4 Baselines

As baselines, we compare our results with primarily two pre-trained language models: the BERT$_{\text{BASE}}$ from Devlin et al. (2019) and the ROBERTA$_{\text{BASE}}$ from Y. Liu et al. (2019). These models are two of the predominant language models in use today for almost all NLP tasks. We also use these model as the root models for adapter injection.

The Houlsby adapter

For the diagnostic task we also consider the model configuration from the Retrograph study (see chapter 3) as a baseline. To our knowledge, this configuration, which can be traced back to work by Houlsby et al. (2019), has not been deployed on the SWAG and COMMONSENSE QA datasets. It is therefore a contribution of this work to evaluate such a configuration on these sets for the first time.

Following Houlsby et al. (2019), we set the size of the adapter modules to 64. This implies a reduction factor of 12 from the original transformer layer size in BERT$_{\text{BASE}}$. We use GELU (Hendrycks and Gimpel, 2016) as the adapter function *f*. Gradients are calculated and updated using the Adam optimizer (Kingma and Ba, 2015). We set the learning rate to *1e-4* with 10000 warm-up steps and weight decay factor of 0.01. We allow the adapter to train for 100000 optimization steps while freezing all the original transformer weights. The resulting adapter module is referred to as CN$_{\text{HOULSBY 100K}}$ throughout the remainder of this chapter.

Training the adapter

The adapter model referred to as CN$_{\text{HOULSBY 100K}}$ is trained on the same subset of ConceptNet (Speer, Chin and Havasi, 2017) as in Lauscher et al. (2020). We use the same corpus file generated from their random walk procedure (see section 3.1.1). To our knowledge, judging from their source code [1], this corpus is built from four predicate types: ANTONYMOF, SYNONYMOF, ISA and MANNEROF. These predicates are extracted randomly through a tree traversal and then subsequently chained so that we get blocks of text in natural language on the following format:

(4.1) possible is a synonym of possibility.
possibility is a concept.
concept is a synonym of conception.
conception is a synonym of fertilization.
fertilization is a enrichment.
enrichment is a gift.
gift is a synonym of douceur.

Lauscher et al. (2020) does not motivate the selection of these predicates in particular. We experiment with different sets of predicates later in this

[1] https://github.com/Wluper/Retrograph

Model	Adapter	Swag	Δ	CSQA	Δ
BERT$_{\text{BASE}}$		81.05		61.17	
ROBERTA$_{\text{BASE}}$		83.90		60.36	
BERT$_{\text{BASE}}$	CN$_{\text{HOULSBY 100K}}$	80.08	−0.97	57.41	−3.76
ROBERTA$_{\text{BASE}}$	CN$_{\text{HOULSBY 100K}}$	83.34	−0.56	56.26	−4.10

Table 4.1: Results (Accuracy) on SWAG (Zellers, Bisk et al., 2018) and CSQA (Talmor et al., 2019)

Model	Adapter	Full	CS	World	NE
BERT$_{\text{BASE}}$		37.15	32.74	13.36	27.69
ROBERTA$_{\text{BASE}}$		41.18	37.08	14.98	**36.92**
BERT$_{\text{BASE}}$	CN$_{\text{HOULSBY 100K}}$	36.76	29.31	**20.09**	34.86
ROBERTA$_{\text{BASE}}$	CN$_{\text{HOULSBY 100K}}$	**41.36**	**39.33**	12.71	29.19

Table 4.2: Results (Matthews correlation coefficient, see section 3.2.2) on the full set and the Common Sense (CS), World Knowledge (World) and Named Entities (NE) categories of the GLUE diagnostic (A. Wang, A. Singh et al., 2018)

chapter. The corpus is processed using MLM, parsed line for line with a MLM probability of 0.15 as in the original BERT paper (Devlin et al., 2019). We also experiment by training on the corpus by a maximum sequence length instead of line by line training. However, this did not affect the performance of the models in any significant way.

4.1.5 Results and discussion

Table 4.1 and 4.2 show the results on the aforementioned datasets. For all downstream tasks, the models are fine-tuned for 3 epochs with a learning rate of 3E-5 and batch size of 16. These hyperparameters were found to be best across all tasks and all models from the following grid search:

LR $\in \{3e-5, 4e-5, 5e-5\}$
EPOCH = 3
BATCH $\in \{16, 32\}$
SEED 42

Table 4.1 shows that the injection of the CN$_{\text{HOULSBY 100K}}$ adapter had little effect on the performance of both BERT$_{\text{BASE}}$ and ROBERTA$_{\text{BASE}}$ on the SWAG task. With a drop in performance of approximately a whole and a half point respectively, the results indicate that the injection at best only slightly impaired the overall performance. However, the deltas might be within the standard deviation of the original models if we run the same experiments multiple times with different seeds. Due to the time complexity of the overall experimental pipeline, however, this is not feasible.

For the COMMONSENSE QA set performance dropped significantly. As the triples from ConceptNet are extracted and translated into natural language in a rigid way, not at all like the free form we find in longer fragments of text, such as news article or novels, we can assume that the form of the multiple choice questions provide a context that makes it difficult to apply the knowledge encoded in the adapters. However, the five possible answers to the questions are all short in length, often just a single token, which intuitively should be easier to classify than the full sentence alternatives in SWAG.

For the diagnostic set from GLUE (A. Wang, A. Singh et al., 2018), our experiments give similar results as the Retrograph study (Lauscher et al., 2020) presented in chapter 3. Since the dataset and the fine-grained categories are of a low sample size, a high variance is expected. The key finding is that the $\text{CN}_{\text{HOULSBY 100K}}$ gives the highest Matthews correlation coefficient for the world knowledge category when applied to a BERT base model. For ROBERTA we observe no effect. We also remark that as in the reproduction study, we get notably higher result for the baseline, the untouched BERT base model, than what is reported in Lauscher et al. (2020).

The results show that the injection of triples from ConceptNet does not improve the performance of language models on the SWAG and COMMONSENSE QA tasks, at least not with the $\text{CN}_{\text{HOULSBY 100K}}$ configuration. Furthermore, due to low sample sizes, the effect observed over the diagnostic set from GLUE is uncertain, although it looks promising for the world knowledge subcategory.

This is not to say, however, that the injection was unsuccessful. The next section presents a series of probing experiments, aimed at inspecting whether or not the specific knowledge encoded into the adapter modules influence the predictions in a more controlled environment.

4.2 Probes and analysis

One of the aims of this work is to study whether or not the combination of masked language modeling and adapters is a feasible method for combining structured information and pre-trained language models. For this to be the case, we argue that the adapter-injected models must be able to use the knowledge gained from the adapter training together with what the models learned during their initial pre-training.

As the parameter spaces of pre-trained models are considerable in size, one major issue of using adapters in the manner presented in this work and in Lauscher et al. (2020) is that the modules are too lightweight: their contribution to the final prediction is a function of their size, proportionate to the size of the overall model. If we add an adapter with the Houlsby configuration to a pre-trained BERT base model, the adapter makes up 2.1% of the total parameters.

During inference, then, the knowledge from the graph, in our case ConceptNet, has considerable less influence on the final prediction — and much of the information extracted using the random walk procedure might be low frequency concepts, not at all present in downstream tasks, such as SWAG and COMMONSENSE QA.

Hence, in order to gauge the effect of the method itself, we present here a series of probing experiments following the work of Petroni et al. (2019) and Elazar et al. (2021).

4.2.1 LAMA

The LAMA (LAnguage Model Analysis) probe (Petroni et al., 2019) allows us to examine the factual and commonsense knowledge present in language models. Facts are presented as statements containing a subject, predicate and an object, where the object is masked. For example, the factual knowledge that the author Henrik Ibsen was born in Skien in 1828 is presented as: "Ibsen was born in [MASK] in the year 1828". Following Jiang et al. (2020), we affirm the existence of a fact in the language model if the model can predict the object underneath the mask:

(4.2)
$$\hat{y} = \underset{y' \in V}{\arg\max}\, P_{LM}\left(y'|x, t_r\right),$$

where V is the vocabulary of the model and $P_{LM}(y'|x, t_r)$ is the probability that the model predicts the correct object given the other tokens in the statement (the subject and the predicate). LAMA is built from multiple knowledge sources, one of which is ConceptNet (Speer, Chin and Havasi, 2017). A list of all the predicate types can be found in appendix A.2.

We train an adapter with the Houlsby configuration on the extracted ConceptNet corpus and also evaluate on the ConceptNet split from LAMA. Consequently, what we test here is not the model's ability to generalize on unseen data, but whether or not it is able to reproduce factual information extracted from the knowledge graph during adapter training and apply it on Cloze-style statements that require the same knowledge. Since both the adapter training and the probe uses the masked language modeling classification task (see section 2.2.4), this happens under a zero-shot setting on different phrasings of the same `subject-predicate-object` triples. For example, one sentence in LAMA derived from the triple `communicating hasSubevent knowledge` is presented in the probing dataset as *Communicating is for gaining* $[MASK]$, while the same triple would be phrased as *communication has subevent knowledge* in the training corpus for the adapters.

Evaluation metrics

Mean P@K. Following Petroni et al. (2019), we use mean precision at different values of k as the evaluation metric over the LAMA resource. Normally, as in information retrieval, we calculate the precision of a retrieved collection as the number of relevant documents proportionate to the total number of retrieved documents:

(4.3)
$$P = \frac{\text{True positive}}{\text{True positive} + \text{False positive}}$$

However, we only have one true positive for collections of all sizes. Thus, the mean precision at various values for k is equal to the whether or not the correct word is a member of the set of predictions of size k. If $k = 100$, we return a precision of 1 if the correct word is one of the 100 predictions.

The mean P@K curves we use in the figures in this section were all calculated as a macro average over all the predicate types. We iterate over the entire ConceptNet split of LAMA and count the number of correct predictions before taking the average over the entire set.

Micro-averaged precision. We also present the micro-averaged precision for different values of k, where we calculate the mean precision per predicate type and then average across all predicates. As remarked in Jiang et al. (2020), when $k = 1$ this is equal to normal micro-averaged accuracy:

(4.4)
$$\frac{1}{|P|} \sum_{(x,y) \in P} \delta(\hat{y} = y),$$

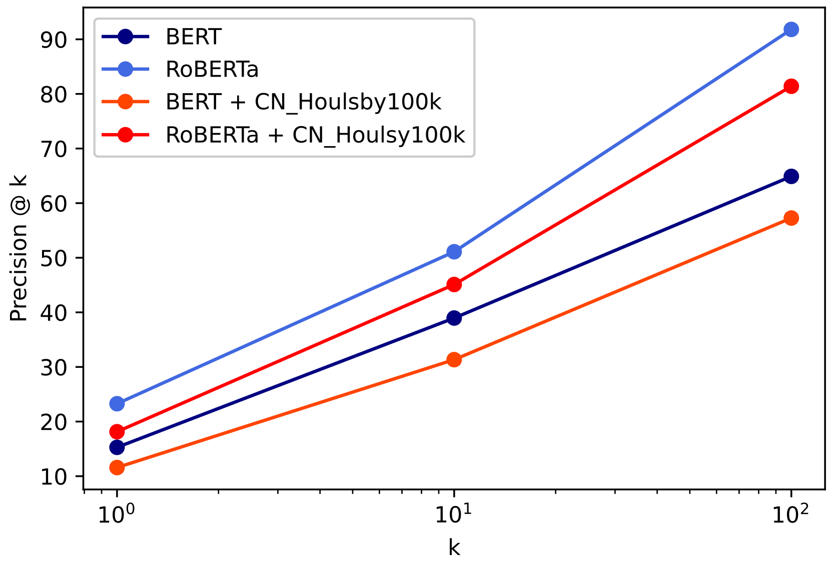
where P is the set of predicate types, $\hat{y}$ the prediction, y the gold label and δ the boolean value of the equality.

Using the predicate set from Lauscher et al. (2020)

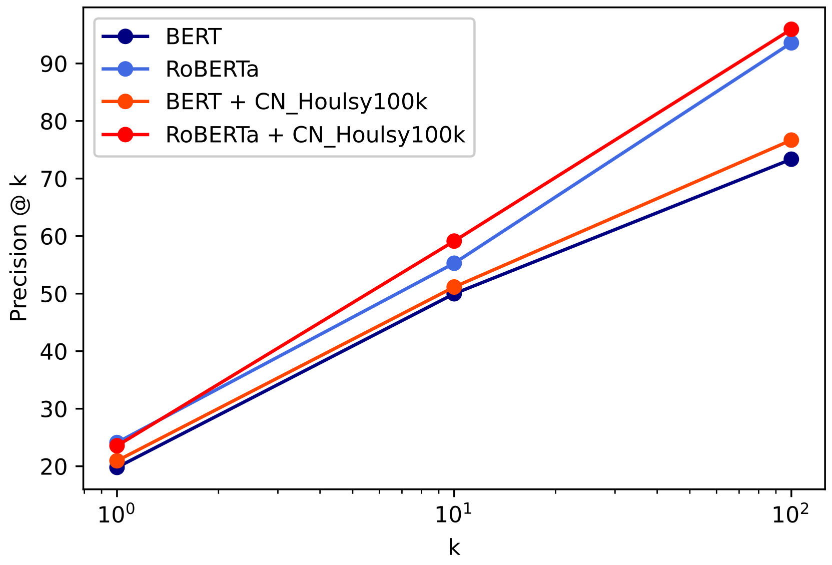
Figure 4.1 shows the mean P@K curves for two language models, with and without an adapter. The left side of the figure shows the result over all the predicates used in the ConceptNet split of LAMA (N=29774) . The injection of the adapter module decreases the performance of both BERT and RoBERTa for all the different values of k. However, the corpus from Lauscher et al. (2020) that the adapters trained on only includes one of the predicates which is also in LAMA. Hence, there is little similarity between the two sets, and the reproduction of factual knowledge cannot be expected here.

The right side of Figure 4.1, on the other hand, shows the same models and adapters, but with a test set restricted only to the ISA predicate — which is present both in the training corpus and in the test set. In the corpus from Lauscher et al. (2020), triples with this predicate make up 23% of the total corpus (N=69843).

Since both resources are extracted from ConceptNet, we check the overlap between the masked tokens in the object position in LAMA and the object position in the triplets in the training set for the adapters .The actual percentage will depend on the random walk procedure, but for the



(a) ALL PREDICATES



(b) THE ISA PREDICATE

Figure 4.1: Mean P@k curve for base models and the Houlsby adapter configuration. Base 10 log scale for X axis. a) shows the result for all the predicates in the ConceptNet split of LAMA while b) shows results for the ISA predicate only

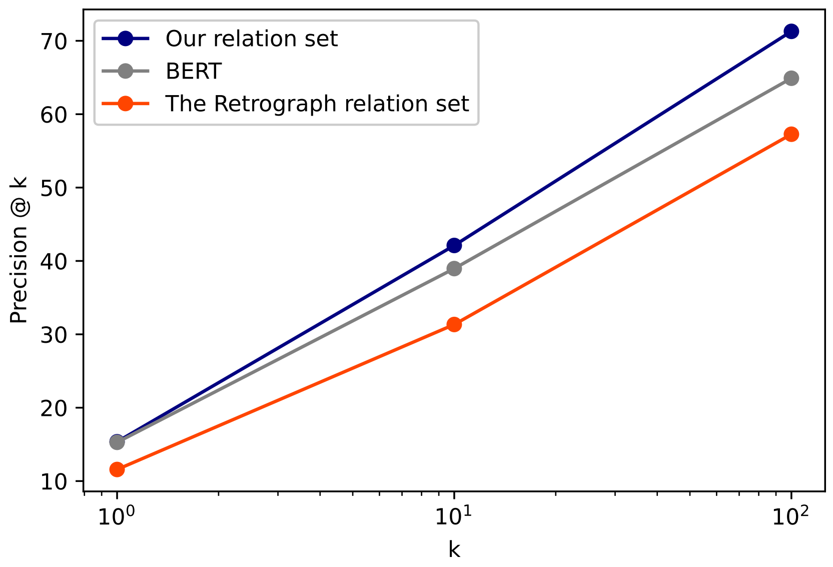


Figure 4.2: The result of different training configurations on the ConceptNet split of LAMA (Petroni et al., 2019). The two models, in dark blue and orange, use BERT base as the root model and the Houlsby configuration for their adapter, but are trained on different predicate sets of the ConceptNet graph. The gray line represents a plain BERT model without any adapter training.

sets used in Figure 4.1 there is a 5.7% overlap between concepts. That is, approximately five percent of the concepts from LAMA that the models predict are also in the training corpus in some form, although they might be included in triples that have a different predicate than what is tested for each instance in LAMA.

Despite this, the adapter-injected models perform consistently better. As this performance gain is achieved by adding only 2.1% additional parameters to the original model, and without adjusting the original weights at all, we interpret the results as a clear indication that this method of knowledge injection is effective.

Using a new predicate set

In order to further probe the effectiveness of the proposed method, we introduce a new corpus (N=99603 triples) — distilled with the same random walk procedure as in Lauscher et al. (2020), but over a new set of predicates, namely the same set of predicate types found in LAMA. By intuition, if the method is effective, the adapter injected models should score higher on average over all these relations than their non-adapter-

PREDICATE	BERT$_{\text{BASE}}$	BERT$_{\text{HOULSBY 100K}}$	Δ
`HasSubevent`	9.1	7.0	-1.1
`MadeOf`	19.9	22.0	$+2.1$
`HasPrerequisite`	16.9	12.7	-4.2
`MotivatedByGoal`	19.4	20.0	$+0.6$
`AtLocation`	15.0	16.1	$+1.1$
`CausesDesire`	14.2	25.2	$+11$
`IsA`	19.8	19.5	-0.3
`NotDesires`	12.9	7.7	-5.2
`Desires`	16.1	14.5	-1.6
`CapableOf`	18.6	14.4	-4.2
`PartOf`	19.5	21.7	$+2.2$
`HasA`	17.7	20.0	$+2.3$
`UsedFor`	16.2	16.4	$+0.2$
`ReceivesAction`	14.2	18.6	$+4.4$
`Causes`	10.3	10.4	$+0.1$
`HasProperty`	9.5	11.3	$+1.8$
Micro-average	15.6	**16.1**	**+0.5**

Table 4.3: Micro-averaged precision for our injected model compared to its non-injected counterpart over the ConceptNet split from LAMA for $k = 1$

injected counterparts.

In the first set, we use the same translations from predicate types to natural language equivalents as Lauscher et al. (2020). For example, ISA becomes `"is a"`, and SYNONYMOF becomes `"is a synonym of"`. In the new corpus, we provide additional translations for the new predicate types that are found in LAMA but were not included in Lauscher et al. (2020). Below is a list containing some of these; the full list of predicate types with their translation can be found in appendix A.2.

(4.5)

CAPABLEOF $\rightarrow$ `is capable of`

MOTIVATEDBYGOAL $\rightarrow$ `is motivated by`

USEDFOR $\rightarrow$ `is used for`

MADEOF $\rightarrow$ `is made of`

Figure 4.2 compares the result of the Houlsby configuration trained over our new corpus with that of the set from Lauscher et al. (2020) for different values of k. As can be seen from the P@K curves, models trained over our predicate set improve the performance on the full ConceptNet split of the LAMA (N= 29774) probe by up to 6.39% for BERT at large values of k. For $k = 1$, where the model must guess the correct masked object "at first try", we see little difference. This is also clear if we just compare the micro-averaged precision for the injected model and standard BERT model over our new corpus, as shown in Table 4.3. The micro-averaged precision show that performance increases with 0.5, 3.0 and 4.5 percentage points for $k = 1$, 10 and 100 respectively (see Table A.1, A.2 and A.3 in the appendix for fine-grained averages per predicate type for all three values of k).

For this new predicate set, the overlap between the training corpus for the adapters and the full ConceptNet split of LAMA is 36% on the object level, meaning that roughly one third of the concepts were seen during adapter training in some form. This provides empirical evidence for the success of the knowledge injection. Models are able to reproduce factual knowledge when queried over the lama probe, even though the phrasing of the questions in LAMA is different than the strict triplet-style of the training corpus. Again, it is important to note that this does not suggest that the models are better able to utilize the injected knowledge on different types of unseen data that require the same type of knowledge. As the experiments on SWAG, COMMONSENSEQA and the diagnostic set from GLUE show, this is unclear and would require an environment where we ensure that the knowledge required in order to classify each sample correctly was indeed included in the adapter training.

4.2.2 Error analysis

Although the results in this chapter show that adapters and masked language modeling can be used to inject structured information, it is not that clear whether or not the information in ConceptNet actually improves the more general reasoning capabilities of the models. The results on SWAG, CSQA and the diagnostic set from GLUE did not show any improvement over non-injected models. If we assume that the problem is not with annotation artifacts in these datasets, we hypothesis that this can in large be explained by problems with our extracted corpus.

Based on manual inspection, which reveals varying data quality, we argue that examples like the one below are too rigid and too specific in order to help the models in general tasks like the one listed above.

When evaluating over the LAMA probe, we measure the overlap between concepts seen during training and concepts present in the evaluation set. As previously mentioned, we would want to do the same for SWAG CSQA and GLUE as well. This, however, would require annotation of concepts involved in each sample for these datasets — an effort beyond the scope of this work.

(4.6) renin is a protein.
protein is a macromolecule.
macromolecule is a molecule.

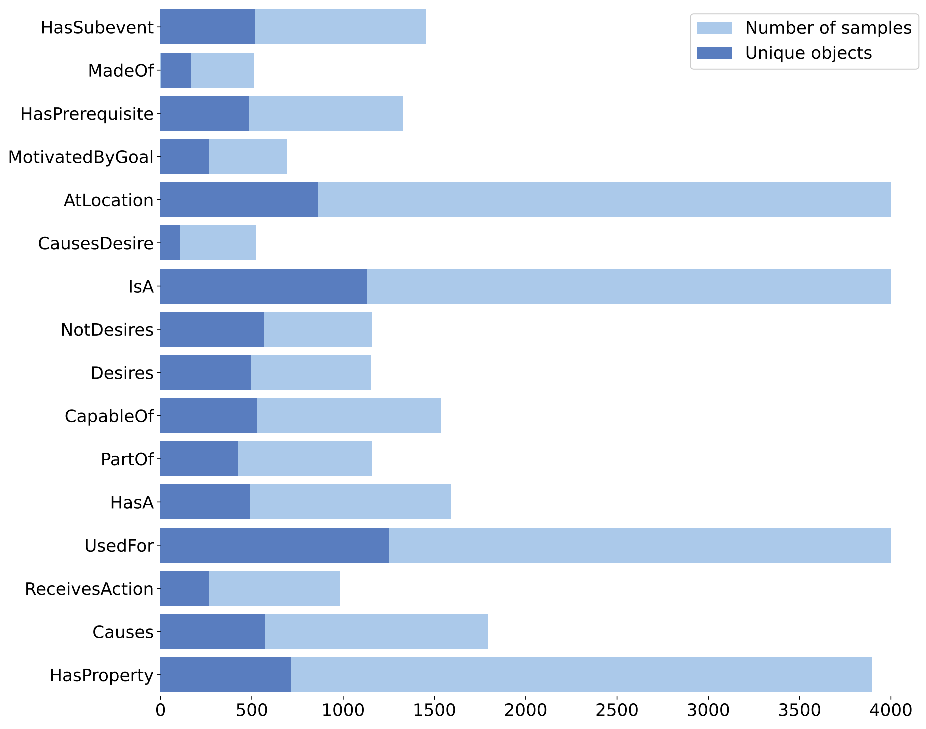


Figure 4.3: The frequency distribution of samples per predicate in LAMA together with their respective number of unique objects.

molecule is at atom.
atom is a material.
material is used for build.

Furthermore, as pointed out in Jiang et al. (2020), the distribution of objects in the LAMA probe is a bit skewed. They observe that for the relation `native_language` (not present in our test set) the object `French` make up over half of the samples. Figure 4.3 shows the number of samples per predicate type in our test set, together with the number of unique objects for each type. Although the ratio between unique objects and total samples is not that problematic in our set, we see that this motivate the need for a macro-averaged metric like the one used in Jiang et al. (2020). If not, we might get results that are either artificially high when the object that is repeated frequently is also repeated in the corpus used for training the adapters, or too low when the repeated object is not included in the training corpus at all or with a low frequency. Figure A.3 and A.4 in the appendix report the frequency distribution of the ten most common words per predicate type.

4.3 Summary

The injection of structured information from ConceptNet using adapter modules with the Houlsby configuration does not increase the performance on three datasets believed to require commonsense and world knowledge reasoning capabilities. The exception is for the world knowledge category of the GLUE diagnostic set, where the adapter increased the accuracy of a BERT base model with approximately 6 percentage points, as was also the case in Lauscher et al. (2020).

Despite the methods failure to increase the performance of language models on such common NLP benchmarks, probing analyses show that the knowledge injection was successful. Using the ConceptNet split of the LAMA knowledge probe, we show that our adapter-injected models perform significantly better on cloze-style statements where the model must predict a masked token. This indicate that models are able to reproduce the factual knowledge encoded into the adapters during adapter training, in spite of the fact that the knowledge is encoded into as little as 2.1% of the total parameter space.

We conclude from this that adapters trained with the masked language modeling objective are a viable technique for combining structured information, like that which is found in knowledge graphs, and pre-trained language models. However, they do not increase the performance on NLP benchmarks believed to require the same kind of knowledge. This might explained by discrepancies between what is encoded in the models and the kind of capabilities required in order to perform well on such benchmarks.

In the next chapter, we try to further improve the result on the LAMA probe by training multiple adapters and an additional fusion layer (Pfeiffer, Kamath et al., 2021). We also prune adapter layers in order to study where the injected knowledge is most influential.

Chapter 5

On Fusing and Pruning Adapters

Even though the results in chapter 4 provide evidence for the effectiveness of adapters for knowledge injection, it is still not clear how the injected knowledge is distributed in the model and if there are other possible architectures that are more fit for our specific use case.

In order to meet these challenges, we attempt to further improve the results on the LAMA probe by training multiple ST-adapters (see 2.3.1) and an additional layer that learns to combine these by using an attentive mechanism. Inspired by Pfeiffer, Kamath et al. (2021) and Meng et al. (2021), we implement at two-stage algorithm that decomposes the ConceptNet knowledge graph by predicate type before initiating a composition step where the extracted knowledge can be exploited in a non-destructive manner. To our knowledge, this is a novel approach for injecting world knowledge into pre-trained language models.

The second half of this chapter studies where the injected knowledge is most influential in the transformer stack. Inspired by the work of Rücklé et al. (2021) and Rogers, Kovaleva and Rumshisky (2020), we prune adapter modules from the encoder layers in BERT (Devlin et al., 2019) in order to better understand how the amount and location of the injected information affects the performance over the LAMA probe. In addition, we evaluate how the pruned models perform over the diagnostic set from GLUE (A. Wang, Pruksachatkun et al., 2019; A. Wang, A. Singh et al., 2018), comparing them with the results from chapter 4.

5.1 AdapterFusion

Although there are numerous configurations of adapters, we decide to experiment with the newly introduced AdapterFusion framework Pfeiffer, Kamath et al. (2021). We implement the same general approach of a two-stage learning algorithm and use it in a similar fashion as in Meng et al. (2021). Although their target domain (biomedical data) is quite different, we argue that the structure of the more general problem is akin to ours.

In the first stage, we train one single-task adapter (see section 2.3.1) for each predicate in a subset of ConceptNet. In the second stage we train a neural network layer, known as a fusion layer, over the collection of ST-adapters with a corpus that contains only the same predicates that the ST-adapters are trained on. What we hope to achieve is perhaps best illustrated with an analogy: we want to train adapters with disjoint expertise, like a panel of experts, and a decision maker that learns which expert to pay the most attention to at any point in time. Our procedure differentiates itself from that of Pfeiffer, Kamath et al. (2021) in the sense that we also include a graph partitioning phase inside the first step and thus regard the process of learning each predicate as a distinct downstream task. The following two sections elaborate on the learning algorithm, as we have implemented it:

5.1.1 First stage: graph partitioning and ST-adapters

The purpose of the first stage is to create a collection of ST-adapters, each specified for a specific semantic relation, that we can fuse during the second stage. By modifying the random walk procedure from Lauscher et al. (2020) (see section 3.1.1), we limit the traversal to only include subject-predicate-object triples that have a specified predicate type. In our experiments, we limit ourselves to training three ST-adapters: one for the predicate type ATLOCATION, one for ISA and one for USEDFOR. The result of the modified traversal is thus three distinct, synthetic corpora, each with its own training and validation split (80-20 split ratio). Below is two excerpts from the corpus for the ATLOCATION type:

(5.1) article is at newspaper.
newspaper is at subway.
subway is at underground.

(5.2) restrooms is at airport.
airport is at city.
city is at county.
county is at state.
state is at country.
country is at europe.

We train a ST-adapter for each predicate type's specific corpus using masked language modeling and inject it into a BERT base model. The configuration of the adapters are the same as in chapter 4, namely the hyperparameters from Houlsby et al. (2019). This training procedure is conducted in sequence, not in parallel: After training one ST-adapter, we only save the weights of the adapter and commence a new training procedure with the next predicate type. Hence, formally, the goal of the training is to learn a set of weights, Φ, for each of the predicate types $n \in \{1, ..., N\}$ such that

(5.3)
$$\Phi_n \leftarrow \arg\min_{\phi} L_n(D_n; \Theta_0, \phi),$$

where, as in Pfeiffer, Kamath et al. (2021), L_n is the masked language modeling loss over predicate type n, D_n is the corpus for that predicate type and Θ_0 is the initial parameters from the root pre-trained language model, such as BERT.

5.1.2 Second stage: fusion

The goal of the second stage is to introduce a new set of weights, Ψ, that learn to combine the ST-adapters from the previous stage in order to solve the target task. For our case, this means that Ψ is trained over a corpus that contains all three of the predicate types used to train the N ST-adapters. The general idea is that the fusion weights should learn which of the ST-adapters are most useful in each instance in the combined corpus. For example, if it is a triple that includes the ATLOCATION predicate type, the weights from the ST-adapter for that type should contribute more to improving the overall training loss than the others: [1]

(5.4)
$$\Psi \leftarrow \underset{\psi}{\arg\min}\, L_m(D_m; \Theta, \Phi_1, ..., \phi_N, \psi),$$

where Ψ is the set of weights learned during the second stage fusion training. The weights in Ψ consist of *Key*, *Value* and *Query* matrices, inserted at each transformer layer. Hence, Ψ learns to combine the ST-adapters as a dynamic function of the combined corpus where all the predicate types are present. It does this using a form of attention (see section 2.2.3), where the output of each ST-adapter at each layer for each time step is used as the *value* and *key* transformations, and the output of the feed-forward layer at each encoder block serves as the query vector (Pfeiffer, Kamath et al., 2021).

Figure 5.1 illustrates how the different components of our procedure relate to each other. In the figure, the bottleneck layer of all the ST-adapters are set to a reduction factor of 16, which for BERT_base implies a reduction of the hidden size from 768 to 48. We choose GELU for our activation function.

5.1.3 Results

Figure 5.2 shows the mean *P@K* curves (see section 4.2.1) for three models: **1)** an unaltered base model, **2)** a standard ST-adapter injected model (as in chapter 4) and **3)** one with three ST-adapters and an AdapterFusion layer on top. The model with a single injected ST-adapter trained its adapter on the combined corpus — which contains instances from all the tree predicate types. The AdapterFusion model first trained three separate ST-adapters on the three aforementioned corpora, one for each predicate type, before training a fusion layer on the same combined corpus as the single injected

[1]We implement the fusion layer using the `adapter-transformers` framework (Pfeiffer, Rücklé et al., 2020)

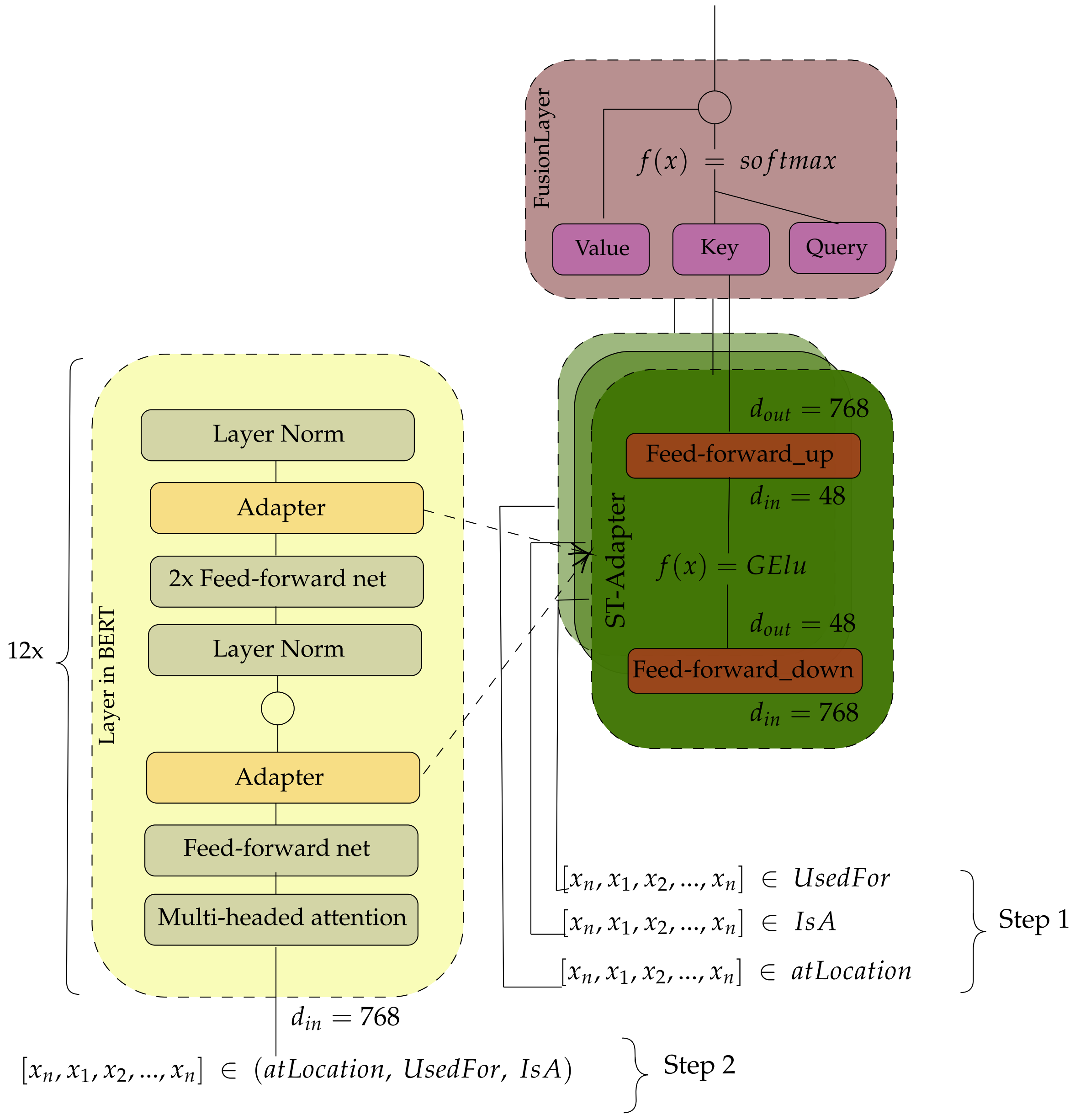


Figure 5.1: Illustration of our implementation of the two-stage algorithm for training an AdapterFusion layer over predicate type specific ST-adapters.

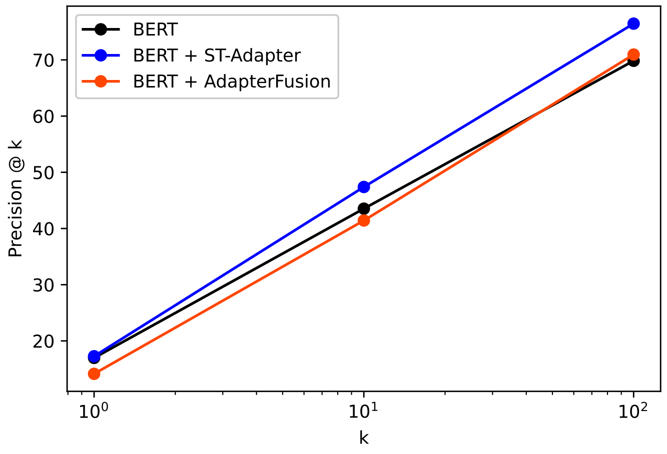


Figure 5.2: The result of AdapterFusion compared to the best configuration from chapter 4 and a standard BERT model. The predicates in this test set are only those that the Fusion model trained on.

model. All models are evaluated in a zero-shot setting over a custom subset of the LAMA probe, where only instances from the ConceptNet split with the relevant predicate types are included ($N = 12000$).

The curves show that the AdapterFusion model, despite having more weights, performs worse than both the unaltered base model and the single ST-adapter model. Following the results from 4, the ST-adapter model performs best, with up to 5.5 percent improvement for large values of k compared to the AdapterFusion model.

Error analysis

The performance of the AdapterFusion model might be explained by the quality of the corpus. Due to the size of ConceptNet and the random walk procedure from Lauscher et al. (2020), the corpora containing only one predicate type, which the three ST-adapters used during stage one of the learning algorithm, are of low quality. As illustrated in examples 5.1 and 5.2, the concept at the object position at line n is used as the subject at line $n + 1$. When the pool of examples to draw from is limited due to the predicate type restriction, these examples become increasingly worse, with the corpora for `usedFor` being of the worst quality. It is the beyond the scope of this work to improve extraction algorithms for knowledge graphs, but future work should study how these graphs can better be translated into natural language.

Furthermore, there should also be some quality assurance mechanism on the graph itself. Consider the statements `building is at rooms` and `motel is at shower` from the example below, which is an excerpt from the `atLocation` corpus. Although it looks like an extraction error, it is actually a triple present in ConceptNet. As the predicate type `atLocation` is asymmetric, it should be possible to guard against such errors by checking if the reversed statement is also in the graph; if so, something is probably off and the statement should be discarded.

(5.5) floor is at restaurant.
restaurant is at building.
building is at **rooms**
rooms is at motel.
motel is at **shower**

Another problem related to this procedure, also evident in examples 5.1 and 5.2, is that each predicate type from the graph only has one natural language equivalent: "ATLOCATION $\rightarrow$ is at". Ideally, this process would be more dynamic, instead relying on some form of dictionary look up that adapts to the semantics of the resulting triple. Consider for example the following extract of the USEDFOR corpus. In the last triple the preposition in the predicate becomes wrong due to the semantic content that arises when the subject is an activity and the object a pronoun. A more natural sounding translation wound change the preposition from *of* to *by*. These tiny details might influence how well the model is able to use the injected knowledge together with what it learned during its original pre-training, where such ungrammatical constructs are rare.

(5.6) fishing is used for sport.
sport is used for playing.
playing is used for learning.
learning is used **for** everyone.

5.2 Pruning

Pre-trained language models are often so large that they are expensive and slow to fine-tune. This motivates the need for parameter-efficient methods that are easy to share, such as adapters. The adapter we added to BERT in chapter 4 increases the total parameter size of 2.1%. While this is not a substantial increase, we nevertheless contribute to the overall complexity of the model. Inspired by the work of Rücklé et al. (2021), the following experiments measure how pruning adapters from a varying number of transformer layers affect performance. Furthermore, by pruning the adapter modules that contain injected knowledge from ConceptNet, we hope to shed light on where this information is distributed in a pre-trained language model.

5.2.1 Experimental setup

We instantiate a BERT base model and insert a ST-adapter with the hyperparameters from Houlsby et al. (2019): a reduction factor of 12 from

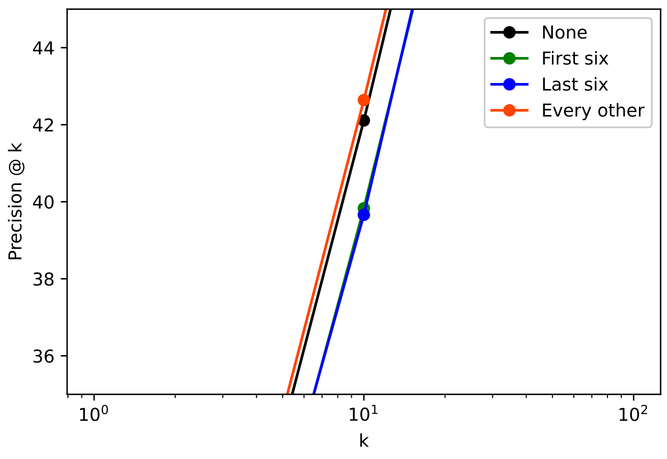


Figure 5.3: The result of different pruning strategies on the ConceptNet split of LAMA (Petroni et al., 2019). Pruning either the first or last six adapters, shown in blue and green, are considerably worse than not pruning at all, shown in black. Pruning every other adapter, on the other hand, improves performance ever so slightly. The figure is enlarged to a smaller interval on the y-axis in order to make visible the improvement for $k = 10$

the layer size in the root model and GELU as the activation function inside the adapters. We study three different pruning scenarios, inspired by the insight into the distribution of different types of knowledge presented in Rogers, Kovaleva and Rumshisky (2020), which argues that the lower layers have the most information about word ordering, the middle about syntactic knowledge and the last layers of BERT are the most task-specific. If the performance drops considerably more when removing adapters from lower layers compared to higher ones, this might indicate that the injected information, which serves as our task-specific knowledge, is better encoded lower in the stack, or vice versa. Consequently, we experiment with the following pruning configurations:

- First six: Pruning the adapter from the first six layers of the root model.
- Last six: Pruning the adapter from the last six layers.
- Every other: Pruning the adapters at layer 1, 3, 5, 7 and 9.

5.2.2 Results

Figure 5.3 shows the mean P@K curves for the three different pruning configurations. Although the effect is perhaps not as large as expected —

Model	**Adapter**	**Full**	**CS**	**World**	**NE**
BERT$_{\text{BASE}}$		37.15	32.74	13.36	27.69
RoBERTa$_{\text{BASE}}$		41.18	37.08	14.98	**36.92**
BERT$_{\text{BASE}}$	CN$_{\text{Houlsby 100k}}$	36.76	29.31	20.09	34.86
RoBERTa$_{\text{BASE}}$	CN$_{\text{Houlsby 100k}}$	**41.36**	**39.33**	12.71	29.19
BERT$_{\text{BASE}}$	CN$_{\text{ALT\_DROPS}}$	35.55	20.24	**22.98**	30.19

Table 5.1: Matthews correlation coefficient results on the full set and the Common Sense (CS), World Knowledge (World) and Named Entities (NE) categories of the GLUE diagnostic (A. Wang, A. Singh et al., 2018)

.

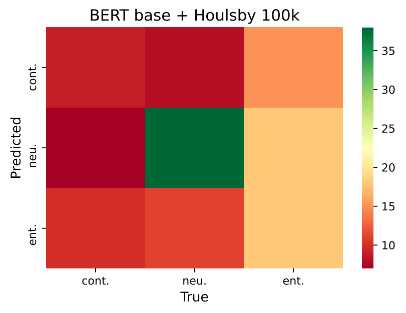
all configurations remove up to half of the introduced adapter parameters — we observe two interesting aspects of the result.

First, that pruning the adapters from either the first six or last six layers give the same result. This is surprising. Previous research, such as that of Rogers, Kovaleva and Rumshisky (2020), argues that the final layers of BERT are the most task-specific. Even though we evaluate our models in a zero-shot setting, the extracted training corpus from ConceptNet resembles the cloze-style statements in LAMA. Hence, we expect that the format of the triples seen during training would be encoded as a more task-specific type of knowledge, most prominently in the adapter modules injected at higher layers of the root model. This result contradicts that intuition, indicating the injected knowledge is instead equally important at the bottom half of the layers as the top.

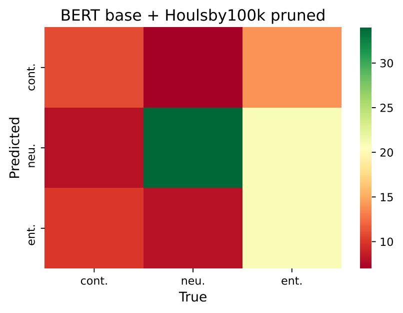
This also opposes the results found in Hao et al. (2019), who report that reverting the weights of a fine-tuned model's higher layers back to their original value hurts the model's performance more than doing the same with lower layers. As we remove the task-specific information encoded into the last six layers, we would expect this to be more detrimental for performance than the removal of the last six.

Second, pruning every other adapter performs on par with or even slightly better than the model with all 12 adapters intact. We interpret this as an indication of the injected knowledge being evenly distributed across the model, which also aligns with the previous point.

As this pruning configuration did not result in a drop in performance over the LAMA probe, we also evaluate it on the diagnostic set from GLUE (A. Wang, A. Singh et al., 2018). Table 5.1 compares the every other pruning configuration with the models from chapter 4. Although it performs considerably worse on the common sense category, it outperforms all other configurations on the world knowledge category. The confusion matrices in Figure 5.4 also show that the pruned model classify the three different labels in the diagnostic set in a similar manner as its non-pruned equivalent. The models in Table 5.1 are all trained on the predicate set from



(a) NORMAL



(b) PRUNED

Figure 5.4: Confusion matrices over the World Knowledge category from the diagnostic set in GLUE.

Lauscher et al. (2020).

As we remark in chapter 3 and 4, results over the diagnostic set might be within the expected variance as the sample size of these categories are very low, and should therefore be taken as empirically motivated indications only.

5.3 Summary

The purpose of this chapter has been two-fold: firstly, to employ the two step learning algorithm introduced in Pfeiffer, Kamath et al. (2021) in a novel way, decomposing the ConceptNet graph into single predicate corpora; and secondly, to prune adapters from transformer layers in order to probe where the injected knowledge is distributed.

The experiments using the AdapterFusion approach (Pfeiffer, Kamath et al., 2021) show that such a two step learning algorithm does not increase the performance over the LAMA probe. However, as discussed, the decomposition from a larger corpus to single predicate corpora creates substantial problems for the the random traversal procedure used in this work. Consequently, we cannot conclude that the learning algorithm itself is an unfruitful approach for injecting structured knowledge. The quality of the single predicate corpora used to train the ST-adapters is low, and we suggest that future work first look into how to create good quality datasets from subsets of knowledge graphs that contains fewer nodes than what is ideal before testing architectures like AdapterFusion. Nevertheless, we argue that this chapter shows how such an algorithm can be employed for similar use cases.

The pruning of adapters from different transformer layers show that the injected information is distributed equally across the model. We interpret this as indicating that the *mlm* objective (see section 2.2.4) does not encode the triples from ConceptNet as task-specific information only, as this would manifest itself in lower performance in experiments where the higher levels are pruned (Rogers, Kovaleva and Rumshisky, 2020). Our experiments show that pruning from the top or bottom six layers result in close to equal performance.

Furthermore, our experiments indicate that the performance achieved in chapter 4, with what is already a highly parameter efficient technique, can be replicated using even less parameters. The pruning of the adapter weights from alternating transformer layers achieves comparable results to non-pruned models over our subset of the LAMA (Petroni et al., 2019) probe and the world knowledge category from GLUE (A. Wang, A. Singh et al., 2018). We take this as an indication of the efficiency of adapter injection as a means of combining structured knowledge and language models.

Chapter 6

Conclusion

6.1 Summary

This thesis explores the problem of combining structured information with language models. By using light-weight neural networks, known as adapters, we try to increase the world knowledge of BERT (Devlin et al., 2019) and RoBERTa (Y. Liu et al., 2019), two popular transformer-based models used for a range of tasks within NLP. Using the masked language modeling objective, the adapters learn to encode the information found in ConceptNet — a knowledge graph capturing a range of commonsense concepts and the relation between these. The trained adapters are injected into BERT and RoBERTa and evaluated on three datasets targeting world knowledge and commonsense reasoning abilities: the diagnostic set from GLUE (A. Wang, Pruksachatkun et al., 2019; A. Wang, A. Singh et al., 2018), SWAG (Zellers, Bisk et al., 2018) and CommonsenseQA (Talmor et al., 2019). Furthermore, we experiment with different configurations of adapter modules and evaluate them over the LAMA probe (Petroni et al., 2019). These experiments include a two-stage algorithm using the AdaperFusion framework (Pfeiffer, Kamath et al., 2021) and an ablation study where we prune adaper modules from different encoder layers in BERT.

The overall approach of our work is inspired by Lauscher et al. (2020). Consequently, as outlined in **RQ1.a** (see 1.2), the logical starting point for this thesis was to reproduce the results achieved there. However, our reproduction (see chapter 3) failed to replicate the experimental results achieved in Lauscher et al. (2020). We conclude that the differences between our experiments and theirs are too large to be explained solely by differences in experimental setup. As our experiments report both worse performance for the adapter injected models and better performance for the baselines, we interpret this as indicating that **1)** BERT is actually better at this type of problems than what is reported in Lauscher et al. (2020), and **2)** our adapter modules do not increase the performance of pre-trained language models on the diagnostic set from GLUE.

The first point is further supported by the experiments in chapter 4, where we get to the same conclusion by testing the same baselines on the same dataset using another distribution of BERT; where our reproduction study uses the code and models provided in Lauscher et al. (2020), chapter 4 uses the `transformers` library Wolf et al. (2019).

We argue, however, based on two key observations, that the second point is still indecisive. Firstly, it is difficult to say something certain about the results due to the low sample size of the diagnostic set ($N = 1104$); when we further partition the dataset into fine-grained categories (world knowledge, commonsense), the evaluation happens over as little as 134 and 150 samples. Secondly, that research on the problems of NLI benchmarks suggest that there are other abilities than having a high level of world knowledge that make or break performance on such datasets (see section 2.5.1).

These two observations motivate the need for a more extensive evaluation. Following the same approach, we expand the experiments from Lauscher et al. (2020) to the RoBERTa model and two more benchmark datasets: SWAG and CommonsenseQA — both believed to require world knowledge and commonsense reasoning abilities.

The experimental results from chapter 4 show that the injected models do not perform better on these two datasets either. Thus, the conclusion for **RQ2** must be that our adapter modules can not be used to inject knowledge from ConceptNet (Speer, Chin and Havasi, 2017) for numerical performance gain. The exception is for the world knowledge category in the diagnostic set — our implementation of the same model as in Lauscher et al. (2020) achieves an increase of 6.73 percentage points.

However, as discussed in section 2.5.2 and 2.5.3, SWAG and CommonsenseQA suffer from the same challenges as the diagnostic set from GLUE. Consequently, it is still not clear whether or not adapters are a viable *technique* for combining structured information with language models — which is the core issue in this work — just that they do not help with performance on these datasets.

In order to evaluate the technique itself, not the overall performance on some downstream task, we use the LAMA probe (Petroni et al., 2019) to quantitatively measure how much of the knowledge seen during training our injected models are able to reproduce when faced with Cloze-style statements. Since both LAMA and the training corpus for our adapter modules are based on ConceptNet, we quantify the percentage of overlap between concepts seen during training and testing (in some form or another). Results show that our injected models perform consistently better under different scenarios, despite the injected knowledge only making up about 2.1% of the models total parameter size (see section 4.2). This indicate that adapters are a viable intermediate between structured information and language models. Furthermore, chapter 5 shows that the injected knowledge is distributed equally across the layers in BERT. Pruning adapters from the bottom half or top half of layers yields the same drop in performance, while pruning every other layer improves the overall performance over the LAMA probe by a small margin.

Our conclusion for **RQ1** must therefore be that adapters can be used to inject knowledge into pre-trained language models, and that the injected knowledge is distributed equally across layers. The injected models are able to reproduce the information seen during training when evaluated on the LAMA probe.

As for **RQ3**, the experiments with the AdapterFusion framework in chapter 5 did not improve performance over the LAMA probe when compared to standard single-task adapter injection and baseline models. However, due to problems with the decomposition from a larger corpus to single predicate corpora, and the previous success of Meng et al. (2021) with a similar approach in the biomedical domain, we conclude that these results are indecisive, and may prove fruitful in the future.

6.2 Contributions

The main contribution of this thesis is to provide evidence for the effectiveness of adapters as a way of combining the structured information found in knowledge graphs with pre-trained language models, as concluded in **RQ1**. Furthermore, our negative results from **RQ2** also contribute to the intuition that the inclusion of background knowledge does not necessarily increase performance on datasets believed to require the same knowledge. Current pre-trained language models consistently exploit annotation artifacts, if present, and as long as the datasets we use to evaluate such models contains such irregularities, it is intuitive to think that increasing the models parameter size will result in more performance gain compared to including more background knowledge.

We also regard our reproduction study of the work by (Lauscher et al., 2020) as a contribution towards a more rigorous evaluation of results in NLP in general. Furthermore, we also argue that our use of the AdaperFusion framework on a knowledge graph like ConceptNet is a novelty — partitioning the graph by what is essentially the edge of a directed, cyclic graph and training one single-task adapter on each predicate type before applying the fusion layer from Pfeiffer, Kamath et al. (2021). Although the conclusion of **RQ3** is that this approach does not improve the performance over the LAMA probe, we argue that we have outlined a general technique that might do just this if further developed.

6.3 Future work

Combining structured information with language models is a difficult problem. If done efficiently, we foresee easier deployment of models into settings where domain knowledge is crucial. We identify two key challenges that needs to be addressed in order for the field to move towards such a goal. We outline these below.

Representation and encoding. In this thesis, the structured information is in the form of a knowledge graph. The encoding procedure, where we go from graph triple to a numerical vector compatible with the architecture of modern language models, rely on the masked language modeling objective. We encourage future work to investigate other ways of both representing and encoding the structured information. In particular, we think that there might be other learning objectives more apt for encoding graph objects. Also, we hypothesize that a system able to incorporate domain knowledge from a database management system, for example a graph database on the RDF format, would be an appealing solution for many scenarios as it would enable model developers to deploy a language model directly over an already existing knowledge source.

We have only considered the scenario where the structured information is injected into the models weights. This has the obvious benefit of reducing the number of components in an overall system, as the knowledge injected model can be shipped as easily as a non-injected one, but there might be more fruitful ways of combining the two. For example, we consider recent work by Borgeaud et al., 2021, where the knowledge source is queried as an ad-hoc resource, to be a promising research direction for this end.

Evaluation. It is not straight-forward to quantitatively assess the knowledge capabilities in a language model. While there has been a lot of recent efforts that attempt to isolate the task of knowledge extraction from more general inference abilities (Bouraoui, Camacho-Collados and Schockaert, 2020; Elazar et al., 2021; Jiang et al., 2020; Petroni et al., 2019), we argue that all these rely on short, straight-forward statements, and that future work should look into ways of achieving the same assessment but over more free-flowing text. This might mean that we should annotate more NLP datasets with fine-grained categories, such as in the diagnostic set from GLUE.

As indicated by Jiang et al. (2020), the way we query the models for knowledge highly influences the outcome. Future work should therefore also look into methods of making language models more robust to paraphrases of semantically equivalent statements, or else we might deem the models less knowledgeable than they really are.

Chapter 7

Bibliography

Bender, E. M. and A. Koller (2020). 'Climbing towards NLU: On Meaning, Form, and Understanding in the Age of Data'. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, pp. 5185–5198. DOI: 10.18653/v1/2020.acl-main.463. URL: https://aclanthology.org/2020.acl-main.463.

Bodenreider, O. (2004). 'The Unified Medical Language System (UMLS): integrating biomedical terminology'. In: *Nucleic Acids Research* 32, pp. D267–D270. ISSN: 0305-1048. DOI: 10.1093/nar/gkh061. URL: https://doi.org/10.1093/nar/gkh061.

Borgeaud, S. et al. (2021). *Improving language models by retrieving from trillions of tokens*. DOI: 10.48550/ARXIV.2112.04426. URL: https://arxiv.org/abs/2112.04426.

Bouraoui, Z., J. Camacho-Collados and S. Schockaert (2020). 'Inducing Relational Knowledge from BERT'. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 05. New York, USA: Association for the Advancement of Artificial Intelligence, pp. 7456–7463. DOI: 10.1609/aaai.v34i05.6242. URL: https://ojs.aaai.org/index.php/AAAI/article/view/6242.

Bowman, S. R., G. Angeli et al. (2015). 'A large annotated corpus for learning natural language inference'. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, pp. 632–642. DOI: 10.18653/v1/D15-1075. URL: https://aclanthology.org/D15-1075.

Bowman, S. R. and G. Dahl (2021). 'What Will it Take to Fix Benchmarking in Natural Language Understanding?' In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, pp. 4843–4855. DOI: 10.18653/v1/2021.naacl-main.385. URL: https://aclanthology.org/2021.naacl-main.385.

Brown, T. et al. (2020). 'Language Models are Few-Shot Learners'. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Online: Curran Associates, Inc., pp. 1877–

1901. URL: https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Cho, K. et al. (2014). 'Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation'. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, pp. 1724–1734. DOI: 10.3115/v1/D14-1179. URL: https://aclanthology.org/D14-1179.

Clark, P. et al. (2018). 'Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge'. In: *CoRR* abs/1803.05457. arXiv: 1803.05457. URL: http://arxiv.org/abs/1803.05457.

Colon-Hernandez, P. et al. (2021). 'Combining pre-trained language models and structured knowledge'. In: *arXiv preprint arXiv:2101.12294*. URL: https://arxiv.org/abs/2101.12294.

Conneau, A. et al. (2018). 'XNLI: Evaluating Cross-lingual Sentence Representations'. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, pp. 2475–2485. DOI: 10.18653/v1/D18-1269. URL: https://aclanthology.org/D18-1269.

Cooper, R. et al. (1996). *Using the framework*. Tech. rep. Technical Report LRE 62-051 D-16, The FraCaS Consortium.

Dagan, I., O. Glickman and B. Magnini (2005). 'The pascal recognising textual entailment challenge'. In: *Machine Learning Challenges Workshop*. Springer, pp. 177–190.

Devlin, J. et al. (2019). 'BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding'. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

Donnelly, K. et al. (2006). 'SNOMED-CT: The advanced terminology and coding system for eHealth'. In: *Studies in health technology and informatics* 121, p. 279.

Elazar, Y. et al. (2021). 'Measuring and Improving Consistency in Pretrained Language Models'. In: *Transactions of the Association for Computational Linguistics* 9, pp. 1012–1031. DOI: 10.1162/tacl_a_00410. URL: https://aclanthology.org/2021.tacl-1.60.

Falke, T. et al. (2019). 'Ranking Generated Summaries by Correctness: An Interesting but Challenging Application for Natural Language Inference'. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 2214–2220. DOI: 10.18653/v1/P19-1213. URL: https://aclanthology.org/P19-1213.

Fedus, W., B. Zoph and N. Shazeer (2022). 'Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity'. In: *Journal of Machine Learning Research* 23.120, pp. 1–39. URL: http://jmlr.org/papers/v23/21-0998.html.

Gajbhiye, A., N. A. Moubayed and S. Bradley (2021). 'ExBERT: An External Knowledge Enhanced BERT for Natural Language Inference'. In: *International Conference on Artificial Neural Networks*. Springer. Online, pp. 460–472. URL: https://arxiv.org/abs/2108.01589.

Glockner, M., V. Shwartz and Y. Goldberg (2018). 'Breaking NLI Systems with Sentences that Require Simple Lexical Inferences'. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Melbourne, Australia: Association for Computational Linguistics, pp. 650–655. DOI: 10.18653/v1/P18-2103. URL: https://www.aclweb.org/anthology/P18-2103.

Goldberg, Y. (2017). *Neural network methods for natural language processing*. Vol. 10. 1. Morgan and Claypool Publishers, pp. 1–309.

Grover, A. and J. Leskovec (2016). *node2vec: Scalable Feature Learning for Networks*. arXiv: 1607.00653 [cs.SI].

Gururangan, S. et al. (2018). 'Annotation Artifacts in Natural Language Inference Data'. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, pp. 107–112. DOI: 10.18653/v1/N18-2017. URL: https://aclanthology.org/N18-2017.

Hagoort, P. et al. (2004). 'Integration of word meaning and world knowledge in language comprehension'. In: *Science* 304.5669, pp. 438–441.

Hao, Y. et al. (2019). 'Visualizing and Understanding the Effectiveness of BERT'. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, pp. 4143–4152. DOI: 10.18653/v1/D19-1424. URL: https://aclanthology.org/D19-1424.

He, B. et al. (2020). 'BERT-MK: Integrating Graph Contextualized Knowledge into Pre-trained Language Models'. In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, pp. 2281–2290. DOI: 10.18653/v1/2020.findings-emnlp.207. URL: https://aclanthology.org/2020.findings-emnlp.207.

Hendrycks, D. and K. Gimpel (2016). *Gaussian Error Linear Units (GELUs)*. arXiv: 1606.08415 [cs.LG].

Hochreiter, S. and J. Schmidhuber (1997). 'Long short-term memory'. In: *Neural computation* 9.8, pp. 1735–1780.

Houlsby, N. et al. (2019). 'Parameter-Efficient Transfer Learning for NLP'. In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by K. Chaudhuri and R. Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. California, USA: PMLR, pp. 2790–2799. URL: https://proceedings.mlr.press/v97/houlsby19a.html.

Jiang, Z. et al. (2020). 'How can we know what language models know?' In: *Transactions of the Association for Computational Linguistics* 8, pp. 423–438.

Jurafsky, D. and J. H. Martin (2021). 'Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition'.

Jurman, G., S. Riccadonna and C. Furlanello (2012). 'A Comparison of MCC and CEN Error Measures in Multi-Class Prediction'. In: *PLoS ONE* 7.8. Ed. by G. Biondi-Zoccai, e41882. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0041882. URL: http://dx.doi.org/10.1371/journal.pone.0041882.

Kim, N. et al. (2019). 'Probing What Different NLP Tasks Teach Machines about Function Word Comprehension'. In: *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM 2019)*. Minneapolis, Minnesota: Association for Computational Linguistics, pp. 235–249. DOI: 10.18653/v1/S19-1026. URL: https://aclanthology.org/S19-1026.

Kingma, D. P. and J. Ba (2015). 'Adam: A Method for Stochastic Optimization'. In: *3rd International Conference on Learning Representations*. New York, USA: International Conference on Learning Representations (ICLR). arXiv: 1412.6980 [cs.LG].

Lauscher, A. et al. (2020). 'Common Sense or World Knowledge? Investigating Adapter-Based Knowledge Injection into Pretrained Transformers'. In: *CoRR* abs/2005.11787. arXiv: 2005.11787. URL: https://arxiv.org/abs/2005.11787.

LeCun, Y. et al. (1999). 'Object recognition with gradient-based learning'. In: *Shape, contour and grouping in computer vision*. Springer, pp. 319–345.

Levine, Y. et al. (2020). 'SenseBERT: Driving Some Sense into BERT'. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, pp. 4656–4667. DOI: 10.18653/v1/2020.acl-main.423. URL: https://aclanthology.org/2020.acl-main.423.

Liu, H. and P. Singh (2004). 'ConceptNet—a practical commonsense reasoning tool-kit'. In: *BT technology journal* 22.4, pp. 211–226.

Liu, Y. et al. (2019). 'Roberta: A robustly optimized bert pretraining approach'. In: *arXiv preprint arXiv:1907.11692*. URL: https://arxiv.org/abs/1907.11692.

Logan, R. et al. (2019). 'Barack's Wife Hillary: Using Knowledge Graphs for Fact-Aware Language Modeling'. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 5962–5971. DOI: 10.18653/v1/P19-1598. URL: https://aclanthology.org/P19-1598.

MacCartney, B. and C. D. Manning (2009). *Natural language inference [Doctoral dissertation, Standford University]*. URL: https://nlp.stanford.edu/~wcmac/.

Madni, A., C. Madni and W. Lin (1998). 'IDEON/sup TM//IPPD: an ontology for systems engineering process design and management'. In: *SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.98CH36218)*. Vol. 3, 2597–2602 vol.3. DOI: 10.1109/ICSMC.1998.725050.

Mahesh, K., S. Nirenburg et al. (1995). 'A situated ontology for practical NLP'. In: *Proceedings of the IJCAI-95 Workshop on Basic Ontological Issues in Knowledge Sharing*. Vol. 19, p. 21.

Meng, Z. et al. (2021). 'Mixture-of-Partitions: Infusing Large Biomedical Knowledge Graphs into BERT'. In: *Proceedings of the 2021 Conference* *on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, pp. 4672–4681. DOI: 10.18653/v1/2021.emnlp-main.383. URL: https://aclanthology.org/2021.emnlp-main.383.

Mikolov, T. et al. (2013). 'Distributed Representations of Words and Phrases and their Compositionality'. In: *Advances in Neural Information Processing Systems*. Ed. by C. Burges et al. Vol. 26. Lake Tahoe, USA: Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf.

Miller, G. A. (1995). 'WordNet: a lexical database for English'. In: *Communications of the ACM* 38.11, pp. 39–41.

Nie, Y. et al. (2020). 'Adversarial NLI: A New Benchmark for Natural Language Understanding'. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, pp. 4885–4901. DOI: 10.18653/v1/2020.acl-main.441. URL: https://www.aclweb.org/anthology/2020.acl-main.441.

Pedersen, T., S. Patwardhan, J. Michelizzi et al. (2004). 'WordNet:: Similarity-Measuring the Relatedness of Concepts.' In: *Preceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI)*. Vol. 4. San Jose, USA: Association for the Advancement of Artificial Intelligence (AAAI), pp. 25–29.

Peters, M. E., M. Neumann, M. Iyyer et al. (2018). 'Deep Contextualized Word Representations'. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, pp. 2227–2237. DOI: 10.18653/v1/N18-1202. URL: https://aclanthology.org/N18-1202.

Peters, M. E., M. Neumann, R. Logan et al. (2019). 'Knowledge Enhanced Contextual Word Representations'. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, pp. 43–54. DOI: 10.18653/v1/D19-1005. URL: https://aclanthology.org/D19-1005.

Petroni, F. et al. (2019). 'Language Models as Knowledge Bases?' In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, pp. 2463–2473. DOI: 10.18653/v1/D19-1250. URL: https://aclanthology.org/D19-1250.

Pfeiffer, J., A. Kamath et al. (2021). 'AdapterFusion: Non-Destructive Task Composition for Transfer Learning'. In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Online: Association for Computational Linguistics, pp. 487–503. DOI: 10.18653/v1/2021.eacl-main.39. URL: https://aclanthology.org/2021.eacl-main.39.

Pfeiffer, J., A. Rücklé et al. (2020). 'AdapterHub: A Framework for Adapting Transformers'. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, pp. 46–54. DOI: 10.18653/v1/2020.emnlp-demos.7. URL: https://aclanthology.org/2020.emnlp-demos.7.

Poliak, A. (2020). 'A survey on Recognizing Textual Entailment as an NLP Evaluation'. In: *Proceedings of the First Workshop on Evaluation and Comparison of NLP Systems*. Online: Association for Computational Linguistics, pp. 92–109. DOI: 10.18653/v1/2020.eval4nlp-1.10. URL: https://aclanthology.org/2020.eval4nlp-1.10.

Poliak, A., Y. Belinkov et al. (2018). 'On the Evaluation of Semantic Phenomena in Neural Machine Translation Using Natural Language Inference'. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, pp. 513–523. DOI: 10.18653/v1/N18-2082. URL: https://aclanthology.org/N18-2082.

Poliak, A., J. Naradowsky et al. (2018). 'Hypothesis Only Baselines in Natural Language Inference'. In: *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*. New Orleans, Louisiana: Association for Computational Linguistics, pp. 180–191. DOI: 10.18653/v1/S18-2023. URL: https://aclanthology.org/S18-2023.

Rebuffi, S.-A., H. Bilen and A. Vedaldi (2017). 'Learning multiple visual domains with residual adapters'. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Long Beach, USA: Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2017/file/e7b24b112a44fdd9ee93bdf998c6ca0e-Paper.pdf.

Roemmele, M., C. A. Bejan and A. S. Gordon (2011). 'Choice of Plausible Alternatives: An Evaluation of Commonsense Causal Reasoning.' In: *AAAI spring symposium: logical formalizations of commonsense reasoning*, pp. 90–95.

Rogers, A., O. Kovaleva and A. Rumshisky (2020). 'A primer in bertology: What we know about how bert works'. In: *Transactions of the Association for Computational Linguistics* 8, pp. 842–866.

Rücklé, A. et al. (2021). 'AdapterDrop: On the Efficiency of Adapters in Transformers'. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, pp. 7930–7946. DOI: 10.18653/v1/2021.emnlp-main.626. URL: https://aclanthology.org/2021.emnlp-main.626.

Schuster, M. and K. K. Paliwal (1997). 'Bidirectional recurrent neural networks'. In: *IEEE transactions on Signal Processing* 45.11, pp. 2673–2681.

Shi, L. et al. (2017). 'Semantic health knowledge graph: semantic integration of heterogeneous medical knowledge and services'. In: *BioMed research international* 2017.

Singh, P. et al. (2002). 'Open mind common sense: Knowledge acquisition from the general public'. In: *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. Springer, pp. 1223–1237.

Sinha, K. et al. (2021). 'UnNatural Language Inference'. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the*

11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). Online: Association for Computational Linguistics, pp. 7329–7346. DOI: 10.18653/v1/2021.acl-long.569. URL: https://aclanthology.org/2021.acl-long.569.

Speer, R., J. Chin and C. Havasi (2017). 'Conceptnet 5.5: An open multilingual graph of general knowledge'. In: *Thirty-first AAAI conference on artificial intelligence*.

Strapparava, C., A. Valitutti et al. (2004). 'Wordnet affect: an affective extension of wordnet.' In: *The International Conference on Language Resources and Evaluation*. Vol. 4. Lisbon.

Suchanek, F. M., G. Kasneci and G. Weikum (2007). 'Yago: a core of semantic knowledge'. In: *Proceedings of the 16th international conference on World Wide Web*, pp. 697–706.

Sun, Y. et al. (2021). *ERNIE 3.0: Large-scale Knowledge Enhanced Pre-training for Language Understanding and Generation*. arXiv: 2107.02137 [cs.CL].

Syed, Z. et al. (2016). 'UCO: A unified cybersecurity ontology'. In: *Workshops at the thirtieth AAAI conference on artificial intelligence*.

Talmor, A. et al. (2019). 'CommonsenseQA: A Question Answering Challenge Targeting Commonsense Knowledge'. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, pp. 4149–4158. DOI: 10.18653/v1/N19-1421. URL: https://aclanthology.org/N19-1421.

Tang, R. et al. (2020). 'Rapidly Bootstrapping a Question Answering Dataset for COVID-19'. In: *CoRR* abs/2004.11339. arXiv: 2004.11339. URL: https://arxiv.org/abs/2004.11339.

Tenney, I., D. Das and E. Pavlick (2019). 'BERT Rediscovers the Classical NLP Pipeline'. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 4593–4601. DOI: 10.18653/v1/P19-1452. URL: https://aclanthology.org/P19-1452.

Vaswani, A. et al. (2017). 'Attention is All you Need'. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Long Beach, USA: Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Wang, A., Y. Pruksachatkun et al. (2019). 'SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems'. In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Vancouver, Canada: Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2019/file/4496bf24afe7fab6f046bf4923da8de6-Paper.pdf.

Wang, A., A. Singh et al. (2018). 'GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding'. In: *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Brussels, Belgium: Association for Computational Linguistics, pp. 353–355. DOI: 10.18653/v1/W18-5446. URL: https://aclanthology.org/W18-5446.

Wang, R. et al. (2021). ‘K-Adapter: Infusing Knowledge into Pre-Trained Models with Adapters’. In: *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Online: Association for Computational Linguistics, pp. 1405–1418. DOI: 10.18653/v1/2021.findings-acl.121. URL: https://aclanthology.org/2021.findings-acl.121.

Williams, A., N. Nangia and S. Bowman (2018). ‘A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference’. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, pp. 1112–1122. DOI: 10.18653/v1/N18-1101. URL: https://aclanthology.org/N18-1101.

Wolf, T. et al. (2019). ‘Huggingface’s transformers: State-of-the-art natural language processing’. In: *arXiv preprint arXiv:1910.03771*. URL: https://arxiv.org/abs/1910.03771.

Zellers, R., Y. Bisk et al. (2018). ‘SWAG: A Large-Scale Adversarial Dataset for Grounded Commonsense Inference’. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, pp. 93–104. DOI: 10.18653/v1/D18-1009. URL: https://aclanthology.org/D18-1009.

Zellers, R., A. Holtzman et al. (2019). ‘HellaSwag: Can a Machine Really Finish Your Sentence?’ In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 4791–4800. DOI: 10.18653/v1/P19-1472. URL: https://aclanthology.org/P19-1472.

Zhang, X. et al. (2022). ‘GreaseLM: Graph REASoning Enhanced Language Models for Question Answering’. In: *CoRR* abs/2201.08860. arXiv: 2201.08860. URL: https://arxiv.org/abs/2201.08860.

Zhang, Z. et al. (2019). ‘ERNIE: Enhanced Language Representation with Informative Entities’. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 1441–1451. DOI: 10.18653/v1/P19-1139. URL: https://aclanthology.org/P19-1139.

Zhu, Y. et al. (2015). ‘Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books’. In: *CoRR* abs/1506.06724. arXiv: 1506.06724. URL: http://arxiv.org/abs/1506.06724.

Appendix A

Figures and tables

(A.1)

1) **Asymmetric**: AtLocation, CapableOf, Causes, CausesDesire, CreatedBy, DefinedAs, DerivedFrom, Desires, Entails, ExternalURL, FormOf, HasA, HasContext, HasFirstSubevent, HasLastSubevent, HasPrerequisite, HasProperty, InstanceOf, IsA, MadeOf, MannerOf, MotivatedByGoal, ObstructedBy, PartOf, ReceivesAction, SenseOf, SymbolOf, and UsedFor
2) **Symmetric**: Antonym, DistinctFrom, EtymologicallyRelatedTo, LocatedNear, RelatedTo, SimilarTo, and Synonym

Figure A.1: All predicate types in ConceptNet.

PREDICATE	BERT$_{\text{BASE}}$	BERT$_{\text{HOULSBY 100K}}$	Δ
`HasSubevent`	9.1	7.0	-1.1
`MadeOf`	19.9	22.0	$+2.1$
`HasPrerequisite`	16.9	12.7	-4.2
`MotivatedByGoal`	19.4	20.0	$+0.6$
`AtLocation`	15.0	16.1	$+1.1$
`CausesDesire`	14.2	25.2	$+11$
`IsA`	19.8	19.5	-0.3
`NotDesires`	12.9	7.7	-5.2
`Desires`	16.1	14.5	-1.6
`CapableOf`	18.6	14.4	-4.2
`PartOf`	19.5	21.7	$+2.2$
`HasA`	17.7	20.0	$+2.3$
`UsedFor`	16.2	16.4	$+0.2$
`ReceivesAction`	14.2	18.6	$+4.4$
`Causes`	10.3	10.4	$+0.1$
`HasProperty`	9.5	11.3	$+1.8$
Micro-average	15.6	**16.1**	**+0.5**

Table A.1: Micro-averaged precision for our injected model compared to its non-injected counterpart over the ConceptNet split from LAMA for $k = 1$ (Petroni et al., 2019)

PREDICATE	BERT$_{\text{BASE}}$	BERT$_{\text{HOULSBY 100K}}$	Δ
`HasSubevent`	23.6	18.9	-4.7
`MadeOf`	51.8	52.8	$+1$
`HasPrerequisite`	40.6	38.0	-2.6
`MotivatedByGoal`	43.5	53.1	$+9.6$
`AtLocation`	42.5	43.3	$+0.8$
`CausesDesire`	35.7	56.6	$+20.9$
`IsA`	49.9	49.2	-0.7
`NotDesires`	34.4	23.3	-11.1
`Desires`	34.4	37.2	$+2.8$
`CapableOf`	44.4	42.0	-2.4
`PartOf`	46.0	53.9	$+7.9$
`HasA`	39.8	48.7	$+8.9$
`UsedFor`	38.1	46.2	$+8.1$
`ReceivesAction`	42.4	46.2	$+3.8$
`Causes`	31.6	31.2	-0.4
`HasProperty`	28.0	34.1	$+6.1$
Micro-average	39.2	**42.2**	**+3.0**

Table A.2: Micro-averaged precision for our injected model compared to its non-injected counterpart over the ConceptNet split from LAMA for $k = 10$ (Petroni et al., 2019)

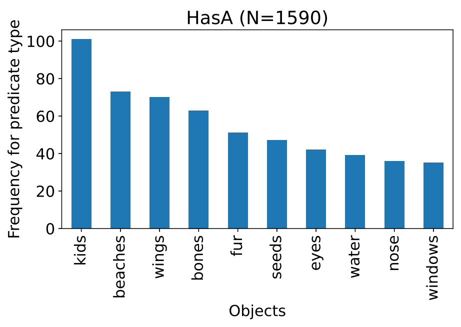
PREDICATE	BERT$_{\text{BASE}}$	BERT$_{\text{HOULSBY 100K}}$	Δ
`HasSubevent`	40.2	33.2	-7.0
`MadeOf`	76.0	80.7	4.7
`HasPrerequisite`	67.3	67.5	-0.2
`MotivatedByGoal`	74.8	78.2	$+3.4$
`AtLocation`	74.4	78.5	$+4.1$
`CausesDesire`	69.2	88.3	$+19.1$
`IsA`	73.3	78.6	$+5.3$
`NotDesires`	54.5	44.8	$-9,6$
`Desires`	64.5	62.9	-1.6
`CapableOf`	73.7	72.2	-1.5
`PartOf`	70.7	78.9	$+8.2$
`HasA`	67.1	76.3	$+9.2$
`UsedFor`	61.3	72.9	$+11.6$
`ReceivesAction`	71.8	76.1	$+4.3$
`Causes`	55.8	61.2	$+5.4$
`HasProperty`	52.5	68.6	$+16.1$
Micro-average	65.4	**69.9**	**+4.5**

Table A.3: Micro-averaged precision for our injected model compared to its non-injected counterpart over the ConceptNet split from LAMA for $k = 100$ (Petroni et al., 2019)

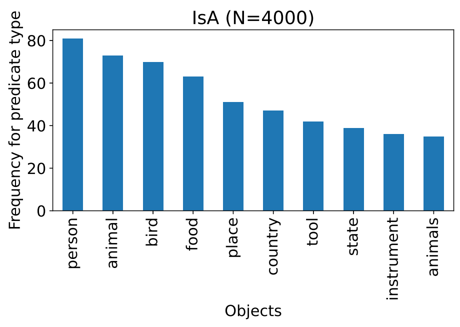
(A.2)

$$\textsc{atLocation} \rightarrow \texttt{is at}$$
$$\textsc{capableOf} \rightarrow \texttt{is capable of}$$
$$\textsc{causes} \rightarrow \texttt{causes}$$
$$\textsc{causesDesire} \rightarrow \texttt{causes desire of}$$
$$\textsc{desires} \rightarrow \texttt{desires}$$
$$\textsc{hasA} \rightarrow \texttt{hasA}$$
$$\textsc{hasPrerequisite} \rightarrow \texttt{has prerequisite}$$
$$\textsc{hasProperty} \rightarrow \texttt{hasProperty}$$
$$\textsc{hasSubevent} \rightarrow \texttt{hasSubevent}$$
$$\textsc{isA} \rightarrow \texttt{is a}$$
$$\textsc{locatedNear} \rightarrow \texttt{is located near}$$
$$\textsc{madeOf} \rightarrow \texttt{is made of}$$
$$\textsc{motivatedByGoal} \rightarrow \texttt{is motivated by}$$
$$\textsc{usedFor} \rightarrow \texttt{is used for}$$
$$\textsc{partOf} \rightarrow \texttt{partOf}$$
$$\textsc{receivesAction} \rightarrow \texttt{receives}$$

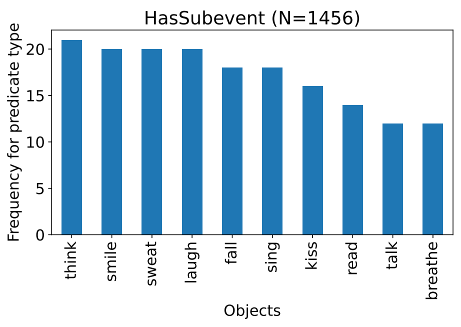
Figure A.2: All predicate types in the ConceptNet split of the LAMA probing dataset.



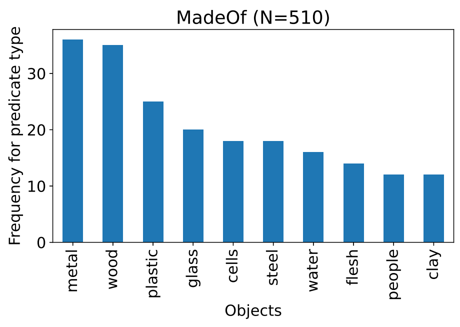
(a) HasA



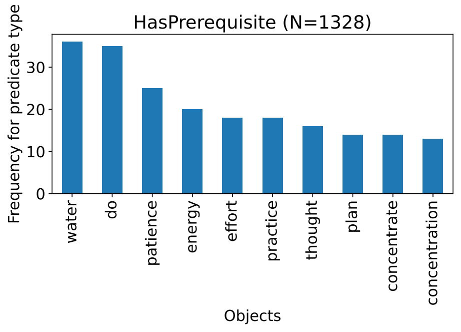
(b) IsA



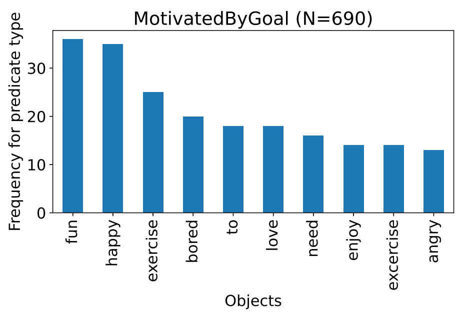
(c) HasSubevent



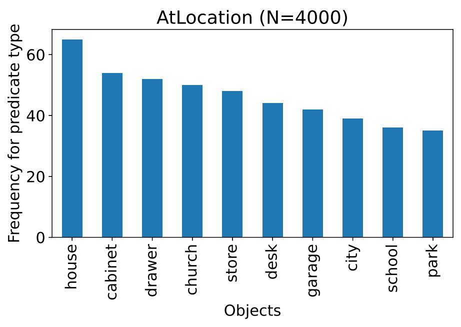
(d) MadeOf



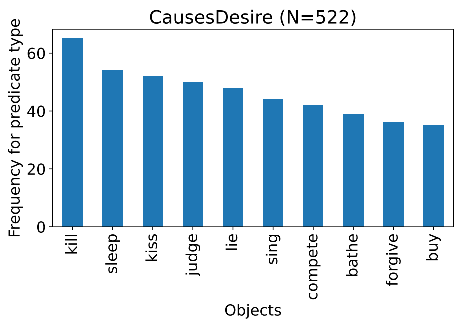
(e) HasPrerequisite



(f) MotivatedByGoal

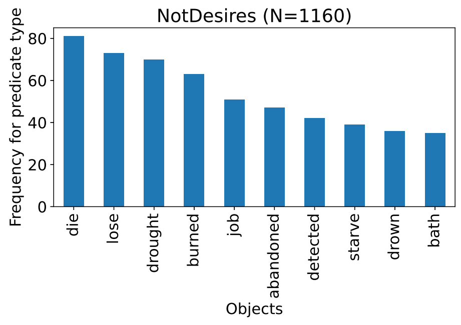


(g) AtLocation

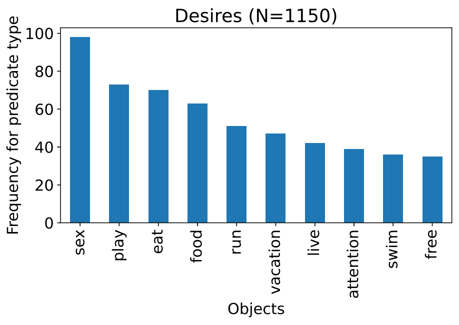


(h) CausesDesire

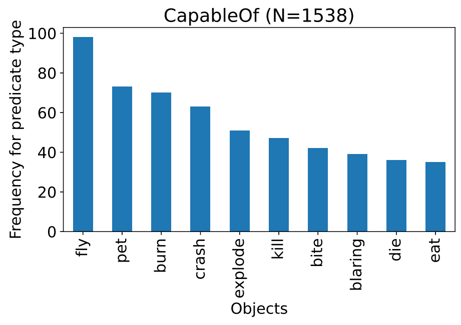
Figure A.3: Distribution of the top ten objects for the first eight predicate types of the ConceptNet split in LAMA



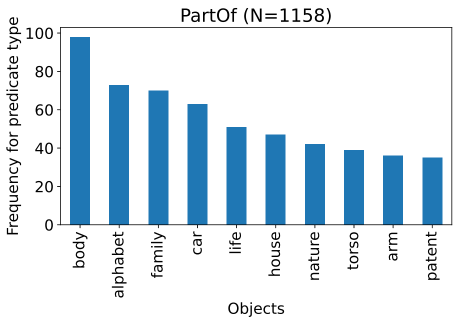
(a) NotDesires



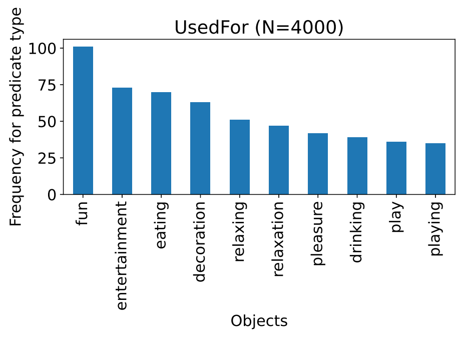
(b) Desires



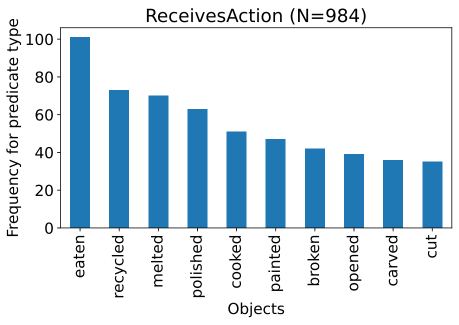
(c) CapableOf



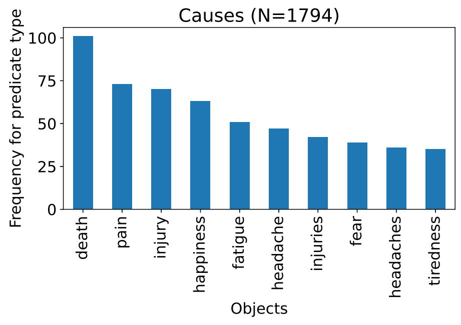
(d) PartOf



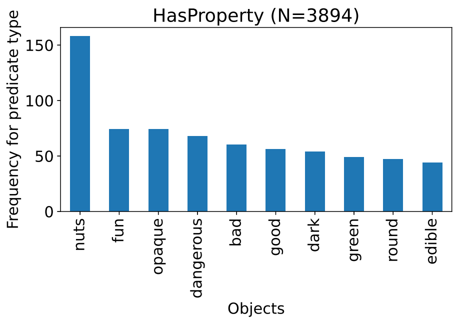
(e) UsedFor



(f) ReceivesAction



(g) Causes



(h) HasProperty

Figure A.4: Distribution of the top ten objects for the last eight predicate types of the ConceptNet split in LAMA

Appendix B

Code and reproducibility

B.1 Environment

We list the most relevant packages for our experiments in the table below. All experiments ran using the seed 42, which is set as the default in our training scripts. The code for the reproduction study in chapter 3 can be found at https://github.com/Wluper/Retrograph.

Package	Used for	Version
`adapter-transformers`	Implementing our adapter modules	2.2.0
`transformers`	Scheduling, modeling and optimization	4.3.3
`huggingface-hub`	Loading of pre-trained language models	0.1.0
`torch`	General ML framework	1.11.0
`python`	Programming environment	3.8.6
`datasets`	Loading and processing datasets	1.15.1
`accelerate`	Thread handling on GPU	0.5.1

Table B.1: Packages used in this thesis

B.2 Initializing a new adapter and injecting it

The `adapter-transformers` library (Pfeiffer, Rücklé et al., 2020) makes it fairly simple to work with adapters together with pre-trained language models initialized using the `transformers` (Wolf et al., 2019) library. The code below shows the sections relevant for the adapter implementation in the script that runs the masked language modeling objective over the extracted ConceptNet corpus used in chapter 4. The entire file is located under `src/run_mlm.py` in the provided repository (an url can be found in the abstract).

```
from transformers import (
AutoModelForMaskedLM,
)

model = AutoModelForMaskedLM.from_pretrained(
        args.model_name_or_path,
        config=config,
)

#adapter_config holds the hyperparameters for the adapter modules.
if args.use_adapter:
        adapter_config = AdapterConfig.load(
                args.adapter_config,
                non_linearity=args.non_linearity,
                reduction_factor=args.reduction_factor
        )
        model.add_adapter(args.adapter_name,
                config=adapter_config)
        model.train_adapter([args.adapter_name])
        model.set_active_adapters(args.adapter_name)

if args.tune_all_parameters == True:
        model.freeze_model(False)
        # keep original transformer weights dynamic

        .
        .
        .

unwrapped_model.save_pretrained(
args.output_dir, save_function=accelerator.save)
```

B.3 Setup of AdapterFusion after ST-adapter training

```
'''
Invoke a new AdapterFusion layer over our ST-adapters
'''
from transformers.adapters.composition import Fuse

model = AutoModelForMaskedLM.from_pretrained(
args.model_name_or_path,
config=config,
)

if args.train_fusion:
        adapters = args.adapter_list
        adapter_names = [x.split("/")[2] for x in adapters]
        for adapter in adapters:
                model.load_adapter(adapter, with_head=False)
        obj = Fuse(*adapter_names)
        model.add_adapter_fusion(obj)
        model.set_active_adapters(obj)
        model.train_adapter_fusion(obj)

        .
        .
        .

#Save the AdapterFusion model
unwrapped_model.save_adapter_fusion("./adapters/" + "fusion/",
",".join(adapter_fusion_object))
unwrapped_model.save_pretrained(args.output_dir)
```