Exploring the Robustness of the Natural Language Inference Capabilties of T5

Bachelor's Thesis of

Dennis Grötzinger

at the Department of Informatics
Institute for Program Structures and Data Organization (IPD)

Reviewer:	Prof. Dr.-Ing. Anne Koziolek (Martens)
Second reviewer:	Prof. Dr. Ralf H. Reussner
Advisor:	M.Sc. Jan Keim
Second advisor:	Univ.-Prof. Dr. Gregor Betz

21. February 2021 – 21. June 2021

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

Abstract

Large language models like T5 perform excellently on various NLI benchmarks. However, it has been shown that even small changes in the structure of these tasks can significantly reduce accuracy. I build upon this insight and explore how robust the NLI skills of T5 are in three scenarios. First, I show that T5 is robust to some variations in the MNLI pattern, while others degenerate performance significantly. Second, I observe that some other patterns that T5 was trained on can be substituted for the MNLI pattern and still achieve good results. Third, I demonstrate that the MNLI pattern translate well to other NLI datasets, even improving accuracy by 13% in the case of RTE. All things considered, I conclude that the robustness of the NLI skills of T5 really depend on which alterations are applied.

Zusammenfassung

Große Sprachmodelle wie T5 schneiden bei vielen NLI-Benchmarks exzellent ab. Es hat sich jedoch gezeigt, dass schon kleine Änderungen in der Struktur dieser Aufgaben die Genauigkeit deutlich verringern können. Ich baue auf dieser Erkenntnis auf und untersuche, wie robust die NLI-Fähigkeiten von T5 in drei Szenarien sind. Erstens zeige ich, dass T5 gegenüber einigen Variationen im MNLI-Muster robust ist, während andere die Leistung deutlich verschlechtern. Zweitens beobachte ich, dass einige andere Muster, auf die T5 trainiert wurde, das MNLI-Muster ersetzt können und dennoch gute Ergebnisse erzielen. Drittens zeige ich, dass das MNLI-Muster gut auf andere NLI-Datensätze übertragbar ist und die Genauigkeit im Fall von RTE sogar um 13% verbessert. Alles in allem komme ich zu dem Schluss, dass die Robustheit der NLI-Fähigkeiten von T5 stark davon abhängt, welche Änderungen vorgenommen werden.

Contents

List of Figures

List of Tables

1 Introduction

To reason is one of the most fundamental and awe-inspiring capabilities of humankind. It is perhaps one of the most significant skills that sets us apart from other animals and has allowed us to establish complex philosophical arguments, develop beautiful mathematical theories and create marvels of engineering. It is therefore not very surprising that the concept of reasoning itself is something many of us like to think and ponder about extensively. And in that line of thought, it is most natural to wonder if we can create something that can reason as we do and perhaps even surpass our capabilities. While computers and their internal logic gates can be seen as doing some kind of inference (and are certainly very fast at those), it is still not the same type of flexible reasoning that we humans are capable of. In recent years, it has become increasingly prevalent to try to develop systems that can mimic this kind of reasoning. The first attempts began as early as the 1950s with what we now call symbolic AI [19]. The primary goal was to explicitly encode handwritten rules into computers to try to distil our abilities. This proved much more difficult than originally thought and from the 1990s research shifted to statistical models. In particular, once computing capacity paved the way for the widespread adoption of neural networks at the turn of the millennium, they caught on and continue to dominate the field to this day [62].

Although we are probably still far from creating a reasoning machine as flexible and general as humans are [36], at this stage we do have some proxies for inference, on which modern neural networks perform quite well. In Natural Language Inference (NLI), a model is given a premise as well as a hypothesis and is asked to predict whether or not the hypothesis follows from the premise. NLI is a subfield of the much broader area of Natural language understanding (NLP) [9].

In recent years, one neural network architecture in particular has begun to eclipse all other methods. It is called Transformer [56] and utilizes a mechanism called attention. Figure 1.1 shows the accuracy of models on a particular NLI dataset, called Multi NLI [63], over time. There is a sharp increase in accuracy between the last non-Transformer model (GenSen [54]) and the first Transformer model (Finetuned Transformer LM [41]). Many networks from that point on were Transformers in NLP [64].

By now there are many variations of the Transformer architecture. In this thesis I will focus on one particular, called Text-to-Text Transfer Transformer [43] or T5 for short. One of the main contributions of this architecture is that it casts every NLP task in a text-to-text format. This means that instead of having different output layers for every task like some architectures do, T5 outputs a sequence of words for every task. For classification tasks like Multi NLI, this means that T5 is trained to output a specific word for every available label. For example, given the premise "At the other end of Pennsylvania Avenue, people began to line up for a White House tour." and the hypothesis "People formed a line at the end of Pennsylvania Avenue.", T5 would be trained to output the string "entailment".

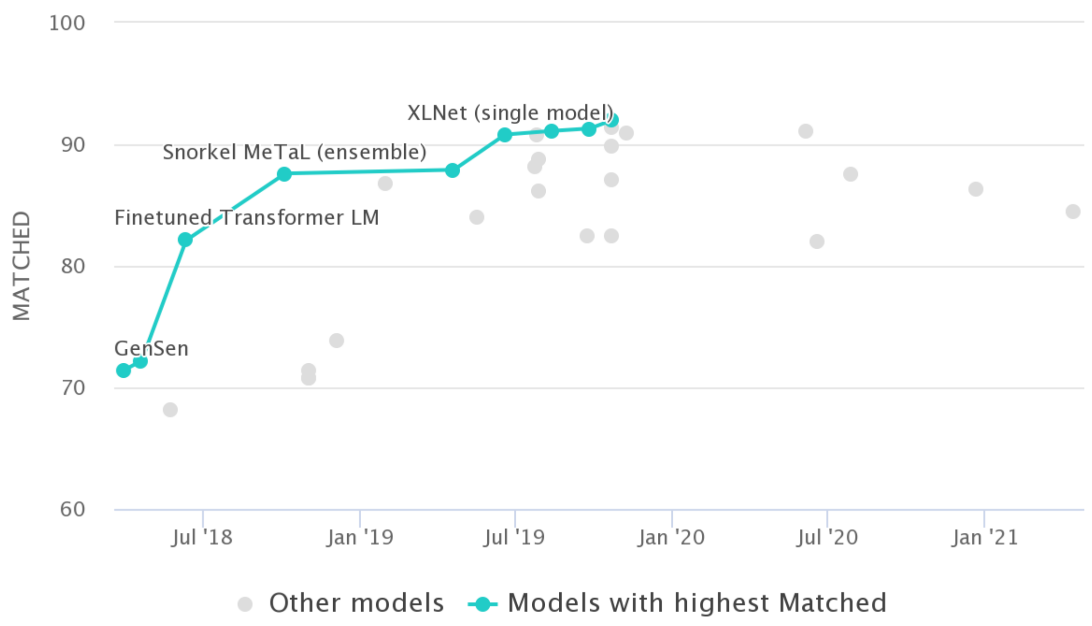


Figure 1.1: Model with the highest accuracy on the MNLI validation_matched dataset over time. [39]

However, T5 is not limited to output only one word. Some tasks, such as summarising, require T5 to output a sequence of words that can be as long as several sentences.

In order for T5 to recognize which task it has to solve, the task instances get embedded in a pattern ([47],[49]). This simply means that a few words are added which indicates the task that has to be solved. For the former example, this could mean that T5 is given the string "mnli hypothesis: People formed a line at the end of Pennsylvania Avenue. premise: At the other end of Pennsylvania Avenue, people began to line up for a White House tour." as input.

The tasks T5 was trained on do not each test for a completely unique skill. Instead, some tasks test for the same or partially overlapping skills. Therefore, it is possible to use the pattern of one task to test skills for another task. For example, since T5 was also trained on question answering, one may adapt the example to

> "question: Does the hypothesis follow from the premise? context: hypothesis: People formed a line at the end of Pennsylvania Avenue. premise: At the other end of Pennsylvania Avenue, people began to line up for a White House tour."

If T5 is able to answer questions and can also solve NLI tasks, it should be able to solve this combination too.

However, it is still an open research question whether this is the case. For example, it might be that all instances of the question-answering task that T5 was trained on are phrased in a particular way (i.e. all questions beginning with "What") and therefore cannot be generalized to the above combination.

Of course, this is not the only combination one can come up with. Instead, there are many ways in which one can change patterns, combine task and use different evaluation methods. If T5 can still perform well with such alterations, it would be robust. If not, it would be brittle and might exploit certain patterns that can serve as unintended shortcuts [18].

In this thesis, I will explore those questions. By extensive investigation, my goal is to provide an intuition on the question if and when T5 is robust to various perturbations in the context of NLI.

The main contributions and findings of this thesis are as follows: In Chapter 2 I provide an extensive overview of the relevant background information for my thesis. With this in mind, I explore the robustness of T5 in the context of NLI in three main experiments in Chapter 4. In Section 4.4 I show that T5 is sometimes robust to slight changes in the MNLI pattern. Then in Section 4.5 I observe that me patterns that T5 was trained on can also achieve good accuracy with MNLI. Furthermore, I detect that T5 might partially rely on statistical cues for its MNLI prediction. Finally, in Section 4.5 I show that the MNLI capabilities of T5 also translate to other NLI datasets, in the case of RTE even improving accuracy by 13% compared to the RTE pattern.

2 Background and Related Work

In this chapter, I will provide a foundation of concepts and terminology that I will use in later chapters. Section 2.1 will explain what natural language processing (NLP) is and common tasks that are associated with it. In Section 2.2 I will further explain a subdivision of NLP, natural language inference (NLI), and the most important datasets that come with it. Section 2.3 will be all about natural language models (NLM) and explain core concepts. In Section 2.4 I will explain the Input-Ouput format of one specific LM, called T5. Section 2.5 will describe different methods of how text can be generated with LMs. In Section 2.7 I will define what statistical cues are and how they are relevant for the following chapters. Section 2.6 will provide some terminology that should help to make later chapters easier to understand. Finally, Section 2.8 will give an overview of all the previous work that is related to mine.

2.1 Natural Language Processing

Natural language processing can be defined as "a theory-motivated range of computational techniques for the automatic analysis and representation of human language" [65]. It is a very broad area of research that ranges from speech recognition to named entity recognition to machine translation and everything in-between. Due to the nature of the models I am working with, I will only consider NLP tasks in my analysis that take text as an input and expect the model to either generate more text (also called natural language generation [4]) or label the input (also called natural language understanding [65]).

As there are many NLP tasks used in this thesis, I will explain the most important ones for this thesis here so that it is easier to follow them in the chapters to come. I will also include some datasets and benchmarks that belong to the different tasks T5 was trained for. In the next subsection, I will give a detailed overview of all these datasets.

Summarization. Summarization can either be done extractive or abstractive [69]. In extractive summarization, the goal is to select text fragments from the text that are highly relevant to describe the core content. In abstractive summarization, the model has to generate the summary itself. T5 was trained to do abstractive summarization with the CNN/Daily Mail dataset [50].

Sentiment Analysis. In sentiment analysis, the goal is to determine the sentiment of a text span, i.e., whether a text span has more positive or more negative connotations. T5 was trained for this task using the SST-2 dataset [53].

Paraphrasing/Sentence Similarity. The goal here is to either detect whether two text

Dataset	Description
CoLA	Determine if a sentence is grammatically acceptable
RTE	Determine if one sentence entails or contradicts another
MNLI	Determine if one sentence entails, contradicts or is neutral to another
MRPC	Determine if two sentences are paraphrases of each other
QNLI	Determine if a question can be answered given a context
QQP	Determine if two questions are duplicates
SST2	Determine if the sentiment of a sentence is positive or negative
STSB	Determine how semantically similar two sentences are on a scale from 1 to 5
CB	Determine if one sentence entails, contradicts or is neutral to another
SQuAD	Determine the answer to a given question and context

Table 2.1: All the datasets that T5 was trained on and a short description of each.

segments are paraphrases of each other or how semantically similar the text segments are. The paraphrase datasets on which T5 was trained are QQP[15] and MRPC [12]. The semantic similarity dataset that T5 was trained on is STS-B [8].

Natural Language Inference. NLI is the task I will mainly deal with in this paper. Given a premise and a hypothesis, the goal of NLI is to determine whether or not the premise entails the hypothesis. For some datasets, it is also necessary to determine if neither is the case and the premise and hypothesis are neutral to each other. NLI datasets on which T5 was trained include MNLI [63], CB [32], and RTE [10]. While RTE only has the labels entailment and contradiction, MNLI and CB also have the label neutral.

Additionally, I provide two tables which serve as an easy-to-access overview. Table 2.1 shows all the datasets that T5 was trained on and a brief description of each corresponding task. Furthermore, Table 2.2 shows the same datasets, but this time with a small example of each.

2.2 Natural Language Inference Datasets

As the investigation in this thesis evolves around testing the robustness of T5 for NLI tasks, it is worthwhile to present the different NLI datasets that I will be using in more detail. In the last section I already mentioned MNLI [63], CB [32] and RTE [10]. Those are the NLI datasets that T5 was likely trained on. There are however more NLI datasets, T5 likely was not trained on, and that will still be important in my analysis. Since the T5 model from huggingface [64] that I am using does not specify on which datasets it was trained on, I will assume that it was trained on the same ones as mentioned in the T5 paper [43]. I will continue by listing all the NLI datasets used in this thesis and their corresponding core characteristics.

Multi Natural Language Inference. Multi natural language inference (MNLI) [63]

Dataset	Example Instance	Corresponding Label
CoLA	"The inspector analyzed the soundness in the building."	unacceptable
RTE	premise: "The fifth World Social Forum (WSF) has kicked off in Porto Alegre, Rio Grande do Sul state, Brazil." hypothesis: "The WSF takes place in Brazil."	entailment
MNLI	premise: "As the organizations became more results-oriented, they often" hypothesis: "They wanted to be the very best."	neutral
MRPC	sentence1: "The identical rovers will act as robotic geologists , searching for evidence of past water" sentence2: "The rovers act as robotic geologists , moving on six wheels ."	not_equivalent
QNLI	question: "The period of time from 1200 to 1000 BCE is known as what?" context: In the Iron Age	entailment
QQP	question1: "Can I block someone from following me on Quora?" question2: "How can I block people on Quora?"	duplicate
SST2	"his supple understanding of the role"	positive
STSB	sentence1: " A man is cutting a potato." sentence2: "A woman is cutting a tomato."	1.25
CB	premise: "B: Yeah. How about Mister Rogers, is he still around? A: Yes. Yeah. They still show Mister Rogers. I don't think he's making new ones," hypothesis: "Mister Rogers is making new Mister Rogers"	contradiction
SQuAD	question: "What is the term for a task that generally lends itself to being solved by a computer?" context: "Computational complexity theory is a branch of the theory of computation in theoretical computer science that focuses on classifying computational problems according to their inherent difficulty, and relating those classes to each other. A computational problem is understood to be a task that is in principle amenable to being solved by a computer, which is equivalent to stating that the problem may be solved by mechanical application of mathematical steps, such as an algorithm."	computational problems

Table 2.2: All the datasets that T5 was trained on with corresponding examples.

is a dataset that provides a set of premises and hypotheses and their corresponding labels. The labels can be either entailment (in case that the premise entails the hypothesis), neutral (in case that the premise and the hypothesis are logically independent of each other) and contradiction (in case the premise contradicts the hypothesis). In addition, the instances in the dataset all have a specific genre. For example, the genre "government" includes only instances that are related to a government or "travel" where the instance has to be related to traveling. While the authors of MNLI used ten genres in total, only five of them are present in the training data. This allows for the construction of two validation sets. The matched validation set contains instances that belong to the same genre as the genres in the training set. In contrast, the mismatched validation set includes all the genres that are not in the training set. The model with the best accuracy currently is the largest version of T5 (T5-11B), with a matched validation accuracy of 92% and a mismatched validation set accuracy of 91.7% [43].

Stanford Natural Language Inference. Stanford natural language inference (SNLI) [5] is the predecessor of MNLI. The instances are therefore of the same form. However, compared to MNLI, SNLI only has one genre, as all the instances were collected from image captions. This makes SNLI easier than MNLI and the SNLI. The best model with the best accuracy currently is EFL [59] with a test accuracy of 93.1%.

Recognizing Textual Entailment. Compared to MNLI and SNLI, Recognizing textual entailment (RTE) [10] only provides the labels entailment and contradiction. Currently, the best model is DeBERTa-1.5B [20] which achieves a test accuracy of 93.2%.

Commitment Bank. Commitment Bank (CB) [32] has the same objective and the same labels as SNLI and MNLI. The difference is that the instances were deliberately chosen to be confusing. The authors achieve this by surrounding the premise with entailment canceling operators such as entailment, but in such a way that they do not change the associated label. For example, one instance is the premise "Richard Breeden hadn't noticed that his new desk had just four telephone lines and one phone ." and the hypothesis "Richard Breeden's new desk had just four telephone lines and one phone" with the label entailment. Here, the "hadn't noticed" is supposed to confuse the model.

Adversarial Natural Language Inference Modern language models achieve very good results for all the NLI datasets mentioned above [30] [43] (in the range of human performance). However, they might rely on spurious statistical patterns to solve the tasks [33] instead of actual understanding. To tackle this shortcoming, the authors of [38] propose to construct adversarial challenge datasets that expose those patterns and eliminate them from the data. They call this dataset adversarial natural language inference (ANLI). This is done in an iterative, human-in-the-loop fashion. First, humans create instances that the current models do not recognize correctly by eliminating certain statistical patterns. Then the model is fine-tuned on those new instances. For this model, humans again create instances that the model labels wrong. This procedure then gets repeated, which results in both stronger models and more challenging testsets.

2.3 Natural Language Models

In this section, I will present different methods that can solve NLP tasks.

In the past, many ways have been proposed to solve problems in NLP. Bag-of-word models [67] have been used to create sparse representations of text in order to be used as features for statistical analysis [45]. More recently, dense vector representations of text that are generated by training on a text corpus has been developed. This enabled the representations to approximate semantic or syntactic relationships [45]. The most iconic method for those word embedding probably is word2vec [35] [34] that introduced some core concepts and made training a lot more efficient [45].

In 2013 neural networks started to become widespread in NLP. The main architectures choices were recurrent neural networks (RNNs) [13] and long short-term memory (LSTM) [21]. Even convolutional neural networks (CNN), which are mostly used in computer vision, started gaining some traction [25].

In 2015 the concept of Attention got introduced [3]. Attention allows one part of a network to focus selectively on the most important information of another part of the network. In 2017 the landmark paper "Attention is all you need" [56] demonstrated that an architecture that solely relies on Attention can outperform those that only use Attention as a subcomponent (The blog post [2] provides an excellent explanation of the Transformer). This purely Attention based architecture was named "Transformer". It sparked a revolution in NLP, with many variations being developed to this date.

In the following subsections, I will explain some basic concepts around the Transformer architecture applied to NLP (although some parts are also applicable to other architectures). As a common subdivision of Transformer language model variations is in autoregressive models, autoencoding models and sequence-to-sequence models, I will go into more detail in the respective subsections.

2.3.1 Common Concepts.

Some concepts apply to all Transformer variations. I will list and explain them in this subsection.

Token/Tokenizer. No neural network can do calculations on words themselves. Instead, the words have to be represented as numbers so that the model can perform calculations. Those numbers are called tokens, and the methods to generate the tokens from text are called tokenizers. There are many tokenizers, all of which have a number of advantages and disadvantages. Some simply map each word to one token (WordLevel) while others are trained on a text corpus to map frequently occurring character sequences to one token (i.e. Byte-Pair-Encoding (BPE) [51]). For example, instead of mapping the word "rainbow" to one token, BPE might map them to two tokens, one that represents "rain" and one that represents "bow". The text "rainbow rain bow" could then be represented with only two tokens, where it would take three when encoding the individual words. For natural languages, this means that the amount of vectors can be decreased substantially with methods like BPE compared to word level embeddings.

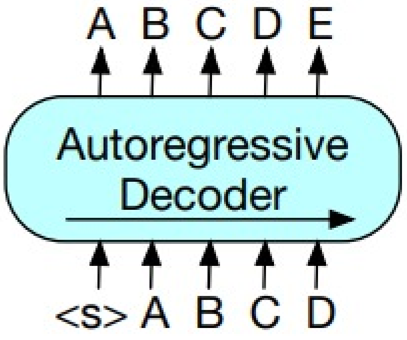


Figure 2.1: Illustration of an autoregressive model [27].

Vocabulary. The vocabulary of a tokenizer/model is the set of all tokens that the model is trained on/the tokenizer can encode. A typical cardinality of a vocabulary is in the order of 10^4. I will denote T as the symbol for the vocabulary.

Input Format. Every Transformer architecture gets a sequence of tokens as input. The Transformer then does internal calculations with them. In the best case, those calculations then lead to a representation of the input that is useful for solving various NLP tasks.

Attention. The attention mechanism is the defining feature of the Transformer architecture. It is used in the internal calculations. On a high level, attention means that the part of the Transformer that corresponds to one token can decide how much information it wants to get from a part that corresponds to another token. To illustrate, consider the sentence: "The Lion sleeps on the ground because it is tired". For a human, it is clear that the word "it" must refer to "the lion" and not to "the ground". However, implementing which word "it" is referring to algorithmically is rather challenging. Attention tackles this issue by establishing an association between "it" and "lion", where "it" gets a lot of information from "lion" and vice versa.

Output Format. In order to solve a specific task, Transformers typically have a head on top of their layers stack. A head most commonly describes a fully connected layer with as many outputs as there are classes that one wants to differentiate. For example, the classes could be all the token in the vocabulary when trying to generate text or some task specific classes, i.e. entailment, neutral and contradiction for MNLI. It is then possible to calculate the softmax [6] over the values of all the classes that the model predicts. This enables the interpretation as class probabilities.

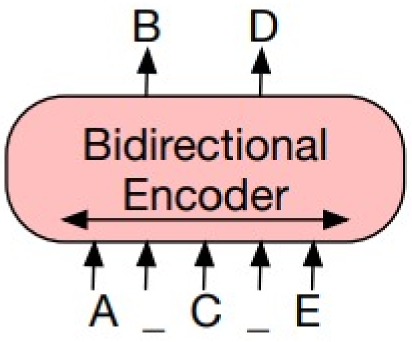


Figure 2.2: Illustration of an autoencoding model [27].

2.3.2 Autoregressive Models

Autoregressive or decoder-only models take a sequence of tokens as input and predict the next token. In Figure 2.1 you can see that the model takes <s> (the start token) and A as input and predicts B. It is now possible to take the predicted token and append it to the input. Now the model has <s> A B as input and predicts C. For a more concrete example, think of A= "The", B="cat" and C="sits". Using this technique, autoregressive models are very well suited to generate text given an initial prompt.

Since the model always just predicts the last token, each token can only attend to itself and to previous tokens (masked self-attention). This speeds up training time compared to the other types of models as it can make many predictions with just one forward pass.

A common example for an autoregressive model is GPT-2[42] or GPT-3 [7].

2.3.3 Autoencoding Models

Autoencoding or encoder-only models are most commonly used to classify/"understand" text as opposed to generate new one. They are pretrained by corrupting part of the input (for example masking it) and then trying to reconstruct the full input. For example, in Figure 2.2 B and D of the input sequence is masked, and the model is asked to predict those. Concretely, if A = "The", C = "sits" and D = "the roof", the model might be asked to fill in the gaps as B = "cat" and D = "on".

After (or during) said pretraining, the model is typically trained on a downstream NLU task. This can mean that the model gets a new head that only maps to certain classes (i.e. to entailment, neutral and contradiction for NLI).

Compared to autoregressive models, the attention mechanism is not masked, which means that every token can attend to every other token in the input.

Some examples of autoencoding models are BERT[11], ALBERT [26] or RoBERTa [30].

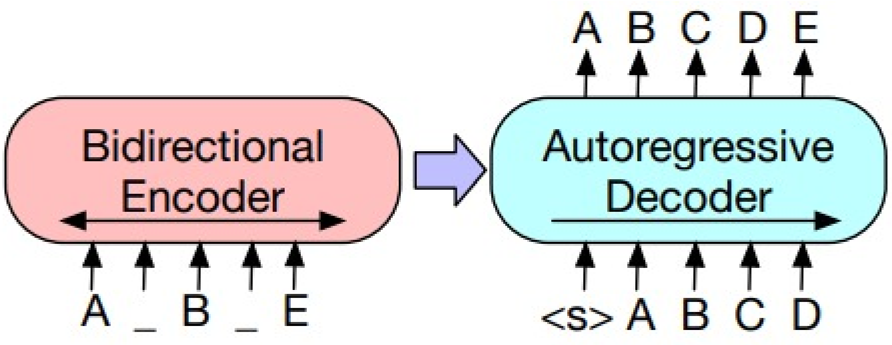


Figure 2.3: Illustration of a sequence-to-sequence model [27].

2.3.4 Sequence-To-Sequence Models

Sequence-to-sequence or encoder-decoder models combine the encoder-only and the decoder-only models. As seen in Figure 2.3 the input gets first fed into the encoder and then the decoder can attend to the output of the encoder.

This type of architecture is most commonly used when transforming the input to another text. The most prominent examples of that are translation and summarization.

Some famous examples of sequence-to-sequence models are T5 [43], BART [27] or PEGASUS [66]. In this thesis, I will be using T5.

2.4 T5

T5 is a sequence-to-sequence model released by Google [43]. T5 introduces a unified text-to-text framework which allowed for a systematic study of various approaches. As I will be using T5 for the experiments in this thesis, I will dedicate this subsection to outline some of its important properties.

Text-To-Text Format. One of the key selling points of T5 is its unified text-to-text framework. This means that T5 always gets text as input and is asked to output some other text. With this technique, every task that T5 is trained on, be it unsupervised pretraining or supervised training for downstream tasks, can be represented consistently.

Unsupervised Denoising Training.T5 was pretrained by unsupervised denoising training. This means that T5 was given text where a part was masked, and the model was asked to predict the masks. For example, in "The mask1 walks in mask2 park.", the model was trained to predict mask1 = "cute dog" and mask2 = "the". In the text-to-text framework, this would be represented as input = "The <extra_id_0> walks in <extra_id_1> park" and output = "<extra_id_0> cute dog <extra_id_1> the <extra_id_2>". The extra_id token serve as a mask token. This unsupervised denoising training was done on the colossal clean crawled corpus(C4).

Colossal Clean Crawled Corpus. To train models like T5, a lot of text is necessary.

The authors in [43] therefore curate their own dataset called the colossal clean crawled corpus (C4). It consists of massive amounts of text scraped from the internet. To improve the quality, the authors applied various cleaning strategies like removing code or deduplicating.

Supervised Training. The authors chose to represent the input with task-specific patterns for supervised downstream task training. For example, if T5 should translate from English to German, the input will be input = "translate English to German: The house is wonderful." and output = "Das Haus ist wunderbar.". For summarization the pattern is "summarize:" and for english french translation "translate english french:". The complete list of tasks and corresponding pattern that I will be using in this thesis is shown in Table 2.2.

Multi-Task Training. Originally, the authors of T5 chose to only train T5 for one downstream task at a time. This means that there needs to be a different T5 model for every task, but this also achieves very good results. In contrast, the people who trained T5 for the huggingface [64] model used multi-task training. This means that T5 gets trained on a lot of tasks simultaneously. The results are not as good as with single-task training, but this means that only one T5 model is needed for every task. I will be using the model from huggingface for my experiments.

Versions.T5 comes in different version that differentiate by the amount of parameters they have. Generally speaking, the more parameters the model has, the better the performance of the model. In my experiments, I will be using the T5-base model. T5-base has 220 million parameters in total (in comparison, the largest T5 model available, T5-11B, has 11 billion parameters). T5-base is structured in 12 decoder and 12 encoder blocks, with each block comprising the self-attention, the feed-forward and the optional encoder-decoder attention network [43].

2.5 Decoding Algorithms

When the decoder of a model does a prediction, it outputs how likely it is for each token to follow the given text sequence (see Subsection 2.3.1). Given this probability distribution, the actual next token in the sequence has to be chosen. While it would be straight-forward to simply choose the token with the highest probability as the final prediction, this has been proven suboptimal for many applications [23]. In fact, this decoding scheme, also called greedy-decoding, leads to a degeneration when using it to generate long texts. Degeneration means that the model repeats one token or a small sequence of tokens over and over. While it is an open question why greedy-decoding leads to text degeneration [61], by now there are multiple techniques to combat this issue.

In the following, I will introduce the most common decoding schemes. To further illustrate which tokens the individual decoding schemes choose, I provide a toy example of output probabilities in Figure 2.4. In this toy example, there are only five tokens that T5 can output the probabilities for.

Greedy Decoding. Greedy decoding always chooses the most probable token for every

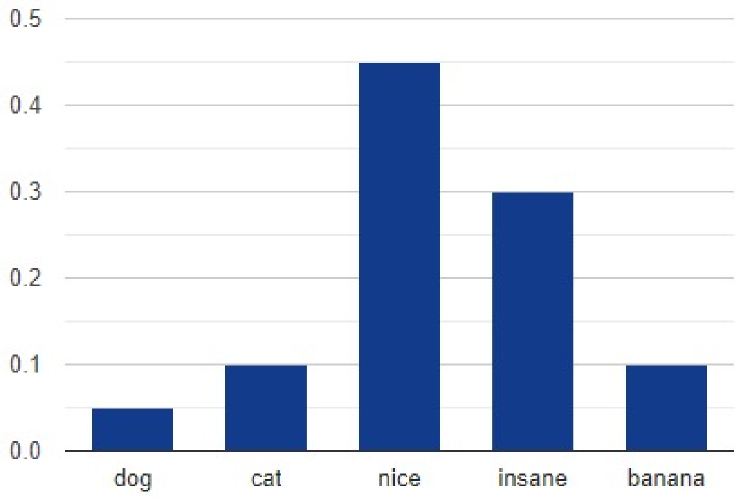


Figure 2.4: A toy example of the output probabilities for individual tokens of a language model.

prediction. This leads to degeneration when generating long text, but is most applicable when only generating one token. In the toy example in Figure 2.4 greedy decoding would choose the token corresponding to "nice".

Sampling. Sampling means that the next token gets randomly chosen according to its probability. In Figure 2.4 this would mean that there is a 5% chance that "dog" will get picked, a 10% chance for "cat", a 45% chance for "nice", a 30% chance for "insane" and a 10% chance for banana.

Top-k Sampling. Top-k sampling follows the same principle as sampling, with the difference that only the k most likely token are taken into consideration and the probability is spread between them[14] [22]. For example, when choosing k=2 for the toy example of Figure 2.4, there would be a 53% chance that "nice" will be the next token and a 47% chance of the token "insane". All other tokens would have a probability of 0%.

Nucleus/Top-p Sampling. Top-p sampling is the same concept as top-k sampling, but instead of taking the k most probable choices into account, top-p sampling instead takes the choices into account that combined have a probability larger than p [23]. For example, when choosing p = 90% one would only sample between the tokens "cat", "nice", "insane" and "banana" as they are the smallest amount of tokens with their combined probability (10% + 45% + 30% + 10% = 95%) being larger than 90%.

Beam Search. If the goal is to find the most probable text sequence, greedy decoding can find suboptimal solutions, as very likely text sequences can start with a token that is not the most likely choice. For instance, consider the extension of the toy example, illustrated in Figure 2.5. It shows the probability that a language model gives to various tokens, depending on which token was chosen earlier. Instead of only predicting one

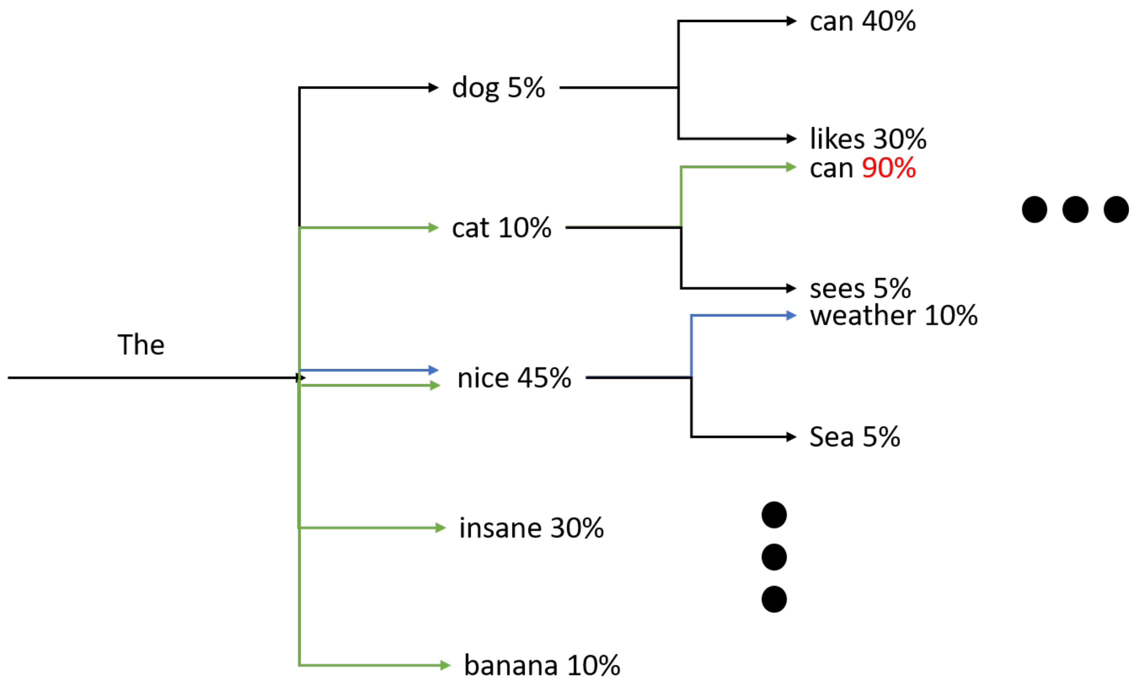


Figure 2.5: A toy example to illustrate how beam search works. Two time steps are shown with the probabilities that a language model assigns certain tokens, given the input before. The blue line demonstrates which result greedy decoding would choose, while the green line shows the result that beam search discovers (which has a higher probability than the one from greedy decoding).

token, I now consider the case where the goal is to predict two tokens and the input text of "The". As already discussed, greedy decoding would choose the token corresponding to "nice" in the first step. This means that in the second step, it would be only possible to choose between "weather" and "Sea". The highest probability of a two token sequence would therefore be $45\%*10\% = 4.5\%$ (shown in blue in the illustration). In contrast, beam search [16] keeps the most likely number of hypotheses (or beams) at each time step (shown in green in the illustration). With four beams, this means that after the first step the tokens "cat", "nice", "insane" and "banana" are kept. In the next step, beam search then analyses the next most likely token for every hypothesis. This means in the toy example that beam search discovers the token "can" after "cat", which results in the probability of $10\%*90\%=9\%$. This is a sequence resulting in double the probability of the sequence that greedy decoding found.

While beam search will always find a solution that is more or equally probable than the greedy solution, it is still a greedy algorithm and not guaranteed to find the most likely sequence.

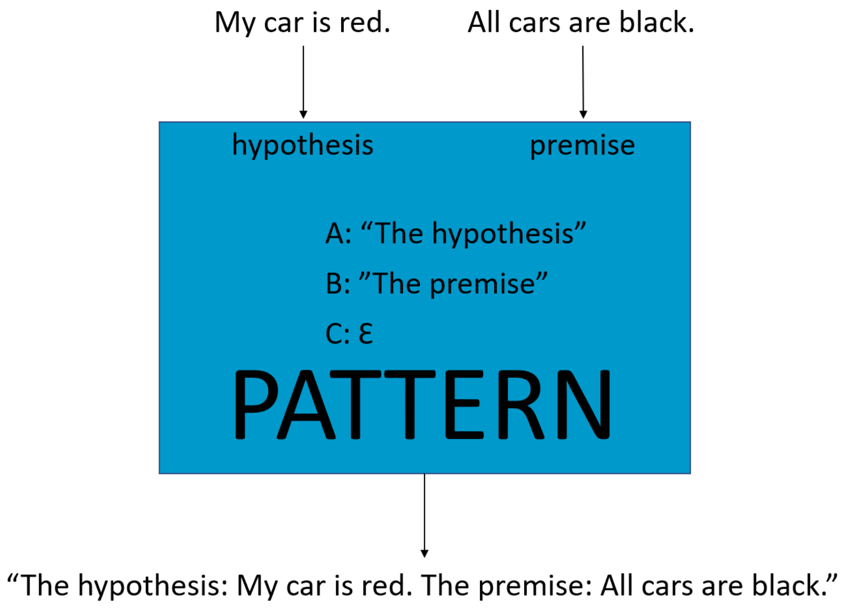


Figure 2.6: An example of a pattern. The pattern is parametrized with A="The hypothesis:", B=" The premise: " and C= ϵ (ϵ is the empty string). It takes a premise and a hypothesis as input.

2.6 Patterns and Verbalizers

Since most of my work will be related to adapting the input in different ways, a consistent notation is necessary. For better comparison, I will use the notation of previous, similar work [47] [49] [55].

Let $D = (X, Y)$ be the trainset of a dataset (typically MNLI for this thesis) with X being the inputs (for MNLI an input would comprise a premise and a hypothesis, $x_i = (premise_i, hypothesis_i) \forall x_i \in X$) and Y being the labels (for MNLI: $y \in Y \Rightarrow y \in \{entailment, neutral, contradiction\}$). Furthermore, let M be a language model (typically T5), T its vocabulary and T^* the set of all token sequences.

Pattern. A pattern P is a function $P : X \longrightarrow T^*$ that maps dataset inputs to the actual encoder input. For MNLI this would be $P : (hypothesis, premise) \longrightarrow A\ hypothesis\ B\ premise\ C$ where $A, B, C \in T^*$. Figure 2.6 provides an illustrated example of a pattern.

Verbalizer. A verbalizer v is a function $v : Y \longrightarrow T^*$ that maps each label to a sequence of tokens. Figure 2.7 provides an illustrated example of a verbalizer.

Together, a pattern P and a verbalizer v form a $pattern - verbalizer - pair$ (PVP).

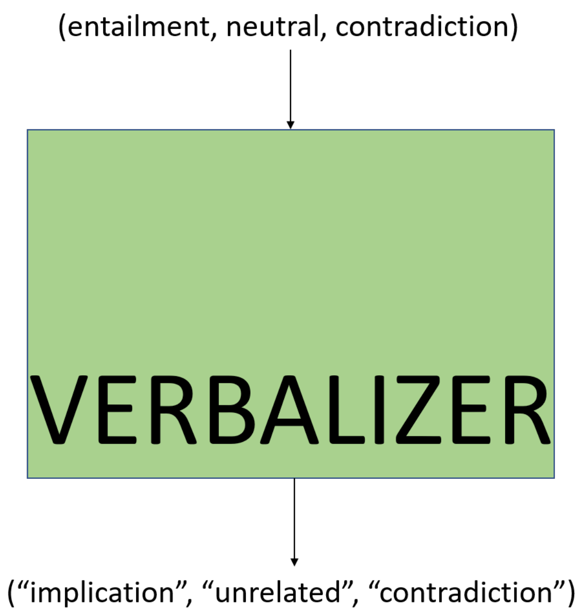

(entailment, neutral, contradiction)
VERBALIZER
("implication", "unrelated", "contradiction")

Figure 2.7: An example of a verbalizer. The verbalizer takes three labels (entailment, neutral, contradiction) as input and outputs the strings (or tokens) that correspond to those labels.

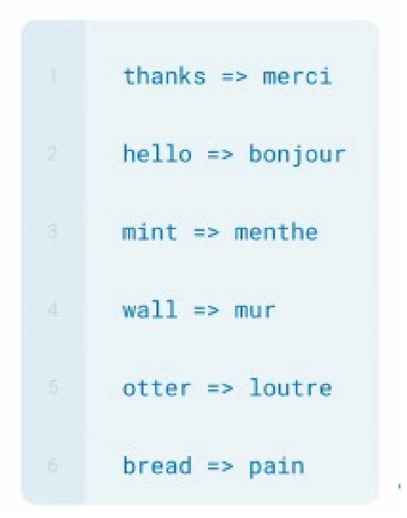
2.7 Statistical Cues

While neural networks nowadays achieve remarkable results in many areas, it is often unclear how those results should be interpreted. For example, while NLI is a proxy for actual reasoning and neural networks can achieve a very high accuracy on various NLI tasks [43], this does not mean that the models have to have actual reasoning capabilities. Instead, they could solve the task due to the existence of shallow statistical heuristics/cues in the dataset [33].

A statistical cue is a feature in the dataset which is highly predictive for a certain output, and thus allows the model to solve a task by relying on this cue. In NLI, an example would be if for most instances in the dataset that are labeled contradiction, there would be the word "not" in the hypothesis. If the model then learns to associate the existence of "not" with the label contradiction, it can achieve a high accuracy.

The problem is that we humans know that the existence of "not" has nothing necessarily to do with the associated label. And if we construct examples where a "not" in instances of the label entailment, the models' accuracy would drop significantly.

That statistical cues are a problem has been shown in many fields that utilize neural networks, including NLI [33] [37] [46], image recognition [58] or visual question answering [1].



```
thanks => merci
hello => bonjour
mint => menthe
wall => mur
otter => loutre
bread => pain
```

Figure 2.8: An example of how to do few-shot learning for English to French translation from [7].

2.8 Related Work

In recent times (and certainly since the introduction of GPT-3 [7]) there has been an increase in interest of how to construct patterns (or prompts) for neural language models [68] [24] [17] [28]. GPT-3 [7] introduced the concept of few-shot learning, where the model is given a few examples of the task in the pattern (see Figure 2.8). In contrast to regular fine-tuning, this has the advantage that no parameter updates have to be calculated. Furthermore, it is possible to use one and the same model to perform various tasks with a simple natural language interface.

However, recent work showed that the language models are very sensitive to the exact wording of the pattern [24] [68]. Sometimes even seemingly trivial choices like the order of the examples in the few-shot pattern made the difference between near random or near state-of-the-art performance [31]. These insights lead to the exploration of two main directions:

1. Adapting the patterns such that they increase the performance of the language model.

2. Exposing the brittleness of language models with regard to the patterns and implementing strategies to mitigate them.

By now, there are numerous methods for the first point. In [24] the authors use a mining and paraphrasing based approach to improve the knowledge extraction capabilities of BERT [11]. The authors of [17] use T5 [43] to automatically generate a variety of patterns and test each of them to find the ones that perform well. While this automates the generation method, it still has the problem that there are far to many patterns to try them all. Instead, the authors in [28] [40] and [52] consider the continuous embedded vector representation of the input and use gradient updates to efficiently search for good patterns. Finally, Schick and Schütze use so-called pattern-exploiting training (PET) in [47] [49] [48] to reformulate down-stream tasks as cloze questions. For point two (revealing brittleness) the authors of [31] note that GPT-3 is very sensitive to the order of the few-shot examples and use artificially generated datasets to find good permutations. The authors of [29] find that GTP-3 is very sensitive to the examples used in its few-shot prompt, and search for semantically similar examples to tackle this issue. Finally, the authors of [68] find that GPT-3 is biased towards examples that are at the end of the pattern, that are frequent in the pattern, or towards answers that are highly common in the training data. They use a content-free test input to estimate the bias of GPT-3 for a given pattern and use this for calibration.

My thesis is related to question two, albeit in a different context. First, I use T5 [43] which is a sequence-to-sequence model. In contrast, most work mentioned above uses either autoregressive or autoencoding models. Second, I do not employ automated methods for pattern generation and instead rely on hand-crafted ones. The reason is that I do not aim to increase the performance of T5 and instead explore the robustness properties that a human can comprehend. Third, while the previous work used many tasks to test their methods, I only explore the NLI capabilities of T5 with particular attention to MNLI (see Section 2.2). This is because of the limited time resources of my bachelor's thesis and the special interest of NLI in my lab. Finally, since T5 was already fine-tuned to perform specific tasks, there already are patterns that perform well and there is no need to explore the entire space of possible token sequences like it is the case in GPT-3. Instead, I focus my analysis on the already given task-specific patterns and how combinations or variations of them affect the NLI capabilities of T5.

3 Preliminary Results With GPT-2

When first starting out with my thesis, I had different goals than what I am now presenting. In this chapter I want to explain what those goals were, why I could not achieve them and the rationale behind what I did instead.

3.1 Initial Goals

Choosing which pattern and verbalizer to use is a question that has to be addressed with any language model, that takes natural language as its input. Some PVPs might lead to state-of-the-art performance for certain tasks, while others might be no better than random. Therefore, my original goal was to search for prompts that lead to a good accuracy on NLI tasks.

While being applicable to all language models, I chose to use GPT-2 for my experiments. The decision was based on the fact that my lab is doing research mainly on GPT-2 and therefore results in this thesis might show to be valuable for other projects. Also, the GPT-2 models available on huggingface [64] are not fine-tuned on any tasks. Therefore, there also is no particular pattern that it was trained with (which would limit the variety of patterns).

While there also is a newer, larger version of GPT-2 available, called GPT-3, this model is not available for the broad public. Even if it were, it would be too computationally expensive to handle with my limited resources. Nevertheless, I thought that even in GPT-2 there is a lot of room for exploration.

3.2 Challenges

As it turned out, using GPT-2 came with a major drawback. While GPT-2 is excellent in natural language generation tasks, performance on natural language understanding which includes NLI is rather poor in existing literature [42]. In fact, I found no existing literature that showed that GPT-2 had above random accuracy on any NLI task without fine-tuning the model. However, this does not mean that it is impossible for GPT to solve those tasks without fine-tuning. In fact, it might just be that it only needs the right PVPs to achieve good results.

However, after hand-crafting a lot of different PVPs and trying different evaluation methods, I found none that exceeded a random baseline. With this observation, I can reasonably conclude that by only using handcrafted PVPs, one is not able to achieve good results for NLI with a GPT-2 version that is not fine-tuned.

The options that I saw on how to continue next were either to fine-tune GPT-2 myself (as I found no GPT-2 versions online that are fine-tuned on a NLI dataset), use automated methods to search for patterns, use a task other than NLI, or switch to a model that already is fine-tuned. I decided to go with the last option.

3.3 Adapted Goals

As I did not find any other autoregressive models that perform well on NLI, I chose the next closest architecture which are sequence-to-sequence models (see Section 2.3 as they still have a decoder, which is totally omitted in autoencoding models). I selected to use T5-base [43] for my analysis.

However, as the version of T5 that I am using is already fine-tuned on NLI datasets using specific patterns, this meant that I also had to change my objective. Just trying hand-crafted patterns that have nothing to do with the patterns that T5 was trained on was unlikely to achieve good results. Instead, I decided to evaluate the robustness of the NLI capabilities of T5 to different perturbations and combinations of patterns that it was already trained on.

4 Experiments With T5

In this chapter, I will outline the main contributions of my thesis. First, in Section 4.1, I will outline different evaluation schemes that I will be working with. The method *string comparison* is the one the authors of the T5 paper [43] use. Additionally, I came up with the methods that utilize the *most predictable token*, including the *baseline evaluation scheme*. In Section 4.2 I will give an overview of all the experiments that I conducted. Section 4.3 will report the baseline results to which I will be comparing my experiments. After that, my main results will be presented in Section 4.4, Section 4.5 and Section 4.6.

4.1 Evaluation Schemes

While it would be desirable if T5 could solve tasks in real natural language (imagine having a conversation with T5 in the same way one would have a conversation with another human) this is far from being a reality. That is why it is necessary to define how to evaluate the capabilities of T5. While the authors of T5 [43] only used one simple evaluating method, I developed some more evaluation schemes that are applicable in more scenarios (although admittedly being less intuitive). I will introduce all those evaluation schemes in the following subsection.

4.1.1 Terminology

T5 was trained only in a text-to-text fashion, which means it has no specific output neurons that would correspond to certain labels. The only thing T5 can do is, given a sequence of token for its encoder and a sequence of token for its decoder, predict how likely it is for each token to follow next. More formally: Given the encoder sequence $a = (a_1, a_2, ..., a_n)$, the decoder sequence $b = (b_1, b_2, ..., b_m)$ and a vocabulary of tokens $T = \{t_1, t_2, ..., t_k\}$, T5 calculates the vector $p = (p_1, p_2, ..., p_k)$, where $p_i = P(t_i|a, b), i \in 1...n$, which is the by T5 assigned conditional probability that the token t_i follows next given the input a, b. Typically, the decoder sequence b will just be the pad token, which indicates that generation should start.

4.1.2 String Comparison

String comparison is the evaluation scheme that the authors of the original T5 paper [43] used. It is only applicable if T5 was trained to output a specific set of words given an instance of a dataset and a certain pattern. In the case of MNLI T5 was trained to either output the word "contradiction", "neutral" or "entailment". At test time, the output is then compared to the correct label. If T5 outputs something other than those three words, the authors count the prediction as wrong. However, they report that this was never the case for them and I observed the same.

4.1.3 Token Comparison

When extending the analysis to other patterns, I encountered the problem that some patterns do not restrict the output to a predefined set of words. For example, in SQuAD [44] T5 was trained to freely answer questions given a context with no output restrictions. In the SQuAD evaluation of the original T5 paper, the authors check if the output is an exact match with the label (string comparison). Now, if I want to use the SQuAD pattern (or any pattern where T5 does not output a limited amount of words) to predict the labels of MNLI, I run into the problem that I might underestimate the NLI capabilities of T5. For it could be the case that even though T5 outputs none of the strings "entailment", "neutral" or "contradiction" it might still assign higher probabilities to one string compared to another.

To capture those possible capabilities, I have to compare the probabilities of certain tokens and not just take the one with the highest probability as is done in the string comparison evaluation scheme. I then choose the token that has the highest probability compared to all other observed tokens as the prediction of T5.

To put it more formally in the context of MNLI: Let $Y = \{entailment, neutral, contradiction\}$ be the labels of MNLI. Let $v : Y \rightarrow T$ (only one token instead of many token compared to the definition in Section 2.6) be the verbalizer with $v(entailment) = t_e, v(neutral) = t_n, v(contradiction) = t_c$, with $e, n, c \in \{1, ..., k\}$. Given the encoder inputs a and the decoder inputs b I then define the prediction $pred$ of T5 as

$$pred = argmax(p_e, p_n, p_c)$$

with p_e, p_n, p_c as defined in Subsection 4.1.1

There are many types of verbalizers that one can use. Of course the most obvious one would be to choose t_e ="entailment", t_n ="neutral" and t_c ="contradiction". However, I could just as well choose something like t_e ="implication" which might achieve better accuracy. Or I could even choose a verbalizer that makes no sense to a human (like t_e ="banana") which might nonetheless give the best accuracy. But as this would be too many combinations to try out, I will stick to token that are sensible to humans for this evaluation scheme.

4.1.4 Most Predictable Token

As already mentioned in the last subsection, the fundamental problem with the *token comparison* evaluation scheme is that I do not know which verbalizer to choose. Even if I try many, there might still be some verbalizer that has a far greater accuracy. And trying all possible combinations would be too much to compute (32128^3 possibilities, since 32128 is the vocabulary length and there are three labels). To tackle this issue, I found a way to systematically choose the verbalizer (and corresponding tokens) that is most predictable for a label.

The key idea is to first determine the average logit score that T5 assigns to every token for the trainset. For example, T5 might assign the token corresponding to "computer" an average logit score of 3 (baseline). Secondly, I again compute the average logits score that T5 assigns for every token, but this time split for every label. For example, T5 might assign the token corresponding to computer a logits score of 8 for every entailment instance, a logit score of 2 for every neutral instance and a logit score of 1 for every contradiction instance. If I now calculate the differences between the total average and the label specific averages, I get the predictability of every token for the labels. In my example the predictability for entailment would be 8-3 = 5, for neutral 2-3 = -1 and for contradiction 1-3 = -2. This would mean that the token corresponding to "computer" would be highly predictable for entailment, but not for neutral and contradiction.

To put this procedure more formally: Let $train = (instance_1, instance_2, ..., instance_m)$ be the trainset of NLI instances of length m, with $instance_i = (a_i, b_i, label_i)$, where a_i is the encoder input (in our case the prompt+premise+hypothesis), b_i is the decoder input (in our case most often just the pad token) and $label_i$ is the corresponding label. For each token t_j in the vocabulary, let $L(instance_i, t_j) = l_{ij}$ be the logit score that T5 assigns to token j for the instance i. I compute the *baseline* $base_j$ for every token t_j as

$$base_j = \frac{1}{m}\sum_{i=1}^{m} l_{ij}$$

After that, I compute the label specific predictability $entailment_j$, $neutral_j$ and $contradiction_j$ for every token t_j as

$$entailment_j = \sum_{i=1}^{m} \begin{cases} base_j - l_{ij} & label_i = entailment \\ 0 & else \end{cases}$$

and $neutral_j$ and $contradiction_j$ are defined analogous.

If I now sort the label specific baselines in descending order, I get the most predictable token for every label. Those tokens can then be used in the *token comparison* scheme. As I need oracle access to the labels of the instances, I have to use the trainset for calculation.

4.1.5 Normalization

While choosing the prediction as the argmax of the probability of three tokens works well for many cases, it struggles if the tokens that are used have very different probabilities. This could be the case when comparing tokens that are more or less prevalent in the English language (i.e. "do" versus "nippy", see also [68]). Or it could be that T5 assigns certain tokens a higher probability for every MNLI instance as a function of the way MNLI instances are constructed.

To tackle this issue I normalize the probabilities p_e, p_n and p_c before calculating the argmax by dividing by the average probability that T5 assigns for those tokens for all test instances.

To put it more formally: Let p_e, p_n and p_c be the probability of the tokens t_e, t_n and t_c that the prediction should be mad with. Let $base_j$ be defined as in Subsection 4.1.4. Then the normalized probabilities pn_e, pn_n and pn_c are calculated as

$$pn_e = \frac{p_e}{base_e}, pn_n = \frac{p_n}{base_n}, pn_c = \frac{p_c}{base_c}$$

4.1.6 Baseline Evaluation Scheme

While using the most predictable tokens for evaluation is certainly an improvement over using random tokens, it is still limited to the information that is available in three of the output logits. It is likely that there is more information in other logits about what label an instance has. So it might be desirable to somehow use all outputs of T5 for classification.

In order to explore this hypothesis, I came up with what I call *baseline evaluation scheme*. The idea builds up on the most predictable token strategy (see Subsection 4.1.4). I again calculate the baseline $base_j$ and the label specific predictabilities $entailment_j$, $neutral_j$ and $contradiction_j$ for every token t_j using the trainset. For an instance i in the testset and every token j I then calculate

$$deviation_{ij} = l_{ij} - base_j$$

This vector captures how much each token deviates from the baseline.

If I now multiply the prediction vector by the deviation vector, the resulting vector will have large entries if the corresponding token is both highly predictive and deviates strongly in the positive direction for a given test instance. More formally: For the test instance i and the token j I calculate

$$entailment\_result_{ij} = deviation_{ij} * entailment_j$$

and analogously $neutral\_result_{ij}$ as well as $contradiction\_result_{ij}$. In practise, it proved empirically beneficial to first calculate the softmax over the $entailment$, $neutral$ and $contradiction$ vectors.

I then sum over the result vectors to achieve the final score:

$$entailment\_score_i = \sum_{j=1}^{k} entailment\_result_{ij}$$

and analogously for $neutral\_score_i$ and $contradiction\_score_i$.

As the last step, I count the prediction $pred_i$ of T5 for the test instance i as

$$pred_i = argmax(entailment\_score_i, neutral\_score_i, contradiction\_score_i)$$

With this method I can use every part of the output of T5 for prediction without having to choose a specific token. This method also does not need normalization as there is no direct comparison between two token.

4.2 Overview of the Experiments

As explained in Section 2.6, it is in the nature of language models that there are many PVPs that could be used to solve a particular task. In this bachelor thesis, I want to explore two major themes:

1. **Cross-PVP-Robustness**. If T5 is able to solve a task reasonably well with some PVP, will there be other PVPs (slightly different or completely new) with T5 can perform as well? Might there be even some PVP that will enable T5 to get a higher accuracy?
2. **Cross-Dataset-Robustness.** T5 was (most likely) only trained on MNLI and RTE. How can the skills acquired be transferred to other NLI datasets?

With those goals in mind, I will organize my experiments in the following way:

1. Experiment 1: Is T5 robust to slight variations in the MNLI pattern?
2. Experiment 2: Is it possible to achieve a good MNLI accuracy with patterns of other tasks that T5 was trained on?
3. Experiment 3: Do the NLI capabilities with the MNLI pattern translate to other NLI datasets?

4.3 Baseline Results

The version of T5 that I am using is a pretrained model from huggingface [64]. It was trained using unsupervised denoising training (Section 2.4) with the Colossal-Cleaned-Crawled-Corpus, also known as C4. C4 comprises an enormous amount of cleaned text from the internet. Furthermore, T5 was trained in a multi-task setting for a variety of tasks. I found no information on exactly which tasks it was trained on, but I assume that it was trained on the same datasets as mentioned in the T5 paper [43]. While there are different T5 model sizes available on huggingface, I use the T5-base model for all my experiments due to computational constraints.

For most of my experiments, I use the matched validation set of MNLI [63]. See Section 2.2 for an explanation of MNLI.

In the original T5 paper [43], the authors report an accuracy of 87.10% on the matched validation set of MNLI using T5-base. They use the evaluation scheme *string comparison* (p.Subsection 4.1.2), using a PVP with the Pattern

$P : X \rightarrow T^*, P(hypothesis, premise)$ = mnli hypothesis: *hypothesis* premise: *premise*

and the verbalizer

$$v : Y \rightarrow T$$

$$v(entailment, neutral, contradiction) = ("entailment", "neutral", ""contradiction)$$

I will refer to this PVP as the baseline-PVP from now on. With the same setup, I achieved an accuracy of 85.62%. I suspect that the slightly worse performance is due to different training procedures and parameters. Most notably, the authors in the original T5 paper [43] allow for single-task fine-tuning, while the model on huggingface was trained in a multi-task setting (see Section 2.4).

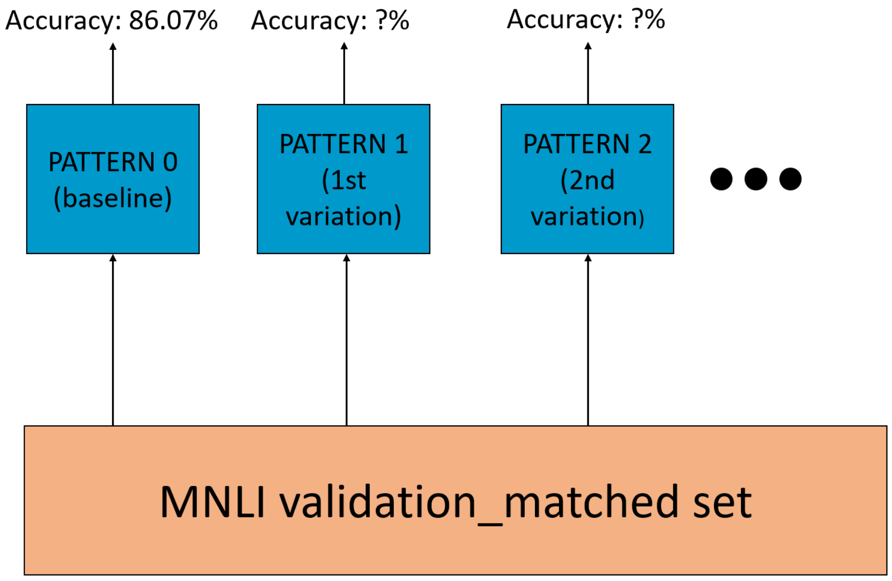

Accuracy: 86.07%
Accuracy: ?%
Accuracy: ?%
PATTERN 0 (baseline)
PATTERN 1 (1st variation)
PATTERN 2 (2nd variation)
MNLI validation_matched set

Figure 4.1: Overview of Experiment 1. I will slightly vary the baseline-pattern and observe how this impacts the accuracy of T5-base on the MNLI matched validation set. The verbalizer is always the baseline-verbalizer.

4.4 Experiment 1: Variation in the MNLI Pattern

I first explore how slight changes in the baseline-PVP affect the performance of T5 on MNLI. I employ the evaluation scheme of *string comparison* (see Subsection 4.1.2) and use the matched validation set of the MNLI dataset for evaluation. I will answer the following questions in this section:

1. Can beam search improve accuracy?
2. Is T5 robust to omitting parts of the baseline-pattern?
3. Is T5 robust to slight variations in the baseline-pattern?
4. Is T5 robust to adding tokens to the baseline-pattern?
5. Is T5 robust to changing the order of the input?

In this section, I will only change the pattern of the baseline-PVP (Section 2.6). As the pattern for MNLI can be described as $P : (hypothesis, premise) \to A\ hypothesis\ B\ premise\ C$ where $A, B, C \in T^*$, I will only say what A, B and C are in the following subsections. If I do not specify one of those variables, it means that they are the same as the baseline-pattern (Section 4.3). Figure 4.1 shows an illustrated overview of this section.

4.4.1 Can Beam Search Improve Accuracy?

Although still being a greedy decoding algorithm, beam search (see Section 2.5) keeps several hypotheses at a time and can therefore find sequences that are more likely compared to greedy decoding.

Decoding Method	Accuracy
Greedy (baseline)	85.62%
Beam search, num_beams=2	86.07%
Beam search, num_beams=3	86.02%
Beam search, num_beams=6	86.02%

Table 4.1: Accuracy of T5-base on the MNLI matched validation set using the evaluation scheme of string comparison and different decoding schemes.

Whereas the label neutral and the label contradiction get tokenized to one token each, the label entailment gets tokenized to four tokens. Therefore, it is possible that using beam search compared to greedy decoding can lead to a better accuracy. Indeed, as shown in Table 4.1, beam search can improve accuracy by 0.45%.

Utilizing those insights, I will be using beam search with two beams for all following experiments.

4.4.2 Is T5 Robust to Omitting Parts of the Prompt?

One of the most straight-forward ways to change the baseline-pattern is to omit part of it. In my experiment I omit either the strings "mnli", "hypothesis:" or "premise:" or a combination.

The results of this experiment are shown in Table 4.2. I observed that omitting each part of the pattern results in a drop in accuracy. How much the accuracy drops depends on which part gets omitted. While omitting either "hypothesis:" or "premise:" and still achieves an accuracy far above the most frequent class baseline, omitting both leads to an accuracy of 0%. In fact, when omitting both, T5 does not predict any of the labels anymore and instead defaults to repeating some part of the input. This means that T5 does not recognize that it has to solve a MNLI task in that case.

For example, given the hypothesis: "Everyone really likes the newest benefits" and the premise = "The new rights are nice enough" (which are part of the MNLI matched validation set), T5 outputs "<extra_id_0> Everyone really likes the newest benefits The new rights are nice enough The new rights are". The token "<extra_id_0>" is used for the unsupervised pretraining task, see Section 2.4. Furthermore, T5 does not detect the task with the pattern only being the string "mnli", as the accuracy is 0% in that case as well.

4.4.3 Is T5 Robust to Slight Variations of the Prompt?

In this section, I will slightly vary some parts of the baseline-pattern. Slightly in the sense that only a few characters are modified, and it would still be understandable by a human. For example, this includes changing "mnli" to "nli", leaving away the colons or using "hypo" and "pre" instead of "hypothesis" and "premise". Note that while those changes seem small for the human reader, they can result in completely different tokens for T5.

The results are shown in Table 4.3. I observe that omitting the colons has virtually no effect, while changing "hypothesis" to "hypo" results in 0% accuracy. The reason for this

A	B	Accuracy
mnli hypothesis:	premise:	86.07%
mnli	premise:	84.30%
mnli hypothesis:	ϵ	80.45%
mnli	ϵ	0%
hypothesis:	premise:	85.94%
ϵ	premise:	84.97%
hypothesis:	ϵ	73.16%
ϵ	ϵ	0%

Table 4.2: Accuracy of T5-base on the MNLI matched validation set using the evaluation scheme of *string comparison* (see Subsection 4.1.2) when omitting part of the prompt. ϵ is the empty string.

A	B	Accuracy
mnli hypothesis:	premise:	86.07%
nli hypothesis:	premise:	86.11%
mnli hypothesis	premise:	85.95%
mnli hypothesis:	premise	86.00%
mnli hypothesis	premise	85.85
mnli hypo	premise	0.00%
mnli hypothesis:	pre	80.45%
mnli hypothesis	pre	68.93%

Table 4.3: Accuracy of T5-base on the MNLI matched validation dataset using the evaluation scheme of *string comparison* (see Subsection 4.1.2) when slightly varying the prompt.

certainly has to do with the fact that "hypothesis" gets encoded as one token, which means that any prefix of "hypothesis" will be a completely different token. However, the same is true when I change "premise" to "pre" and here I do not observe a complete failure mode.

Compared to Table 4.2 where I completely omit the "hypothesis:" and still achieving good results, changing it to "hypo" leads to 0% accuracy, demonstrating severe brittleness to one token. When I change "mnli" to "nli" I get an accuracy that exceeds the baseline pattern by 0.04%. While it possible to interpret this as a sign that there might be patterns that can achieve a better MNLI accuracy than the baseline-pattern, in this case the gain is far from being significant.

To conclude: T5 generally has a higher accuracy the more similar the pattern is to the baseline-pattern. While some changes affect the accuracy next to none, others degrade it significantly.

A	B	C	Accuracy
mnli hypothesis:	premise:	""	86.07%
mnli *consider_pattern* hypothesis:	premise:	ϵ	86.00%
mnli_pattern hypothesis:	premise:	ϵ	69.22%
mnli hypothesis:	*consider_pattern* premise:	ϵ	84.89
mnli *greek_pattern* hypothesis:	premise:	ϵ	56.94%
mnli hypothesis:	*greek_pattern* premise:	ϵ	56.51%
hypothesis:	*greek_pattern* premise:	*question_pattern*	85.65%
mnli conclusion, hypothesis:	assumption, premise:	ϵ	73.27%

Table 4.4: Accuracy of T5-base on the MNLI matched validation dataset using the evaluation scheme of *string comparison* (see Subsection 4.1.2) when adding some tokens. In order to improve clarity, abbreviations are used. *consider_pattern* = "Consider the following", *greek_pattern* = "A greek soldier drank some lemon juice.", *mnli_pattern* = "multi natural language inference" and *question_pattern* = "Should the label be entailment, neutral or contradiction?". ϵ is the empty string.

4.4.4 Is T5 Robust to Adding Tokens to the Prompt?

In this section, I investigate whether the NLI capabilities of T5 are robust to adding a few tokens to the pattern. I insert either random text ("A greek soldier drank some lemon juice") or text that would be sensible to a human ("multi natural language inference" instead of mnli and also "Should the label be entailment, neutral or contradiction?").

Results in Table 4.4 are mixed. Some text seems to matter little ("Consider the following") while other text degenerates accuracy substantially ("A greek soldier drank some lemon juice."). In fact, for the latter, I observed that T5 just stops outputting "entailment" all together.

4.4.5 Is T5 Robust to Changing the Order?

In this subsection, I change the order of the hypothesis and the premise. This means that the pattern is now $P : (hypothesis, premise) \rightarrow A\ premise\ B\ hypothesis\ C$. Note that this does not change the label of an instance and is logically equivalent to the pattern in normal order. For evaluation, I use A = "mnli premise: " and B=" hypothesis: " for comparison with the baseline-PVP.

With this setup, T5 only achieves an accuracy of 59.02%. This demonstrates that the NLI capabilities of T5 are very brittle to changes in the order. It also supports previous research [31] which observed brittleness to changes in the order of few-shot patterns in GPT-3.

4.4.6 Conclusion of Experiment 1

In conclusion, the results of the robustness to slight variations in the baseline-pattern is mixed. I classify the variations in three different categories:

1. Variations that achieve roughly the same accuracy as the baseline-pattern.
2. Variations that achieve worse accuracy than the baseline-pattern, but still far above the most frequent class baseline (over 55%).
3. Variations that achieve 0% accuracy because T5 does not recognize that it is a MNLI task anymore.

Since most variations that I tried fall into the first or second category, I conclude that T5 is at least somewhat robust to variations in the baseline-pattern.

4.5 Experiment 2: Cross-Task-Robustness

T5 was trained in a multi-task fashion (see Section 2.4). That is why there are a lot of specific patterns that correspond to a task. Table 2.2 shows a full list of tasks and patterns that T5 was trained on and that I am using. This provides the opportunity to cast the NLI objective to another task.

For example, one task that T5 was trained on is question answering. In this format, T5 gets a question and a context (that has the answer to the question) and T5 has to generate the correct answer (see Table 2.2).

One way of casting a NLI instance to the question answering format would be to state the question as "Does the premise entail, contradict or is neutral to the hypothesis?" and as context the concatenation of the premise and the hypothesis. The whole input for T5 would thus be

> "question: Does the premise entail, contradict or is neutral to the hypothesis?
> context: hypothesis: *hypothesis* premise: *premise*"

If T5 were to understand the question correctly and "apply its NLI skills", I would expect the same accuracy as with the MNLI pattern.

I call this type of casting logically equivalent, in the sense that the label for a NLI instance with this casting should be X if and only if the label of that instance with the baseline-pattern is X.

There are other tasks where it is harder to find a mapping to NLI. For example, sentiment analysis might correlate with NLI (i.e. hypothesis-premise pairs that have a positive sentiment might be more likely to have the label entailment) but is definitely not logically equivalent.

The third option is when a part of the MNLI labels can be represented logically equivalent, but not all of them. For example, this is the case with RTE which only has the labels entailment and not_entailment. If T5 predicts entailment with the RTE pattern, the MNLI label will be entailment as well. However, if T5 predicts not_entailment, the MNLI label could be either contradiction or neutral. With this in mind, I will explore three following questions:

- How robust are the NLI capabilities of T5 when using the pattern of another task, which is logically equivalent to the NLI task?

- How robust are the NLI capabilities of T5 when using the pattern of another task, which can partially logically equivalent represent the NLI labels?
- For tasks which there are no logically equivalent ways to cast a NLI instance to, are there combinations where T5 still has an accuracy over the most frequent class baseline?

The first and the second question explore the robustness to (partially) logical equivalent task perturbations. Here, T5 would be robust if it achieves similar accuracy as with the baseline-pattern. For the third question, the opposite would be true. Accuracy above a most frequent class baseline would indicate the existence of statistical cues in the dataset (for example, that T5 uses sentiment in its NLI prediction).

In the following subsections, I will explore different tasks that T5 was trained on, which all give insight to one of those questions. For tasks which T5 is trained to output a limited set of tokens, I use the *string comparison* evaluation method (see Subsection 4.1.2). The verbalizer has to be adjusted here to match the task specific labels. For generative tasks, where T5 can output any token, I use the *token comparison* evaluation (see Subsection 4.1.3) or *baseline evaluation scheme* (see Subsection 4.1.6).

To illustrate the pattern that each task will use, I will use "All cars are black." as premise and "My car is red." as hypothesis. While this premise and hypothesis pair is not part of MNLI, it will serve as a good demonstration to make it easier to replicate my results.

When the labels for the task is not the same as the MNLI labels, a new verbalizer has to be found. In the following subsections I will use tables to specify the new verbalizer: The top row will always show the three labels (entailment, neutral and contradiction) while the other rows will show which tokens the label will get mapped to. This means that each row is a new verbalizer with the corresponding accuracy. If two labels get mapped to the same tokens (in case that the specific task does not have three labels like MNLI) I count a classification as correct, if either of the two labels apply when the model predicts those tokens. For those cases, the most frequent class accuracy is 68.25%.

See Figure 4.2 for an illustrated overview of this experiment.

4.5.1 CommitmentBank Dataset (CB)

CB [32] is a NLI dataset where the goal is to determine if a premise entails, contradicts or is neutral to a hypothesis. However, compared to MNLI the premises were chosen deliberately such that they are surrounded by an entailment canceling operator like negation but without changing the associated label.

The authors of T5 chose to encode this task the same in the text-to-text framework as MNLI with the only difference being that the changed "mnli" to "cb". See Table 4.5 for an example.

As CB has the same labels as MNLI, I can use the same verbalizer as I do for MNLI. This also means that I can cast a MNLI instance logically equivalent to the CB pattern and can investigate the first question (see Section 4.5).

Results in Table 4.6 show that the accuracy does not change substantially. I conclude that the NLI capabilities of T5 are robust for this mapping. However, note that this can

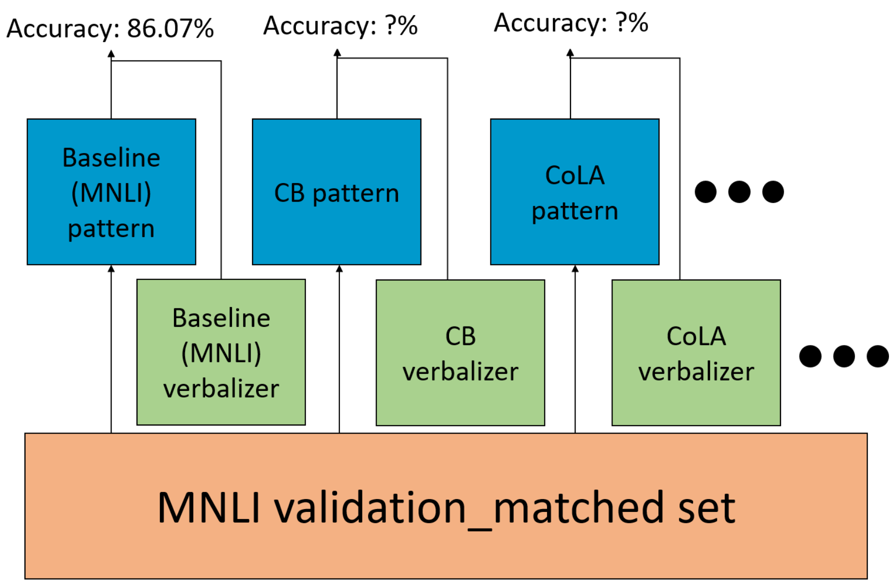


Figure 4.2: Overview of Experiment 2. I will try the PVP of different task with the MNLI matched validation set. The goal is either to see if NLI capabilities transfer across task or to find indications of the use of statistical cues.

Task	Toy Example With Task Specific Pattern
MNLI	mnli hypothesis: My car is red. premise: All cars are black.
CB	cb hypothesis: My car is red. premise: All cars are black.

Table 4.5: Illustration of the pattern that I use for CB with a toy example. "All cars are black" is the premise and "My car is red." is the hypothesis.

A	B	Accuracy
mnli hypothesis:	premise:	86.07%
cb hypothesis:	premise:	85.80%

Table 4.6: Accuracy of T5-base on the MNLI matched validation dataset using the evaluation scheme of *string comparison* (see Subsection 4.1.2) and the prompt for CB.

Task	Toy Example With Task Specific Pattern
MNLI	mnli hypothesis: My car is red. premise: All cars are black.
CoLA	cola sentence: My car is red. All cars are black.

Table 4.7: Illustration of the pattern that I use for CoLA with a toy example. "All cars are black" is the premise and "My car is red." is the hypothesis.

Entailment	Neutral	Contradiction	Accuracy
unacceptable	acceptable	acceptable	63.39%
acceptable	unacceptable	acceptable	64.62%
acceptable	acceptable	unacceptable	64.27%

Table 4.8: Accuracy of T5-base on the MNLI matched validation dataset using the CoLA pattern.

also be seen as a slight variation of the MNLI pattern (Subsection 4.4.3) which means that results might have been different if the authors used completely different patterns for MNLI and CB.

4.5.2 The Corpus Of Linguistic Acceptability (CoLA)

The CoLA [60] task involves determining whether or not a sentence is grammatically acceptable. The pattern for the CoLA task is shown with a toy example in Table 4.7.

As the T5 was trained to output either "acceptable" or "unacceptable" for CoLA, I have to adjust the verbalizer. As I did not see a way to cast a MNLI instance logically equivalent to CoLA, I will use it to investigate the third question (Section 4.5) and look for statistical cues. The verbalizers and corresponding accuracies I tried are in shown in Table 4.8.

No verbalizer achieved an accuracy which exceeded the most frequent class baseline of 68.25% significantly. I conclude that T5 does not use any statistical cues for MNLI that would overlap with any patterns that it uses to solve CoLA.

4.5.3 Recognizing Textual Entailment (RTE)

RTE [10] is a NLI dataset as well. However, instead of having to predict the three labels "entailment", "neutral" and "contradiction" like in MNLI the goal in RTE is only to predict "entailment" vs. "not_entailment". This means that it is only possible to represent parts of the MNLI labels. I will use RTE to investigate question 2 (Section 4.5).

Table 4.9 demonstrated the pattern that is used with RTE.

In Table 4.10 I show the different verbalizers that I tried. If I use the most obvious choice and map the label entailment to "entailment" and the labels neutral and contradiction to "not_entailment" I achieve an accuracy of 84.85%. This is significantly higher than the most frequent class baseline of 68.25% for two labels.

Task	Toy Example With Task Specific Pattern
MNLI	mnli hypothesis: My car is red. premise: All cars are black.
RTE	rte sentence1: My car is red. sentence2: All cars are black.

Table 4.9: Illustration of the pattern that I use for RTE with a toy example. "All cars are black" is the premise and "My car is red." is the hypothesis.

Entailment	Neutral	Contradiction	Accuracy
entailment	not_entailment	not_entailment	84.85%
entailment	not_entailment	entailment	59.47%
entailment	entailment	not_entailment	61.04%

Table 4.10: Accuracy of T5 on the MNLI matched validation dataset using the RTE prompt.

I conclude that the way that T5 solves RTE is also somewhat useful to solve MNLI. In this sense, the NLI capabilities are at least to some degree robust. However, one can observe that the accuracy for detecting entailment versus contradiction and neutral is still far beyond that baseline.

4.5.4 Microsoft Research Paraphrase Corpus (MRPC)

In MRPC [12] the goal is to determine if two sentences are equivalent or not. The pattern that is used in MRPC is shown in Table 4.12. Note that it is very similar to the RTE pattern.

As there are once again only two labels, I can only represent part of the MNLI label and will therefore investigate question 2 Section 4.5. The verbalizers and corresponding accuracies that I tried are shown in Table 4.13. Logically speaking, the MRPC label not_equivalent should be verbalized to "contradiction" and "neutral". While two statements where one entails the other do not necessarily have to be equivalent, there still might be some significant overlap.

The results in Table 4.13 show all the verbalizers I tried and the corresponding accuracies. The party logically equivalent mapping does indeed show the best results, far above the most frequent class baseline.

Since I do not know which entailment instances in the MNLI testset are also equivalent, it is hard to draw any conclusion. However, it is possible to check the internal consistency of T5 by measuring how many instances T5 labels as entailment when swapping the premise and the hypothesis.

Figure 4.3 shows an illustration on how to achieve this. First, I determine all the instances of the T5 matched validation set where T5 thinks that the premise entails the hypothesis (called entailment in the diagram) using the baseline pattern. Then I use the pattern $P : X \rightarrow T^*, P(hypothesis, premise)$ = "mnli hypothesis: " *premise* " premise: " *hypothesis* to determine all the instances where T5 thinks that the hypothesis entails the premise (called reversed entailment in the diagram). The instances in the intersection of those two sets are the instances that T5 thinks are equivalent when using the mnli pattern.

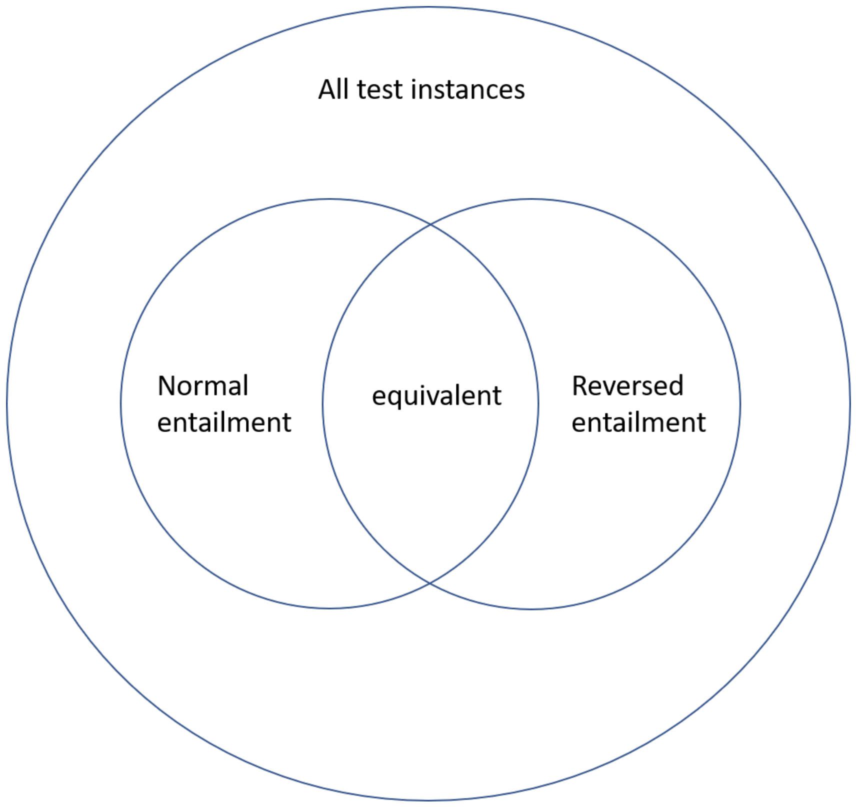


Figure 4.3: An illustration on how to determine which mnli matched validation instances T5 thinks are equivalent.

Method To Determine Equivalence	Fraction Of Equivalence
MNLI (normal entailment / reversed entailment)	11.04%
MRPC	21.67%

Table 4.11: Which fraction of MNLI matched validation instances T5 thinks are equivalent using either the MRPC pattern or the intersection of the normal and the reversed MNLI pattern with entailment.

Task	Toy Example With Task Specific Pattern
MNLI	mnli hypothesis: My car is red. premise: All cars are black.
MRPC	mrpc sentence1: My car is red. sentence2: All cars are black.

Table 4.12: Illustration of the pattern that I use for MRPC with a toy example. "All cars are black" is the premise and "My car is red." is the hypothesis.

The result of those calculations is that T5 thinks that 11.03% of all matched validation instances are equivalent. That is less than a third of how much T5 thinks is just entailment (34.41%).

I can now compare those results with the MRPC results. Table 4.11 shows which fraction of the MNLI matched validation set T5 thinks are equivalent using either the MRPC prompt or combining the normal and reversed prompt of MNLI as described above. With MRPC T5 thinks there are almost double the equivalent instances than with the MNLI combination. This shows a significant mismatch and shows that T5 is inconstant and not very robust in its equivalence predictions.

4.5.5 Stanford Question Answering Dataset (QNLI)

The objective of QNLI [44] is to determine whether or not a given context contains the answer to a given question. The pattern that I use for QNLI is shown in Table 4.14. As there is no obvious way to cast a premise and a hypothesis to question and context, I simply use the hypothesis as the question even though there is no question involved.

I could not come up with a way to cast the labels logically equivalent. That is why I use this task to investigate question 3 (see Section 4.5). If my setup achieves an accuracy of

Entailment	Neutral	Contradiction	Accuracy
equivalent	not_equivalent	not_equivalent	76.46%
equivalent	not_equivalent	equivalent	52.07%
equivalent	equivalent	not_equivalent	61.40%

Table 4.13: Accuracy of T5 on the MNLI matched validation dataset using the prompt of mrpc and different label mappings.

Task	Toy Example With Task Specific Pattern
MNLI	mnli hypothesis: My car is red. premise: All cars are black.
QNLI	qnli question: My car is red. sentence: All cars are black.

Table 4.14: Illustration of the pattern that I use for QNLI with a toy example. "All cars are black" is the premise and "My car is red." is the hypothesis.

Entailment	Neutral	Contradiction	Accuracy
entailment	not_entailment	not_entailment	79.34%
entailment	not_entailment	entailment	61.25%
entailment	entailment	not_entailment	59.42%

Table 4.15: Accuracy of T5 on the MNLI matched validation dataset using the evaluation scheme of *string comparison* (see Subsection 4.1.2) with the prompt of QNLI.

more than the most frequent class baseline, this is an indication that T5 is using spurious statistical cues.

Table 4.15 shows that it is indeed possible to achieve an accuracy far above the 68.25% most frequent class baseline. This could be a sign that T5 exploits some kind of pattern in the two sentences and does not actually answer the question when solving QNLI. However, if this task is given to humans and their responses are limited to the labels, it could very well be the case that they provide similar answers as the task is not correctly specified.

4.5.6 Quora Question Pairs (QQP)

QQP [15] is a dataset consisting of pairs of questions. The goal is to determine whether or not the questions are duplicates in the sense that they are asking for the same thing.

Table 4.17 illustrates the pattern that is used in QQP. The task is very similar to MRPC although now for questions instead of statements. Of course, neither the premise nor the hypothesis of MNLI are questions. Still, if T5 learned something more general about duplicates that goes beyond questions, it might be able to differentiate the equivalent MNLI pairs from the non-equivalent ones.

The results in Table 4.18 show that this setup can achieve accuracy significantly above the most frequent class baseline. However, as with MRPC it has to be differentiated if T5 finds instances that are actually equivalent or if it relies on some other pattern. As I once again do not have the possibility to find out which instances in the MNLI matched validation set really are equivalent, I can only test for consistency (see Subsection 4.5.4).

Table 4.16 shows the result using the same setup as in Table 4.11, only this time with QQP instead of MRPC. There again is a significant mismatch between which instances T5 thinks are equivalent using the different methods. I conclude that T5 is also not very robust in this regard.

Method To Determine Equivalence	Fraction Of Equivalence
MNLI (normal entailment / reversed entailment)	11.04%
QQP	26.78%

Table 4.16: Which fraction of MNLI matched validation instances T5 thinks are equivalent using either the QQP pattern or the intersection of the normal and the reversed MNLI pattern with entailment.

Task	Toy Example With Task Specific Pattern
MNLI	mnli hypothesis: My car is red. premise: All cars are black.
QNLI	qqp question1: My car is red. question2: All cars are black.

Table 4.17: Illustration of the pattern that I use for QNLI with a toy example. "All cars are black" is the premise and "My car is red." is the hypothesis.

Entailment	Neutral	Contradiction	Accuracy
duplicate	not_duplicate	not_duplicate	76.91%
not_duplicate	duplicate	not_duplicate	51.67%
duplicate	duplicate	not_duplicate	55.20%

Table 4.18: Accuracy of T5 on the MNLI matched validation dataset using the evaluation scheme of string comparison and the pattern on QQP.

Task	Toy Example With Task Specific Pattern
MNLI	mnli hypothesis: My car is red. premise: All cars are black.
SST2	sst2 sentence: My car is red. All cars are black.

Table 4.19: Illustration of the pattern that I use for SST2 with a toy example. "All cars are black" is the premise and "My car is red." is the hypothesis.

Entailment	Neutral	Contradiction	Accuracy
positive	negative	negative	59.81%
negative	positive	negative	53.74%
positive	positive	negative	60.87%

Table 4.20: Accuracy of T5-base on the MNLI matched validation dataset using the evaluation scheme of string comparison and the prompt of SST2.

4.5.7 Stanford Sentiment Treebank (SST2)

SST2 [53] is a sentiment classification dataset, where the goal is to classify a sentence to either "positive" or "negative".

Table 4.19 shows the pattern that is used for SST2 and how I combined it with the MNLI instances.

I explored all label mappings as different verbalizers. As the mappings are not logically equivalent, my goal is to explore question 3 (see Section 4.5) and look for statistical cues. For example, it might be conceivable that T5 would give MNLI instances with the label contradiction lower sentiment as instances with the label entailment.

The results are shown in Table 4.20. No mapping achieves an accuracy of more than the most frequent class baseline. I conclude that T5 does not use statistical cues that relate to that particular sentiment analysis in order to solve MNLI.

4.5.8 Semantic Textual Similarity Benchmark (STSB)

In STSB [8] the task is to predict the semantic textual similarity of two sentences as a score between 0 and 5.

The pattern that is used for STSB is shown in Table 4.21. Given this pattern, T5 was trained to output a number between 0 and 5 in steps of 0.2. While I could not directly verbalize labels to a continuous score, I could still look for patterns that T5 applies. To that end, I first calculated the sum of the predictions for every label.

Results in Table 4.22 show that T5 predicts much higher values for MNLI instances with the label entailment. Using this knowledge, I can make predictions for MNLI by introducing thresholds for different labels.

Table 4.23 shows the accuracy for different thresholds when trying to differentiate between entailment and neutral/contradiction. Using a threshold of 3.6, it is possible to predict "entailment" correctly with an accuracy of 78%, which is far above the most frequent class baseline of 68.25%.

Task	Toy Example With Task Specific Pattern
MNLI	mnli hypothesis: My car is red. premise: All cars are black.
STSB	stsb sentence1: My car is red. sentence2: All cars are black.

Table 4.21: Illustration of the pattern that I use for SST2 with a toy example. "All cars are black" is the premise and "My car is red." is the hypothesis.

Label	Sum	Average By Label
entailment	12192	3.50
neutral	5778	1.85
contradiction	7060	2.20

Table 4.22: Averaged and summed prediction score of T5 base using the prompt of STSB for every instance and corresponding label of MNLI.

Threshold	Accuracy
2.7	73.00%
3	74.43%
3.3	78.28%
3.5	78.35%
3.6	78.35%
3.8	74.89%
3.9	73.52%
4.0	73.52%

Table 4.23: Accuracy of T5 on the MNLI matched validation dataset using the evaluation scheme of string comparison, the prompt of STSB and different thresholds for differentiating between entailment and contradiction/neutral.

Threshold Entailment	Threshold Contradiction	Accuracy
3.6	1.85	42.49%
3.6	1.6	48.01%
3.6	1.5	48.01%
3.6	1.4	47.68%

Table 4.24: Accuracy of T5 on the MNLI matched validation dataset using the evaluation scheme of string comparison, the prompt of STSB and different thresholds for differentiating between entailment and contradiction as well as between contradiction and neutral.

Since STSB does not only have two labels, it is possible to use two thresholds to predict the whole MNLI label set. In Table 4.24 I explored different thresholds. If the prediction is above the entailment threshold, I predict entailment. If the prediction is between the entailment threshold and the contradiction threshold, I predict contradiction. If the prediction is below the contradiction threshold, I predict neutral. Using this scheme, I get an accuracy of 48.01%, which is far above the most frequent class baseline of 36.5%.

This indicates that there are statistical cues in the dataset, in the sense that T5 can partly rely on the same methods that it uses to solve STSB. This could be actual semantic similarity, or it could be some other statistical cue, since there is also no way to guarantee that T5 solves STSB like humans would do. Furthermore, this does not mean that T5 actually uses those methods with the MNLI pattern, but it is a strong indication for it.

4.5.9 Stanford Question Answering Dataset (SQuAD)

In SQuAD [44] the goal is to answer a question given a context. The answer is not limited to a predefined set of labels, but instead can be any word. This effectively makes SQuAD a generation and not a classification task. For me, this meant that I could not simply use the string comparison method as before. Instead, I will use token prediction (see Subsection 4.1.3) and baseline prediction (see Subsection 4.1.6) for the evaluation. Since the question can be construed arbitrarily, I can logically equivalent cast the MNLI objective to SQuAD and explore question 1 (see Section 4.5).

Table 4.25 shows the general pattern that I will use in this subsection. The italic question will be substituted for various actual questions.

There are a lot of different ways to cast the MNLI objective to a question and a context. For a first step, I explore a few different patterns and verbalizers that would be sensible for a human using the token prediction scheme. Table 4.26 provides a list of all the questions I came up with and the abbreviations that I gave them to make the following tables clearer. Table 4.27 shows a list and abbreviations of the corresponding contexts. Note that if I still use the strings "hypothesis" and "premise", the pattern is actually just a slight variation of the baseline-pattern, like seen in Section 4.3.

Table 4.28 shows various combinations of patterns and verbalizers (the question and context get arranged in the same fashion as shown in Table 4.25). It provides the corresponding accuracies and also the normalized accuracies (Subsection 4.1.5). Consistent

Task	Toy Example With Task Specific Pattern
MNLI	mnli hypothesis: My car is red. premise: All cars are black.
SQuAD	question: *question* context: My car is red. All cars are black.

Table 4.25: Illustration of the pattern that I use for SQuAD with a toy example. "All cars are black" is the premise and "My car is red." is the hypothesis. The string *question* is a placeholder for an actual question.

Question	Abbreviation
Does the premise entail, contradict or is neutral to the hypothesis?	*question1*
Does the sentence1 entail, contradict or is neutral to the sentence2?	*question2*

Table 4.26: List of all the questions I use with SQuAD and the corresponding abbreviations that I define.

with the result of Subsection 4.4.4, adding a lot of tokens to the baseline-PVP makes the accuracy drop substantially, but is still far above the most frequent class baseline. When changing the question and context in a way such that no token of the original prompt is available to the model anymore, the accuracy drops to random chance. However, using a different verbalizer (last three rows in the table) it is possible to achieve an accuracy over the most frequent class baseline. But that is only possible when using a verbalizer that is the exact opposite of what a human would find sensible (last row in the table). This hints at the fact that T5 may use spurious statistical cues in this context.

Next, I will follow a more systematic approach. I will fix the pattern and look for verbalizers (and corresponding tokens) which are highly predictive of a label (see Subsection 4.1.4). Since the actual labels of the instances are needed to evaluate how predictable they are, I use the trainset to find the highly predictable verbalizers. Using this technique, I was able to find tokens that resulted in higher accuracy on the MNLI matched validation set, see Table 4.29. For *question1* and *context1* I could improve accuracy by about 5% compared to the verbalizers that are sensible for humans. For *question2* and *context2* I could improve accuracy by about 9%.

It is notable that the most predictive tokens are completely senseless to a human, some not even being English words. Since it is possible to improve accuracy significantly with

Context	Abbreviation
hypothesis:[hypothesis] premise: [premise]	*context1*
sentence2:[hypothesis] sentence1: [premise]	*context2*

Table 4.27: List of all the contexts I use with SQuAD and the corresponding abbreviations that I define. [hypothesis] and [premise] stands for the hypothesis and premise of an actual MNLI instance.

Question	Context	Entailment	Neutral	Contradiction	Accuracy	Normalized
question1	*context1*	entailment	neutral	contradiction	52.96%	57.81%
question2	*context2*	entailment	neutral	contradiction	35.46%	38.72%
question1	*context1*	implication	unrelated	exclusion	35.45%	30.02%
question2	*context2*	implication	unrelated	exclusion	35.45%	25.25%
question2	*context2*	exclusion	implication	unrelated	31.82%	43.47%

Table 4.28: Accuracy of T5 on the MNLI matched validation dataset using the evaluation scheme of string comparison when using the prompt of SQuAD and different labels.

Question	Context	Entailment	Neutral	Contradiction	Accuracy	Normalized
question1	*context1*	vin	helpful	weder	32.02%	62.71%
question1	*context1*	tôt	tinde	neither	32.47%	57.80%
question2	*context2*	égi	formează	0.0	32.27%	52.63%
question2	*context2*	repayment	ally	0	33.11%	50.28%

Table 4.29: Accuracy of T5-base on the MNLI matched validation dataset using the token evaluation scheme with highly predictable tokens. I use the prompt of SQuAD.

those tokens, this is a strong indication that T5 uses some kind of statistical cues which are incomprehensible to a human reader.

Even though I now used the most predictable verbalizers for prediction, T5 might still perform better if it were to have access to the prediction qualities of its whole output. Therefore, I now use the *baseline evaluation scheme* (see Subsection 4.1.6) which means that the whole output of T5 gets used for prediction.

The results are shown in Table 4.30. Compared to using the most predictable verbalizer for classification, I now get an accuracy increase of about 5% for *question1* and *context1* and an accuracy increase of about 1.5% for *question2* as well as *context2*.

All in all, I conclude that it is possible to use the SQuAD pattern to get a decent accuracy on MNLI. In this sense, the NLI capabilities of T5 are somewhat robust. However, this is only possible when using forms of evaluation that are not sensible to a human reader. So the NLI capabilities are not robust in the sense that T5 would "think" in the same sense that humans would do.

Question	Context	Accuracy
question1	*context1*	67.38%
question2	*context2*	54.09%

Table 4.30: Accuracy of T5-base on the MNLI matched validation dataset using the *baseline evaluation scheme*. I use the prompt of SQuAD.

4.5.10 Conclusion of Experiment 2

To summarize the results of experiment 2, I want to come back to the initial questions that I asked (see Section 4.5).

The first question was how robust the NLI capabilities of T5 are when logically equivalent casting to a pattern other than the MNLI pattern. This was the case for CB and SQuAD. The PVP of CB worked as well as the baseline MNLI pattern. So in this case, the NLI capabilities are robust. However, this is not very surprising as the CB PVP is almost the same as the MNLI PVP.

How well the PVP of SQuAD works depends on how the question and the context is constructed. If there still is a part of the MNLI prompt left (i.e. "hypothesis: " and "premise: ") the accuracy is far above the most frequent class baseline. Those results are comparable to the slight variation of adding tokens to the baseline-pattern (see Subsection 4.4.4), as the pattern is constructed similarly and also the accuracy that is achieved is similar. When there is no part of the MNLI pattern left, accuracy drops significantly, but is still above the most frequent class baseline. So I conclude that the NLI capabilities are also at least somewhat robust in that regard.

The second question asked about the robustness of the NLI capabilities of T5 when it is only possible to logically equivalent cast some of the MNLI labels. This was the case for RTE, MRPC and QQP. With all of those patterns, I was able to achieve an accuracy that exceeded the most frequent class baseline. However, in the case of MRPC and QQP I also found that T5 is not consistent in which instances of the MNLI matched validation set it labels as equivalent.

For the third question, I looked at tasks where there is no (partially) logically equivalent mapping available. This was the case for CoLA, QNLI, SST2 and STSB. For CoLA and SST2, I did not find a way to achieve an accuracy above the most frequent class baseline, indicating that T5 does not rely on the same statistical cues for those datasets and MNLI. For QNLI and for STSB I found ways to achieve an accuracy above the most frequent class baseline. This shows that T5 partially uses similar patterns to solve QNLI and STSB as it does to solve MNLI.

4.6 Experiment 3: Cross-Dataset-Robustness

In the last sections, I have carried out all evaluations with the MNLI matched validation set. However, as described in Section 2.2 this is not the only NLI dataset available. This section is about how robust the NLI capabilities that T5 learned with MNLI are with those other datasets.

One issue I ran into is that I found no documentation on which tasks the version of T5 that I am using (from huggingface [64]) was trained on. I assume that the version was trained on the training set of all the task that are mentioned in the T5 paper [43].

See also Figure 4.4 for an illustrated overview.

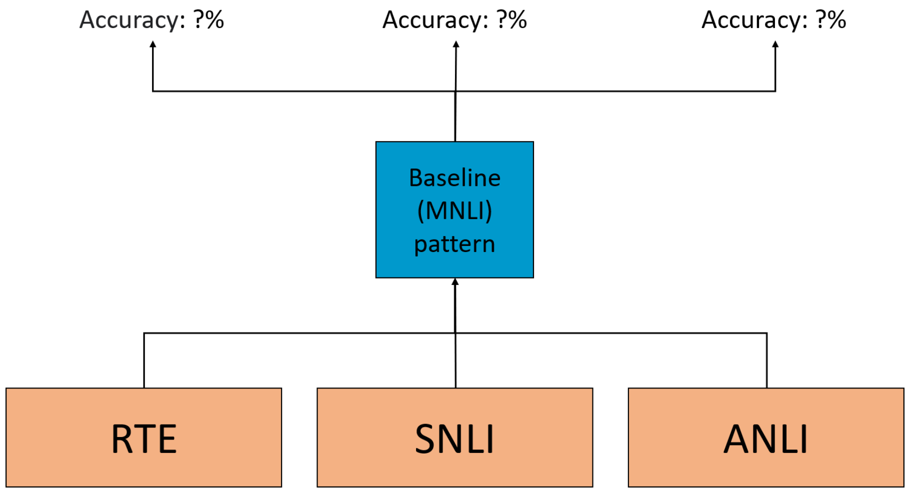


Figure 4.4: Overview of Experiment 3. I will try different NLI datasets with the baseline PVP and as such determine if the NLI capabilities that T5 learned for MNLI extend to other NLI datasets. The MNLI verbalizer is not included to increase readability.

Evaluation Scheme	Accuracy
String comparison	78.50%
Baseline prediction	77.91%

Table 4.31: Accuracy of T5-base on the SNLI testset for with evaluation schemes *string comparision* (see Subsection 4.1.2) Subsection 4.1.2 and baseline prediction Subsection 4.1.6 (using the SNLI trainset as baseline).

4.6.1 The Stanford Natural Language Inference (SNLI) Corpus

In this subsection, I will test if capabilities that T5 has to solve MNLI also translate to solving SNLI [5]. To that end, I will use the baseline-PVP (see Section 4.3) for evaluation. Table 4.31 shows the accuracy of T5-base on the SNLI testset with different evaluation schemes. T5 achieves an accuracy that is far above a random baseline. I conclude that the patterns that T5 uses to solve MNLI are also helpful to solve SNLI.

4.6.2 The Adversarial Natural Language Inference (ANLI) Corpus

This subsection will be concerned with the question if the NLI capabilities of T5 with MNLI also extend to ANLI [38]. ANLI has three different testsets which are also called rounds. Each round increases in difficulty. Table Subsection 4.6.2 shows the accuracy of T5 on all rounds using string comparison (see Subsection 4.1.2) and the baseline-PVP (see Section 4.3). T5 can achieve an accuracy that is very slightly above the random baseline for the first round, but drops to random for round two and three.

ANLI Round	Accuracy
1	38.20%
2	30.40%
3	32.08%

Table 4.32: Accuracy of T5-base on the ANLI testset for different rounds.

Pattern	RTE Set	Accuracy
MNLI	validation	79.06%
RTE	validation	66.06%
RTE (paper)	test	80.10%

Table 4.33: Accuracy of T5-base on the RTE validation set for with evaluation schemes string comparison (see Subsection 4.1.2) and the reported RTE accuracy with the RTE pattern from the paper [43].

I conclude that the NLI capabilities of T5 do not generalize well to ANLI. Of course, this has to do with the fact that ANLI was specifically designed to be hard to solve for current language models [38].

4.6.3 The Recognizing Textual Entailment (RTE) Corpus

This subsection examines whether T5's NLI capabilities with MNLI also extend to RTE [10]. RTE has only the labels entailment and not_entailment, instead of the three labels of MNLI. So if T5 outputs either neutral or contradiction, I will count it as not_entailment. Since RTE is part of the GLUE [57] benchmark, the test set is only available after submitting a model. Instead, I use the publically available validation set for my analysis. Since T5 was presumably already trained on RTE, the validation set was most likely also used. The results in this subsection should therefore be taken with a grain of salt.

However, another consequence of T5 being already trained on RTE is that I can compare my results with the actual RTE pattern. Table 4.33 shows three results. The first two show the accuracy on the RTE validation set when using the MNLI or the RTE pattern (see Table 2.2 for how the patterns are constructed). The last result is the reported accuracy of the paper [43] on the testset of RTE using the RTE pattern.

First, the results show that there is a mismatch between what I measure and how it is reported in the paper. This could again be because the version of T5 that I am using was trained in a multi-task fashion, whereas the version of T5 from the paper [43] allowed for individual fine-tuning. Second, using the MNLI pattern increases the accuracy by 13% for the validation set compared to the RTE pattern.

This might be a sign that the NLI capabilities of T5 could be improved by careful analysis and combination of patterns.

4.6.4 Conclusion of Experiment 3

I conclude that T5 with the MNLI pattern is quite robust to new NLI datasets. The accuracy on SNLI is far above the random baseline, and the accuracy on RTE even increases compared to the RTE pattern. Only for ANLI the results of T5 with the MNLI pattern are not impressive (it does not exceed a random baseline). However, this certainly has to do with the fact that ANLI was specifically designed to be hard to solve for models that are already good on other NLI benchmarks.

5 Conclusion

This thesis provides an extensive investigation of the robustness of the NLI capabilities of T5. In the first chapter, I provide an extensive overview of background information necessary to understand this work.

The main chapter is subdivided in three experiments. Compared to most previous work, I do not focus on developing methods to increase the accuracy of language models through patterns. Instead, I focus on hand-crafted variations and combinations of already known patterns and investigate how those impact NLI performance. Note that the experiments are neither exhaustive and nor do I strive to make them be. Instead, I should serve as an explorative endeavor to gain an intuition about the robustness of the NLI capabilities of T5.

In the first experiment, I explored how robust T5 is to slight variations in the MNLI pattern. I found out that the robustness significantly depends on the type of variations. Some variations lead to no decrease in accuracy at all, some make the accuracy drop heavily (though still far above the most frequent class baseline) and some make the accuracy drop to 0%, as T5 does not recognize that it has to solve a NLI task anymore. For further work, it might be worthwhile to employ automated methods to induce the slight changes in the MNLI prompt. This would provide a more complete view of the corresponding effects.

In the second experiment, I analyzed how robust the NLI capabilities of T5 are when using different patterns from tasks that T5 was already trained on. The results again depended on the pattern. While I was able to achieve an accuracy that exceeded the most frequent class baseline with all patterns that could be (partially) logically equivalent cast, that accuracy varied substantially. For the tasks that could not be logically equivalent cast (and with which T5 should therefore not have good accuracy from a human perspective) I found mixed results as well. With CoLA and SST2, accuracy did not exceed the most frequent class baseline. However, for QNLI and STSB, my analysis exceeded this baseline. This indicates that T5 partially uses statistical cues for its MNLI prediction. In further work, one could confirm those results by constructing a NLI dataset that specifically controls for those cues.

In the third experiment, I looked at how the MNLI capabilities of T5 translate to other NLI datasets. For SNLI and RTE I found that T5 achieves an accuracy that is far above the most frequent class baseline. For RTE, I even discovered that T5 can achieve better accuracy with the MNLI pattern instead of the RTE pattern. For ANLI T5 does not exceed random accuracy. The results indicate that T5 might not have achieved its maximum performance yet and could be improved by further combining tasks.

All in all, I conclude that the robustness of the NLI skills of T5 depends greatly on which type of perturbation is applied.

Beyond that, it might be a worthwhile endeavor to extend this analysis to other language models and to other tasks. Among other things, this might lead to the discovery of even more statistical cues and as a result a better understanding of the general capabilities of large language models.

Bibliography

[1] Aishwarya Agrawal, Dhruv Batra, and Devi Parikh. "Analyzing the Behavior of Visual Question Answering Models". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 1955–1960. DOI: 10.18653/v1/D16-1203. URL: https://www.aclweb.org/anthology/D16-1203.

[2] Jay Alammar. "The Illustrated Transformer". In: (2018). URL: https://jalammar.github.io/illustrated-transformer/.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2016. arXiv: 1409.0473 [cs.CL].

[4] John Bateman and Michael Zock. "Natural Language Generation". In: *The Oxford Handbook of Computational Linguistics* (Jan. 2012). DOI: 10.1093/oxfordhb/9780199276349.013.0015.

[5] Samuel R. Bowman et al. "A large annotated corpus for learning natural language inference". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2015.

[6] John S. Bridle. "Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition". In: *Neurocomputing*. Ed. by Françoise Fogelman Soulié and Jeanny Hérault. Berlin, Heidelberg: Springer Berlin Heidelberg, 1990, pp. 227–236. ISBN: 978-3-642-76153-9.

[7] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL].

[8] Daniel Cer et al. "SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation". In: *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Vancouver, Canada: Association for Computational Linguistics, Aug. 2017, pp. 1–14. DOI: 10.18653/v1/S17-2001. URL: https://www.aclweb.org/anthology/S17-2001.

[9] Gobinda G. Chowdhury. "Natural language processing". In: *Annual Review of Information Science and Technology* 37.1 (2003), pp. 51–89. DOI: https://doi.org/10.1002/aris.1440370103. eprint: https://asistdl.onlinelibrary.wiley.com/doi/pdf/10.1002/aris.1440370103. URL: https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/aris.1440370103.

[10] Ido Dagan, Oren Glickman, and Bernardo Magnini. “The PASCAL Recognising Textual Entailment Challenge”. In: *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment.* Ed. by Joaquin Quiñonero-Candela et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 177–190.

[11] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* 2019. arXiv: `1810.04805 [cs.CL]`.

[12] William B. Dolan and Chris Brockett. “Automatically Constructing a Corpus of Sentential Paraphrases”. In: *Proceedings of the Third International Workshop on Paraphrasing (IWP2005).* 2005. URL: `https://www.aclweb.org/anthology/I05-5002`.

[13] J. Elman. “Finding Structure in Time”. In: *Cogn. Sci.* 14 (1990), pp. 179–211.

[14] Angela Fan, Mike Lewis, and Yann Dauphin. *Hierarchical Neural Story Generation.* 2018. arXiv: `1805.04833 [cs.CL]`.

[15] *First Quora Dataset Release: Question Pairs.* URL: `https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs`.

[16] Markus Freitag and Yaser Al-Onaizan. “Beam Search Strategies for Neural Machine Translation”. In: *Proceedings of the First Workshop on Neural Machine Translation* (2017). DOI: `10.18653/v1/w17-3207`. URL: `http://dx.doi.org/10.18653/v1/W17-3207`.

[17] Tianyu Gao, Adam Fisch, and Danqi Chen. *Making Pre-trained Language Models Better Few-shot Learners.* 2020. arXiv: `2012.15723 [cs.CL]`.

[18] Robert Geirhos et al. “Shortcut learning in deep neural networks”. In: *Nature Machine Intelligence* 2.11 (2020), pp. 665–673. ISSN: 2522-5839. DOI: `10.1038/s42256-020-00257-z`. URL: `http://dx.doi.org/10.1038/s42256-020-00257-z`.

[19] John Haugeland. *Artificial intelligence: The very idea.* MIT press, 1989.

[20] Pengcheng He et al. *DeBERTa: Decoding-enhanced BERT with Disentangled Attention.* 2021. arXiv: `2006.03654 [cs.CL]`.

[21] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: `10.1162/neco.1997.9.8.1735`. URL: `https://doi.org/10.1162/neco.1997.9.8.1735`.

[22] Ari Holtzman et al. *Learning to Write with Cooperative Discriminators.* 2018. arXiv: `1805.06087 [cs.CL]`.

[23] Ari Holtzman et al. *The Curious Case of Neural Text Degeneration.* 2020. arXiv: `1904.09751 [cs.CL]`.

[24] Zhengbao Jiang et al. *How Can We Know What Language Models Know?* 2020. arXiv: `1911.12543 [cs.CL]`.

[25] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. *A Convolutional Neural Network for Modelling Sentences.* 2014. arXiv: `1404.2188 [cs.CL]`.

[26] Zhenzhong Lan et al. *ALBERT: A Lite BERT for Self-supervised Learning of Language Representations*. 2020. arXiv: 1909.11942 [cs.CL].

[27] Mike Lewis et al. *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension*. 2019. arXiv: 1910.13461 [cs.CL].

[28] Xiang Lisa Li and Percy Liang. *Prefix-Tuning: Optimizing Continuous Prompts for Generation*. 2021. arXiv: 2101.00190 [cs.CL].

[29] Jiachang Liu et al. *What Makes Good In-Context Examples for GPT-3?* 2021. arXiv: 2101.06804 [cs.CL].

[30] Yinhan Liu et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. arXiv: 1907.11692 [cs.CL].

[31] Yao Lu et al. *Fantastically Ordered Prompts and Where to Find Them: Overcoming Few-Shot Prompt Order Sensitivity*. 2021. arXiv: 2104.08786 [cs.CL].

[32] Marie-Catherine de Marneffe, M. Simons, and J. Tonhauser. "The CommitmentBank: Investigating projection in naturally occurring discourse". In: 2019.

[33] R. Thomas McCoy, Ellie Pavlick, and Tal Linzen. *Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference*. 2019. arXiv: 1902.01007 [cs.CL].

[34] Tomas Mikolov et al. "Distributed representations of words and phrases and their compositionality". In: *arXiv preprint arXiv:1310.4546* (2013).

[35] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781 [cs.CL].

[36] Vincent C. Müller and Nick Bostrom. "Future Progress in Artificial Intelligence: A Survey of Expert Opinion". In: *Fundamental Issues of Artificial Intelligence*. Ed. by Vincent C. Müller. Cham: Springer International Publishing, 2016, pp. 555–572. ISBN: 978-3-319-26485-1. DOI: 10.1007/978-3-319-26485-1_33. URL: https://doi.org/10.1007/978-3-319-26485-1_33.

[37] Aakanksha Naik et al. "Stress Test Evaluation for Natural Language Inference". In: *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, Aug. 2018, pp. 2340–2353. URL: https://www.aclweb.org/anthology/C18-1198.

[38] Yixin Nie et al. *Adversarial NLI: A New Benchmark for Natural Language Understanding*. 2020. arXiv: 1910.14599 [cs.CL].

[39] paperswithcode. "Natural Language Inference on MultiNLI Leaderboard". In: (2003). URL: https://paperswithcode.com/sota/natural-language-inference-on-multinli.

[40] Guanghui Qin and Jason Eisner. *Learning How to Ask: Querying LMs with Mixtures of Soft Prompts*. 2021. arXiv: 2104.06599 [cs.CL].

[41] A. Radford and Karthik Narasimhan. "Improving Language Understanding by Generative Pre-Training". In: 2018.

[42] Alec Radford et al. “Language Models are Unsupervised Multitask Learners”. In: 2019.

[43] Colin Raffel et al. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer.* 2020. arXiv: `1910.10683 [cs.LG]`.

[44] Pranav Rajpurkar et al. *SQuAD: 100,000+ Questions for Machine Comprehension of Text.* 2016. arXiv: `1606.05250 [cs.CL]`.

[45] Sebastian Ruder. *A Review of the Neural History of Natural Language Processing.* `http://ruder.io/a-review-of-the-recent-history-of-nlp/`. 2018.

[46] Ivan Sanchez, Jeff Mitchell, and Sebastian Riedel. “Behavior Analysis of NLI Models: Uncovering the Influence of Three Factors on Robustness”. 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, June 2018, pp. 1975–1985. DOI: `10.18653/v1/N18-1179`. URL: `https://www.aclweb.org/anthology/N18-1179`.

[47] Timo Schick and Hinrich Schütze. *Exploiting Cloze Questions for Few Shot Text Classification and Natural Language Inference.* 2021. arXiv: `2001.07676 [cs.CL]`.

[48] Timo Schick and Hinrich Schütze. *Few-Shot Text Generation with Pattern-Exploiting Training.* 2020. arXiv: `2012.11926 [cs.CL]`.

[49] Timo Schick and Hinrich Schütze. *It’s Not Just Size That Matters: Small Language Models Are Also Few-Shot Learners.* 2021. arXiv: `2009.07118 [cs.CL]`.

[50] Abigail See, Peter J. Liu, and Christopher D. Manning. “Get To The Point: Summarization with Pointer-Generator Networks”. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).* Vancouver, Canada: Association for Computational Linguistics, July 2017, pp. 1073–1083. DOI: `10.18653/v1/P17-1099`. URL: `https://www.aclweb.org/anthology/P17-1099`.

[51] Rico Sennrich, Barry Haddow, and Alexandra Birch. *Neural Machine Translation of Rare Words with Subword Units.* 2016. arXiv: `1508.07909 [cs.CL]`.

[52] Taylor Shin et al. *AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts.* 2020. arXiv: `2010.15980 [cs.CL]`.

[53] Richard Socher et al. “Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank”. In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing.* Seattle, Washington, USA: Association for Computational Linguistics, Oct. 2013, pp. 1631–1642. URL: `https://www.aclweb.org/anthology/D13-1170`.

[54] Sandeep Subramanian et al. *Learning General Purpose Distributed Sentence Representations via Large Scale Multi-task Learning.* 2018. arXiv: `1804.00079 [cs.CL]`.

[55] Derek Tam et al. *Improving and Simplifying Pattern Exploiting Training.* 2021. arXiv: `2103.11955 [cs.CL]`.

[56] Ashish Vaswani et al. *Attention Is All You Need.* 2017. URL: `https://arxiv.org/pdf/1706.03762`.

[57] Alex Wang et al. *GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding*. 2019. arXiv: 1804.07461 [cs.CL].

[58] Jianyu Wang et al. *Visual Concepts and Compositional Voting*. 2017. arXiv: 1711.04451 [cs.CV].

[59] Sinong Wang et al. *Entailment as Few-Shot Learner*. 2021. arXiv: 2104.14690 [cs.CL].

[60] Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. "Neural Network Acceptability Judgments". In: *arXiv preprint arXiv:1805.12471* (2018).

[61] Sean Welleck et al. *Neural Text Generation with Unlikelihood Training*. 2019. arXiv: 1908.04319 [cs.LG].

[62] Wikipedia. *History of artificial intelligence — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=History%20of%20artificial%20intelligence&oldid=1026015479. [Online; accessed 02-June-2021]. 2021.

[63] Adina Williams, Nikita Nangia, and Samuel Bowman. "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, 2018, pp. 1112–1122. URL: http://aclweb.org/anthology/N18-1101.

[64] Thomas Wolf et al. *HuggingFace's Transformers: State-of-the-art Natural Language Processing*. 2020. arXiv: 1910.03771 [cs.CL].

[65] Tom Young et al. *Recent Trends in Deep Learning Based Natural Language Processing*. 2018. arXiv: 1708.02709 [cs.CL].

[66] Jingqing Zhang et al. *PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization*. 2020. arXiv: 1912.08777 [cs.CL].

[67] Yin Zhang, Rong Jin, and Zhi-Hua Zhou. "Understanding bag-of-words model: A statistical framework". In: *International Journal of Machine Learning and Cybernetics* 1 (Dec. 2010), pp. 43–52. DOI: 10.1007/s13042-010-0001-0.

[68] Tony Z. Zhao et al. *Calibrate Before Use: Improving Few-Shot Performance of Language Models*. 2021. arXiv: 2102.09690 [cs.CL].

[69] Chujie Zheng et al. *Topic-Aware Abstractive Text Summarization*. 2020. arXiv: 2010.10323 [cs.CL].