Dialog generation using language models

Bachelor Thesis

A thesis submitted to attain the degree of

Bachelor of Science

by

Kevin Bönisch

written at the

Department of Text Technology

at the Goethe-Universität Frankfurt am Main

Head: Prof. Dr. A. Mehler

July 27, 2023

Abstract

Large language models have become widely available to the general public, especially due to ChatGPT's release. Consequently, the AI community has invested much effort into recreating language models of the same caliber as ChatGPT, since the latter is still a technical blackbox. This thesis aims to contribute to that cause by proposing R.O.B.E.R.T., a Robotic Operating Buddy for Efficiency, Research and Teaching. In doing so, it presents a first implementation of a lightweight environment which produces tailor-made, instruction-following language models with a heavy focus on conversational capabilities that instruct themselves into a given domain-context. Within this environment, the generation of datasets, the fine-tuning process and finally the inference of a unique R.O.B.E.R.T. instance are all carried out as part of an automated pipeline.

Zusammenfassung

Große Sprachmodelle sind inzwischen für die breite Öffentlichkeit zugänglich, insbesondere durch die Veröffentlichung von ChatGPT. Infolgedessen hat die KI-Gemeinschaft viel Mühe in die Erstellung von Sprachmodellen gleichen Kalibers wie ChatGPT investiert, da Letzteres immer noch eine technische Blackbox ist. Die vorliegende Thesis möchte einen Beitrag dazu leisten, indem sie R.O.B.E.R.T., einen Robotic Operating Buddy for Efficiency, Research and Teaching vorstellt. Dabei präsentiert sie die erste Implementierung einer leichtgewichtigen Umgebung, die maßgeschneiderte, anweisungsgetreue Sprachmodelle mit einem starken Fokus auf Konversationsfähigkeit erzeugt, die sich selbst in einen gegebenen Domänenkontext einarbeiten. Innerhalb dieser Umgebung werden die Generierung von Datensätzen, der Feinabstimmungsprozess und schließlich die Inferenz einer einzigartigen R.O.B.E.R.T.-Instanz als Teil einer automatisierten Pipeline durchgeführt.

Contents

List of Figures

List of Tables

1. Introduction

The ability to initiate and follow dialogs is a crucial task for language models since it enables them to interact with human input in a natural and corresponding way. But doing so is not a trivial task. Towards solving this problem, Sutskever, Vinyals, and Q. V. Le (2014) proposed a new sequence to sequence model that enabled DNNs (Deep Neural Networks) to handle sequences of unknown lengths since thus far these were limited to inputs and targets represented by encoded vectors of fixed dimensionality, which are insufficient for human input. Adapting this model, Vinyals and Q. Le (2015) were able to create simple conversations by predicting the next sentence on the basis of previous ones. Due to the simplicity of the prediction however, this approach has various limitations - especially when capturing the goal of a human conversation, which is information exchange rather than sentence prediction. (Vinyals and Q. Le 2015, p. 2).

Since then, the research in this field has advanced significantly, mainly due to the introduction of the Transformer architecture (Vaswani et al. 2017), which brought forth models like GPT-1 (Radford, Narasimhan, et al. 2018), GPT-2 (Radford, Wu, et al. 2019) and GPT-3 (T. Brown et al. 2020). With the introduction of ChatGPT OpenAI (2022) and GPT-4 (OpenAI 2023) new state of the art conversational language models have been released.

1.1. Motivation

Since ChatGPT was released, much effort has been invested into the objective of making conversational and instruction-following language models more accessible to the AI community and public in general. While ChatGPT shows impressive results, it is still very much a technical blackbox, restrained by general concepts and monetary efforts. Projects like ColossalAI (Bian et al. 2021), Stanford Alpaca (Taori et al. 2023) or GPT4ALL (Anand et al. 2023) aim to whiten this blackbox by providing enviroments and language models that can be finetuned and used locally - often even without the need for larger GPU infrastructures. This thesis is an effort to contribute to that cause.

1.2. The Objective

In this thesis I propose a first instance of R.O.B.E.R.T. (Robotic Operating Buddy for Efficiency, Research and Teaching): a domain-based, instruction-following language model with a heavy focus on conversational capabilities that instructs itself into the given domain-context. By doing so, I want to introduce an environment that is completely accessible to the public and can be adapted to various specific use cases without the need for large datasets or costly resources. My goal is to provide a language model that is an expert on a given context, but also knows its limitations and thus minimizes the probabilities of hallucination and biasing.

2. Related Work

2.1. Large Language Models

Language Modeling has been a fundamental task in natural language processing for several decades. P. F. Brown et al. (1990) published an influential paper that introduces the concept of n-gram language models based on statistical probabilities of a word given all the other words that precede it in a sentence. Odell (1995) then emphasized the importance of context in language modeling for speech recognition. A concept, that would be crucial for the large-scale language models used today. For the first time, Bengio, Ducharme, and Vincent (2000) then introduced the use of neural networks for language modeling, which eventually led to the creation of Transformer-based architectures (Vaswani et al. 2017) widely used today. Finally, Grave, Joulin, and Usunier (2016, p. 1) proposed the concept of *Neural Cache Models* using a continuous cache to improve the efficiency and performance of neural language models. This would eventually lead to the concepts of pre-training and fine-tuning a large language model, also commonly used today.

2.1.1. Pre-training and Fine-tuning

Peters et al. (2018) introduced a new type of deep contextualized word representation and addressed the issue of traditional word embeddings (see Mikolov et al. (2013) or Turian, Ratinov, and Bengio (2010)) by providing context-dependent word representations. The premise of these *ELMos* (Embeddings from Language Models, see Peters et al. (2018, p. 1)) is to pre-train a bidirectional LSTM (Long Short-Term Memory) language model on a large corpus using a language modeling objective. By doing so, the model learns to predict the next word in a sentence given both the preceding and following words, thus capturing contextualized dependencies in the embedding. Since then, this concept has spawned multiple other approaches for pre-training, such as RoBERTa (Liu et al. 2019) or T5 (Raffel et al. 2020), which utilize larger corpora and masked language modeling.
Pre-training is done on large, unlabeled datasets. In the context of T5, the *Colossal Clean Crawled Corpus* (C4) dataset was used, which is 750GB of text (Raffel et al. 2020, p. 7).

Having pre-trained a model, it can now be fine-tuned to complete a specific downstream task such as question answering or text summarization. Fine-tuning a pre-trained language model only takes a labeled fraction of the datasets needed to pre-train it (Howard and Ruder 2018, p. 6).

2.1.2. LLaMA

Touvron et al. (2023) have introduced LLaMa, a collection of foundational language models ranging from 7B to 65B parameters, showing that LLaMA-13B outperforms GPT-3 on multiple benchmarks and therefore reaching state of the art results. It was pre-trained on a variety of known datasets, including C4, totaling in 4.75TB of text.

2.2. Instruction-following Models

Since the release of ChatGPT, a large variety of instruction-following language models have been created within the AI community. This is due to the release of LLaMA, on which most of these models are based upon, the publication of the Self-Instruct paper (Wang et al. 2023) and the release of ChatGPT itself.

Within the afore mentioned Self-Instruct paper, the authors present a pipeline that is capable of self-inducing instruction-following capabilities into pre-trained language models with minimal human-labeled date. By doing so, they were able to fine-tune GPT-3 on this semi-automated dataset, showing an improvement of 33% over GPT-3 and nearly matching the performance of InstructGPT_{001} by Ouyang et al. (2022).

This process of self-generating labeled data by a strong language model to fine-tune language models has then been widely adapted, often by using ChatGPT to generate the instructions and outputs for the data. This led to models such as:

- **Stanford Alpaca** (Taori et al. 2023), an instruction-following model which was fine-tuned with 52k datasets, mostly created by ChatGPT on LLaMA 7B.
- **Vicuna-13B** (Chiang et al. 2023), a chatbot that was fine-tuned with 70k datasets, solely constisting of conversations with ChatGPT shared by ShareGPT (domeccleston 2022) on LLaMA 13B.
- **ColossalChat** (Bian et al. 2021), an instruction-following model that was fine-tuned with 104k datasets, also constisting of conversations with ChatGPT on LLaMA.

These models aim to mimic or even outperform ChatGPT's performance, claiming that Vicuna-13B achieves more than 90% of ChatGPT's quality and Stanford Alpaca claiming to have "very similar performance" (Taori et al. 2023, p. 2). It is important to note however that these claims are based solely on preliminary human evaluations rather than extensive, well-designed systematic tests.

2.3. Limitations

While the models listed in Section 2.2 achieve stellar performances, given the relatively small resources used to create them, they do come with limitations - most prominently the act of hallucination (Taori et al. 2023) and lack of simple mathematical abilities like counting items in a list (Bian et al. 2021). Further limitations also include biasing, toxicity and the lack of multi-turn chatting abilities (Bian et al. 2021). Moreover, given the sizeable pre-training corpus, these models may have a wide range of broad knowledge, but they lack extensive domain-specific knowledge that is not accessible to the public.

3. The Environment

With R.O.B.E.R.T. (Robotic Operating Buddy for Efficiency, Research and Teaching), I want to address the limitations introduced in Section 2.3 and propose an environment towards solving those.

3.1. Capabilities

To do so, R.O.B.E.R.T. is going to be an expert on a very specific domain-context, which doesn't need to be public knowledge. In order to accomplish this, the training environment offers a pipeline that accepts custom-defined, dynamic contexts, which can be described by supplying text files with bulletpoints. R.O.B.E.R.T. then gets fine-tuned on datasets, specifically generated for the given context. Upon fine-tuning, R.O.B.E.R.T. will be able to answer questions and follow instructions within the given context, but excuses himself when facing an input that is outside of the context's scope, minimizing the risk of hallucination, bias and toxicity.

Additionally, I place a lot of emphasis on R.O.B.E.R.T.'s ability to carry on conversations in an organic, proactive, and corresponding manner in order to be an authentic assistant and match ChatGPT's conversation skills.

3.2. Dataset Generation

Firstly, a domain-context has to be chosen, on which one R.O.B.E.R.T. instance is being fine-tuned on. For this thesis, I created a made-up scenario that enables R.O.B.E.R.T. to play the role of "Rob", an assistant to TTL Corporation students in virtual reality. To do so, everything I want Rob to be and to know has to be outlined in a *parameters.txt* file. Figure 3.1 shows three exemplary bullet points from a total of 30 (the complete list can be found in the appendix, Figure A.1).

The amount of bullet points in the *parameters.txt* determines the quantity of datasets required to sufficiently fine-tune R.O.B.E.R.T.. The bigger the domain-context, the more datasets are required. Chapter 4 will elaborate on this statement by trying to find a benchmark to solve this problem. Once the context has been outlined by the *parameters.txt*, the pipeline starts the dataset generation.

- The roboter's name is Rob. He is a Virtual Reality Assistant. He works for the Text Technology Lab.
- The Va. Si. Li. Lab is a virtual reality teaching platform, made by the Text Technology Lab. It simulates real life scenarios in Virtual Reality.
- TTL Corporation is a college. It has 100 associates and 500 students.
[...]

Figure 3.1.: Abstract of the *parameters.txt* file used to outline the domain-context.

3.2.1. Intruction-Following Datasets

Adapting the principles of the Self-Instruct paper (Wang et al. 2023) and projects like Stanford-Alpaca (Taori et al. 2023) or Vicuna (Chiang et al. 2023), I used strong language models to auto-generate instruction-following datasets. Each dataset has the following form:

```
{
    "instruction": "Can I book Room A13 for a seminar as a student?",
    "input": "",
    "output": "Absolutely, Room A13 is available for students!"
},
```

Figure 3.2.: Example dataset

Whereas *instruction* and *output* are mandatory and *input* is optional.

To generate datasets in that particular form, which are designed for the specific domain-context, I implemented two main steps.

Step 1: Generate the instruction

To do so, I created an *input_generateQ.txt* file as a template, shown in Figure 3.3a, where *[CONTEXT]* will be replaced with one to three randomly chosen bullet points from the *parameters.txt* file and *[QUESTION_TYPE] [TYPE]* will be filled in with different instructions such as: "simple question" or "rather long instruction".

The resulting prompt will be sent to the language model, which in return prints out a question that is then stored. Figure 3.3b shows an example of this.

Step 2: Generate the output

For that, step one will be repeated with a different template, this time incorporating the instruction generated in the first step (see Figure 3.4). Finally, the generated instruction along with the output is stored as one dataset.

You are a student. You have the following context:
[CONTEXT]
#####
Rob, the virtual assistant roboter, will try to answer the questions/instructions you have. Formulate a [QUESTION_TYPE] [TYPE] for him, precisely about this context.

(a) Template for generating an instruction

You are a student. You have the following context:
- Michelle is a student. She is smart and has good grades. She really likes Tom. Michelle often is in the kitchen.
- Mister Stein is in his bureau. He is in charge of the students. He's working in his bureau.
#####
Rob, the virtual assistant roboter, will try to answer the questions/instructions you have. Formulate a short question for him, precisely about this context.

Output: What is Mister Stein doing in his bureau?

(b) Example for generating an instruction

Figure 3.3.: Instruction Generation

You are Rob, a robotic Virtual Reality Assistant. You know the following:
[CONTEXT]
#####
You only act or answer when you are truthfully sure you can do that. Don't make up information.
#####
A student says/asks: [QUESTION]
#####
Answer on the basis of the given context. Answer only when you think you know the answer. Otherwise excuse yourself, followed by a short explanation.

(a) Template for generating an output

You are Rob, a robotic Virtual Reality Assistant. You know the following:
- Michelle is a student. She is smart and has good grades. She really likes Tom. Michelle often is in the kitchen.
- Mister Stein is in his bureau. He is in charge of the students. He's working in his bureau.
#####
You only act or answer when you are truthfully sure you can do that. Don't make up information.
#####
A student says/asks: What is Mister Stein doing in his bureau?
#####
Answer on the basis of the given context. Answer only when you think you know the answer. Otherwise excuse yourself, followed by a short explanation.

Output: Mister Stein is working in his bureau.

(b) Example for generating an output

Figure 3.4.: Output Generation

3.2.2. The Language Model

In Section 3.2.1, I've frequently alluded to the usage of a "strong" language model to generate the datasets as described. Choosing the right language model is crucial to the whole project, since the generated datasets decide over the total quality of the fine-tuned R.O.B.E.R.T.. Consequently, the obvious choice would be to use ChatGPT, as it is very reliant, has been successfully used by Stanford-Alpaca and Vicuna 13B before and offers an easy to use API that doesn't require an extensive GPU environment.

While I did use ChatGPT, I also incorporated GPT4All, which is an "[...] ecosystem to train and deploy powerful and customized large language models that run locally on consumer grade CPUs." (Anand et al. 2023, p. 0). GPT4All provides a list of trained language models to choose from. I decided to use the Vicuna 13B model[1] in the scope of this thesis, since it is an instruction-following-based model which achieves state of the art results. By doing so, I hope to achieve similar results, thus making the usage of ChatGPT optional and not mandatory. Towards answering this question, I trained multiple R.O.B.E.R.T. versions, which will be further evaluated in Chapter 4.

3.2.3. Dialog Capabilities

As I've already mentioned, I want to emphasize R.O.B.E.R.T.'s conversational abilities. To do so, two requirements have to be met:

1. R.O.B.E.R.T. has to store some form of context or chat history in order to truthfully follow a conversation and answer on the basis of that. Also, contextual dependencies within a dialog, such as questions like "What does *that* mean.", have to be answered.

2. R.O.B.E.R.T. can't only follow instructions, but has to be proactive about it. Having a conversation also includes asking follow-up questions and forming statements to specify the instructions.

To meet those requirements, I propose cooperating dialog-driven datasets into the fine-tuning process of R.O.B.E.R.T.. Besides generating datasets as shown in Section 3.2.1, I added the automatic creation of datasets that include an *input* (see Figure 3.2), containing chat-histories. To do so, I created two new template text files to better prompt the strong language model for the dialog-driven outputs. I then structurally followed the two steps described in Section 3.2.1, but needed to add steps and modify them.

Step 1: Fetch an existing instruction-following dataset

The creation of dialog-driven datasets within the pipeline is queued after the creation of instruction-following datasets. Consequently, it is resourceful to take random dataset instructions and use them as the start of the dialog instead of having to generate one again.

Dialog so far:

Student: Rob, can you book Room A13 for me?

[1] Precisely: ggml-vicuna-13b-1.1-q4_2.bin

Step 2: Create R.O.B.E.R.T.'s response

To create the response, I modified the template for generating an output shown in Figure 3.4 to additionally pass in the chat history so far and to tell the language model to incorporate it into "Rob's" answer. The output continues the dialog.

Dialog so far:
Student: Rob, can you book Room A13 for me?
Rob: Yes, I can book Room A13 for you. Please provide me with the date and time you would like to book it for.

Step 3: Create the student's response

To do so, I modified the template for generating an instruction shown in Figure 3.3b to additionally pass in the chat history so far and tell the language model to incorporate it into the student's instruction.

Dialog so far:
Student: Rob, can you book Room A13 for me?
Rob: Yes, I can book Room A13 for you. Please provide me with the date and time you would like to book it for.
Student: Can you book it for tomorrow at 3 pm?

Step 4: Store and repeat

For each student's instruction and "Rob's" response, a new dataset is stored, this time also utilizing the optional *input* field to store the dialog-history. After that, steps two and three are repeated to generate as much dialog-turns as seen fit. This number of turns however is limited by the prompt size of the language model generating the next continuations and heavily influences the speed at which these continuations are being printed out. As a consequence, I set the maximum dialog turns for the dataset generation to six for this thesis. After that, a new dialog is started with step one.

3.2.4. Paraphrasing

"Paraphrase generation aims to improve the clarity of a sentence by using different wording that conveys similar meaning" (Egonmwan and Chali 2019, p. 1). In the same paper, the authors propose a sequence-to-sequence model alongside a transformer to use both models for paraphrase generation. The resulting experiments show improvements over state of the art performances (Egonmwan and Chali 2019, p. 4). Using the principle of paraphrasing, Okur, Sahay, and Nachman (2022) have successfully shown that as a data augmentation technique to train Natural Language Understanding (NLU) models on small-size task-specific datasets, paraphrase generation improves the overall quality of the NLU-model.

To expand upon this, I added a paraphrasing layer into the dataset generation pipeline

Original dataset:

```
{
    "instruction": "I need to go to the bathroom by the kitchen.",
    "input": "",
    "output": "I can take you to there."
},
```

Paraphrasing *instruction* and *output* six times:
Instruction:
["I need to go to the bathroom next to the kitchen.", "I want to go to the bathroom next to the kitchen.", "Rob, please take me to the bathroom.", "Rob, I need to go to the bathroom next to the kitchen.", "You should take me to the bathroom next to the kitchen.", "Take me to the bathroom next to the kitchen."]
Output:
["I can take you to the bathroom.", "I can take you to the bathroom next to the kitchen.", "I will take you to the bathroom next to the kitchen.", "I'll take you to the bathroom next to the kitchen.", "I can bring you to the bathroom next to the kitchen.", "You can go to the bathroom next to the kitchen."]

Figure 3.5.: Paraprashing a dataset

using *pegasus_paraphrase*[2], a PEGASUS model fine-tuned for paraphrasing, available in the Hugging Face community. By doing so, I hope to augment the datasets shown in Section 3.2.1 and 3.2.3 to further solidify the conversational capabilities of R.O.B.E.R.T. and to minimize the data generation and thus resources. For that, I paraphrased the *instruction* and *output* of a dataset six times (see Figure 3.5), potentially sixfolding the amount of datasets. The results will be evaluated in Chapter 4.

3.3. Fine-tuning

I fune-tuned several versions of R.O.B.E.R.T. on the LLaMA model (Touvron et al. 2023). To do so, I used the more lightweight LLaMa 7B weights as my pre-trained base. As the original LLaMA code is GPL-licensed and I needed to incorporate a variety of different projects and tools, I chose Lit-LLaMA (Lightning-AI 2023), an independent implementation of LLaMA under the Apache 2.0 license.
The fine-tuning process of each R.O.B.E.R.T. version took five to six hours (using LoRA) on a NVIDIA L4 with 24GB Memory, provided by a Google Compute Engine. Since Google offered me 300$ worth of starting balance and I still have 40$ left after training several R.O.B.E.R.T.s, the actual cost for the fine-tuning process comes down to 0$.

[2]Available on Hugging Face: https://huggingface.co/tuner007/pegasus_paraphrase (opened 2023-07-27)

3.3.1. LoRA

When fine-tuning a large pre-trained language model, all parameters of the pre-trained model will be updated. For instance, fine-tuning GPT-3 (T. Brown et al. 2020) means updating 175 billion trainable parameters, which requires massive resources. Towards solving this problem Hu et al. (2021) have proposed Low-Rank Adaption (LoRA), a technique to freeze pre-trained model weights and instead injecting trainable rank decomposition matrices into the Transformer layers. This massively reduces the amount of trainable parameters for downstream tasks. Despite grealty reducing the number of trainable parameters and thus GPU requirements, LoRA performs on-par or even better than other adaption methods on the GLUE benchmark (Hu et al. 2021, p. 6).
To fine-tune R.O.B.E.R.T., I will also utilize LoRA to minimize the cost and resources to create a lightweight training environment.

3.4. Versions

The dataset-generation-pipeline produces tailor-made datasets for individual contexts. This raises the issue of how many datasets are required for various context sizes. Additionally, I incorporated three techniques into the training environment:

- Creating instruction-following datasets
- Creating conversational datasets
- Paraphrasing datasets

Finally, I mentioned the usage of two strong language models in Section 3.2.2:

- ChatGPT
- GPT4All (Vicuna 13B)

By training multiple R.O.B.E.R.T. versions, I want to test the number of datasets needed, the application of various dataset generation techniques and the effectiveness of free language models against ChatGPT to eventually create comparisons and benchmarks and therefore finding the best performance. Table 3.1 lists all R.O.B.E.R.T. versions.

3.5. Technical Implementation

The source code for R.O.B.E.R.T. and its environment is publicly available on GitHub[3] and was mainly written in python. The repository is divided into two parts: the training part and the inference part. It also incorporates the already mentioned Lit-LLaMA and GPT4All repositories.

While it is not necessary to further explain every line of code, I want to shed light on some technical implementations and parameters which are crucial to the understanding of the evaluation.

[3]https://github.com/TheItCrOw/R.O.B.E.R.T. (opened 2023-07-27)

Table 3.1.: Table of relevant R.O.B.E.R.T. versions included in this thesis. The numerals always indicate the amount of datasets used in that specific generation-technique. The term "para" stands for "paraphrased" and always references the technique described in the previous column. Column "model" describes the language model used to generate the datasets of that fine-tuned R.O.B.E.R.T. instance.

	model	instruction-following	para	dialog	para	total
robert_1k	ChatGPT	1.000	-	-	-	1.000
robert_5k	ChatGPT	5.000	-	-	-	5.000
robert_10k	ChatGPT	10.000	-	-	-	10.000
robert_10k_gpt4all	GPT4All	10.000	-	-	-	10.000
robert_5k_chat_only	ChatGPT	-	-	5.000	-	5.000
robert_21k_chat_only	ChatGPT	-	-	5.000	16.000	21.000
robert_23k_para_chat	ChatGPT	6.000	12.000	5.000	-	23.000
robert_45k_chat_para	ChatGPT	12.000	12.000	5.000	16.000	45.000

3.5.1. Parameters

Every R.O.B.E.R.T. version has been implemented and tested with the following parameters:

temperature: **0.8**
top_k: **200**
The top-k parameter used during text generation to limit the number of possible tokens that the model considers at each step.
max_new_tokens: **100**
Max amount of tokens R.O.B.E.R.T. may utilize to answer.
dtype: **float32**
Datatype and level of precision of the model's values and variables. Consumes ~24GB of GPU but can be lowered to ~14GB when using float16 precision.
r:[4] **8**
The rank of the update matrices, expressed in int.
dropout: **0.05**
alpha: **8**

3.5.2. Context

To provide R.O.B.E.R.T. with contextual awareness, a mechanism to store the last dialog-turns has to be implemented. This is done by again using the optional *input* field in the instruction-prompt for each R.O.B.E.R.T. instance. There, the dialog-history will be injected which is stored in an array within the *robert* class.

[4]See also: https://huggingface.co/docs/peft/conceptual_guides/lora (opened 2023-07-27)

4. Evaluation

4.1. Automatic Methods of Evaluation

To evaluate R.O.B.E.R.T.'s overall performance and to benchmark the various versions showcased in Table 3.1, the test methods have to be defined. To do so, I have evaluated multiple approaches, most of which I deemed useless for this specific project.

4.1.1. ROUGE

C.-Y. Lin (2004) has introduced the wide-spread ROUGE measures, which are primarily used for evaluating summaries. ROUGE calculates three scores: precision, recall and F1, all based on n-gram matches between a generated text and the reference. Originally used for evaluating summaries, I still tested the ROUGE measures by running 1.000 instruction-following datasets also used for the fine-tuning on each R.O.B.E.R.T. version and calculated the generated response of R.O.B.E.R.T. against the original output of the dataset.

4.1.2. BLEU Score

In addition to the ROUGE measures, I also incorporated the calculation of the BLEU score (Papineni et al. 2002), which is a metric commonly used for evaluating the quality of generated text. This score is also based on the n-gram overlap but also incorporates several other factors such as brevity penalty (Papineni et al. 2002, p. 5) and modified n-gram precision (Papineni et al. 2002, p. 3). Parallel to the ROUGE measures, the BLEU score also needs a reference to score R.O.B.E.R.T.'s output. As a consequence, besides calculating the ROUGE measures on 1.000 instruction-following datasets, I also injected calculation of the BLEU score into the same pipeline.

Table 4.1.: ROUGE and BLEU results on the R.O.B.E.R.T. versions on 1.000 datasets

model	rouge1		rouge2		rougeL		rougeLsum		bleu
	precision	F1	precision	F1	precision	F1	precision	F1	
robert_1k	0.57	0.54	0.43	0.41	0.52	0.50	0.54	0.52	-
robert_5k	0.80	0.79	0.73	0.71	0.78	0.77	0.79	0.78	0.74
robert_10k	0.77	0.75	0.68	0.67	0.74	0.72	0.75	0.74	0.7
robert_10k_gpt4all	0.53	0.45	0.36	0.31	0.48	0.41	0.50	0.43	0.38
robert_5k_chat_only	0.67	0.51	0.44	0.37	0.56	0.46	0.57	0.48	0.42
robert_21k_chat_only	0.59	0.44	0.38	0.29	0.52	0.39	0.53	0.40	0.32
robert_23k_para_chat	0.56	0.42	0.35	0.27	0.49	0.37	0.51	0.38	0.31
robert_45k_chat_para	0.72	0.70	0.62	0.59	0.68	0.66	0.70	0.68	0.63

4.1.3. Results

Table 4.1 shows the results of the ROUGE and BLEU measures. The following can be observed:

1. *robert_5k* shows the best results, which gives the impression that 5.000 datasets is the optimal amount to train a R.O.B.E.R.T. on a context consisting of 30 bullet points.
2. *robert_10k_gpt4all* shows the worst results of all purely trained instruction-following models, giving the impression that models trained on datasets generated by GPT4All are inferior to models trained on ChatGPT datasets.
3. *robert_5k_chat_only* has, on average, better scores than his paraphrasing counterpart *robert_21k_chat_only*, giving the impression that paraphrasing to augment the datasets is not only useless, but damaging.
4. All models which had dialog-driven datasets incorporated in their fine-tuning process performed worse then the models which were fine-tuned on instruction-following datasets only. This might give the impression that this method is flawed and should be avoided.

However, a closer examination reveals that all of these assertions are false.

1. The 1.000 datasets used to calculate the ROUGE and BLEU metrics have all been generated by ChatGPT and were therefore used to fine-tune all models except *robert_10k_gpt4all*. Manual testing done by forming new instructions and looking at the given output reveals that *robert_1k* and *robert_5k* are unable to adapt to instructions which are not massively similar to those they were fine-tuned on. This defies the claim that 5.000 datasets is sufficient to create a R.O.B.E.R.T. on the given context, which will be further supported by the human evaluation.
2. Following this thought, it becomes clear why *robert_10k_gpt4all* performed so badly on this test. The datasets were not used to fine-tune this version and thus the given outputs differed greatly from the datasets targets in comparison to the other models, resulting in ”bad“ answers according to the used metrics.
3. Augmenting the datasets leads to enlarging the models vocabulary, which is, looking through the lens of ROUGE and BLEU, a problem, since these metrics are primarily based on similarity between the prediction and the target. Any deviations, whether positive or negative, will only result in a lower score. These metrics are unable to assess the efficiency of paraphrasing or the method for incorporating dialog-driven datasets into the fine-tuning process.

Towards solving these issues, it is recommended to use datasets both generated by GPT4All and ChatGPT. However, this only masks the issue. For meaningful test results, a set of completely new instructions is required, while also using different metrics to rate the models.

robert_1k	**7.67**
robert_5k	**8.71**
robert_10k	**8.68**
robert_10k_gpt4all	**7.70**
robert_5k_chat_only	**8.30**
robert_21k_chat_only	**8.11**
robert_23k_para_chat	**7.64**
robert_45k_chat_para	**8.48**

Figure 4.1.: ChatGPT results on the R.O.B.E.R.T. versions on 1.000 datasets

4.1.4. ChatGPT

Ding et al. (2023) have built a "systematically designed, diverse, informative, large-scale dataset of instructional conversations, UltraChat [...]" (Ding et al. 2023, p. 1). In doing so, they introduced a dataset which consists of 1.5 million multi-turn dialogs. To benchmark their results, they used ChatGPT to rate the questions and answers generated by their model on a scale from 1 to 10. Adapting this technique, I repeated the test described in Section 4.1.1 but this time letting ChatGPT rate the given answers. The average results are listed in Figure 4.1.

The results show that all versions perform very similarly. Even *robert_10k_gpt4all*, which was not partly fine-tuned on the datasets used for this test, reaches a solid score. At first glance, ChatGPT might seem like a reliable metric to rank the various models. This could then be tested using completely new datasets before ranking the models once more. However, it becomes obvious that ChatGPT is not a good choice when manually comparing the scores it assigns to the provided instruction and output.

The problem lies within the fact that R.O.B.E.R.T. has a distinct domain-based knowledge pool, which ChatGPT is completely unaware of. Unlike UltraChat (Ding et al. 2023), R.O.B.E.R.T. is trained on a context which is not public domain with the primary focus of conveying correct information. Therefore, ChatGPT cannot determine whether the output R.O.B.E.R.T. prints is factualy correct. To counter that, I attempted to prompt ChatGPT with the context excerpt needed to fact-check the response, but manual checks indicate that this is insufficient.

Figure 4.2 shows an example. In that case, *robert_1k* gave a factually false answer to the provided question, which becomes obvious when looking at the provided context excerpt. Nevertheless, ChatGPT assigns an 8 out of 10, implying that the given answer was not only sufficient, but nearly perfect. However, the correct answer would include that R.O.B.E.R.T. doesn't know for sure how to book a room, but would try to help the student. Therefore, I deem ChatGPT insufficient of rating the answers provided by the different R.O.B.E.R.T. versions. Prompting ChatGPT to rate 1.000 datasets of eight different R.O.B.E.R.T. versions resulted in 8.000 prompts, which costs around 60$ using *text-davinci-3*.

CONTEXT:
- Room A13 is a seminar room and available. It is right next to the kitchen. It has a projector and a whiteboard. It does not have a computer though. Room A13 is accessable for students if they book it.
- A student can pick up objects by just grabbing it.
- Tom is a student. He is lazy and always wants to take vacations. He really likes Michelle.

QUESTION:
"How could Tom book Room A13 for a seminar? "

ANSWER:
"To book Room A13 for a seminar, Tom needs to contact the college's administration office. He can do this by calling the number 010-0000."

ChatGPT Score: 8

Figure 4.2.: Example of ChatGPT rating a factually false answer with a great score.

4.2. Generating Test Instructions

Section 4.1 showcased the need for generating completely new instructions to properly test the different R.O.B.E.R.T. versions. To do so, ChatGPT or GPT4All could be used again as demonstrated in Section 3.2.1, but this thesis seeks to use ChatGPT as little as possible and it hasn't been evaluated yet whether GPT4All produces good enough datasets to be an alternative to ChatGPT.

Instead, I propose using the already generated datasets to fine-tune a counterpart to the different R.O.B.E.R.T. versions: A student. While R.O.B.E.R.T. receives instructions and produces an output, the student model generates an instruction for R.O.B.E.R.T. as its output, while considering a given context. By doing so, I'm adapting the similar works of Du, Shao, and Cardie (2017), Rao and Daumé III (2018) or Bulathwela, Muse, and Yilmaz (2023).

This leads to a self-sufficient environment in which the generated datasets are used to develop additional models that, in turn, aid in the development of better R.O.B.E.R.T.s.

4.2.1. Fine-tuning Students

To fine-tune a student model, I repeated the steps showcased in sections 3.3 and 3.2.1, but instead of creating new datasets, I used the already existing ones and extract their *instruction* fields. The instruction is the targeted *output* of the student model. I also injected an excerpt of the context this instruction is about into the optional *input* field. Figure 4.3 shows an example of a dataset used to fine-tune the student.

By doing so, I fine-tuned two student models shown in Table 4.2.

Manually testing these models reveals that *student_85k_para* is producing mostly nonsensical instructions. This is probably due to the fact that the majority of datasets used to fine-tune this model are generated by GPT4All and then paraphrased, leading to more faulty datasets if already faulty to begin with. A more detailed evaluation of GPT4All's overall performance will be presented in Section 5.5.

```
{
    "instruction": "Formulate an instruction or a question
                    for Rob about the given input",
    "input": "Mister Stein is working is his bureau.",
    "output": "What is Mister Stein doing in his burea?"
},
```

Figure 4.3.: Example dataset of a student model

Table 4.2.: Table of relevant student models for this thesis

	model	instr.-following	para	dialog	para	total
student_24k_para	ChatGPT	12.000	12.000	-	-	24.000
student_85k_para	ChatGPT+GPT4All	23.000	62.000	-	-	85.000

student_24k_para on the other hand produces high-quality instructions that are tailor-made to the context that is presented in the *input* field. Furthermore, first tests indicate that the given context is not limited to the context the student model has been fine-tuned on, meaning that the student model will formulate instructions about any context that is being presented. Figure 4.4 shows a variety of instructions generated by the student model about the given input. Consequently, *student_24k_para* will be used to generate instruction-following test-datasets.

Instruction: Formulate an instruction or a question for Rob about the given input.

Input: The Va. Si. Li. Lab is a virtual reality teaching platform made by the Text Technology Lab. It simulates real life scenarios in Virtual Reality.
Output: What features does the Va. Si. Li. Lab offer?

Input: To register for a course, you have to fill in form W8.
Output: Can you help me fill in the form W8?

Input: It takes 3 hours to watch The Lord of the Rings.
Output: Can you tell me how long it would take to watch all three movies if I watched one movie per day for three hours?

Figure 4.4.: *student_24k_para* creating instructions about any given context-input.

Student: Rob, please tell me what R.O.B.E.R.T stands for.
Rob: R.O.B.E.R.T stands for ”Robotic Operating Buddy for Efficiency, Research, and Teaching“.
Student: That's helpful!
Rob: You're welcome!
Student: Can you tell me more about your job?
Rob:I'm a robotic operating buddy for efficiency, research, and teaching. I was designed to help you with research, teaching, and efficiency.
Student: Can you help me with my homework?

Student: I am holding an object in my hand.
Rob: Yes, that is a pen.
Student: What color is it?
Rob: It's blue.
Student: Can you tell me what it's used for?

Figure 4.5.: Exemplary dialogs created by a student and robert model prompting to each other.

4.3. Generating Test Dialogs

Thus far, I only emphasized the testing of instruction-following capabilities, but it is highly necessary to test R.O.B.E.R.T.'s dialog capabilities as well. To do so, I adhere to the concept shown in Section 4.2.1 by creating another student model, which will be fine-tuned with the dialog-driven datasets generated by ChatGPT, used to also fine-tune a variety of the listed R.O.B.E.R.T. versions. But this time, instead of training the model to answer as ”Rob“, I trained the student model to continue the dialog as the ”student“ would.

Consequently, I fine-tuned *student_22k_chat_para*, consisting of 22.000 dialog-driven datasets, of which 16.000 were paraphrased. I then used this student model together with *robert_21k_chat_only_para* and have them prompt to each other for a maximum of six turns. All dialogs that ended with a students turn were then hand-filtered (since many generated dialogs with more than 4 turns were nonsensical and/or wrongly formatted) and used as test-datasets, with the instruction to each R.O.B.E.R.T. version to appropriately continue the dialog. Examples are displayed in Figure 4.5.

4.4. Human Evaluation

Sections 4.1.1, 4.1.2 and 4.1.4 have shown the need for human evaluation of completely new instructions. Using the student models showcased in Sections 4.2 and 4.3, I produced 100 test datasets that were all generated from a student's perspective containing 50 instruction-following and 50 dialog-driven prompts. Each R.O.B.E.R.T. version will be asked to complete the 100 instructions and dialogs while the resulting outputs will be rated on a Likert Scale by humans. The testers were given the following criterias to rate Rob's outputs:

- Rob should never make up any information outside of the given context scope. In order

Context: The bathroom is always available for everyone. Its right next to the kitchen. **Instruction**: Can you tell me where the bathroom is located? **Output**: The bathroom is right next to the kitchen.	**Dialog so far** Student: Can you book Room A13 for me? Rob: Yes, I can book Room A13 for you. Please let me know the specific time and date you would like to book the room. Student: The time and day would be at 4pm on Friday. **Output**: I will see to it.

Figure 4.6.: Exemplary test-datasets for humans to rate (On the left side, an instruction-following test and on the right side a dialog-driven test.)

to fact-check the given output, the relevant context excerpt is provided.

- Rob's output must never contain bias or toxicity.
- Rob's answer should be natural, proactive and polite.

Figure 4.5 displays an instruction-following and dialog-driven test-dataset given to a tester to rate. Figure A.2 in the Appendix shows a screenshot of the website.

5. Results and Conclusions

Table 5.1.: Results of the human evaluation, showcased by their average score in the given category, based on a Likert Scale (1-5). All purely instruction-following trained models are highlighted in gray, all models which have paraphrased datasets induced are highlighted yellow.

model	instruction-following	dialog-driven	total
robert_1k	**2.56**	**1.22**	**1.89**
robert_5k	3.7	2.88	3.29
robert_10k	**4.1**	2.46	3.28
robert_10k_gpt4all	3.46	1.54	2.5
robert_5k_chat_only	3.66	**3.34**	3.5
robert_21k_chat_only	3.42	3.0	3.21
robert_23k_para_chat	3.3	3.22	3,26
robert_45k_chat_para	4.0	3.22	**3,61**

The results of the human evaluation described in Section 4.4 are shown in Table 5.1. The following sections will elaborate and conclude on the basis of the given table. As Section 3.2 describes, the goal was to create "Rob", a fictional virtual reality assistant to the students of TTL Corporation, outlined by a context consisting of 30 bullet points.

5.1. Instruction-following Capabilities

robert_1k predictively scored the lowest in the instruction-following category, followed by *robert_23k_para_chat*. *robert_5k* achieves a better score of 3,7 while *robert_10k* outperforms all models in the instruction-following category with an average score of 4,1. This leads to the conclusion that, for this thesis' context of 30 bullet points, generating 1.000 datasets is not sufficient, whereas 5.000 datasets already produce good quality outputs, trumped by 10.000 datasets reaching a nearly optimal score on average.

Conclusion 1:
10.000 instruction-following datasets (without paraphrasing) are sufficient enough to produce a high-quality, instruction-following Rob.

When looking at the results of the dialog-driven category, all purely instruction-following trained models score subpar, even by a substantial amount. It is also to observe that *robert_10k* performed worse than *robert_5k*, stating that more datasets produced less quality in this specific category. This leads to conclusion number two.

Conclusion 2:
To induce contextual dialog capabilities into Rob, the use of dialog-driven datasets is not optional but mandatory.

5.2. Dialog Capabilities

Surprisingly, *robert_5k_chat_only* achieved the best score of 3,34 in the dialog-driven category with only 5.000 dialog-driven datasets, followed by *robert_23k_para_chat*. It is also *robert_5k_chat_only* that performed second best in the instruction-following category of all models which have dialog-driven datasets induced. It performs on par with *robert_5k*, which consists of instruction-following datasets only.

Conclusion 3:
While instruction-following datasets primarily induce instruction-following capabilities, dialog-driven datasets seem to train both instruction-following and dialog-driven behaviour alike, possibly rendering the usage of instruction-following datasets optional and not mandatory.

robert_5k_chat_only consisting of 5.000 purely dialog-driven datasets scored best with a 3,34 on average, which is not optimal. The initial idea of using more datasets by combining instruction-following datasets, dialog-driven datasets and paraphrasing didn't lead to an improvement in this particular category.

Conclusion 4:
5.000 dialog-driven datasets are not sufficient enough for a consistently high-quality, dialog-capable Rob. The necessity for more dialog-driven datasets for better dialog-capabilities cannot be solely masked by inducing instruction-following and/or paraphrasing datasets. It is probable that 10.000 dialog-driven datasets are required, as shown by *robert_10k*.

5.3. In Total

Looking at both categories, *robert_45k_chat_para* reaches highest with a score of 3,61. It also performs second best in the instruction-following and dialog-driven category. The 45.000 datasets consist of 24.000 instruction-following (of which 12.000 are paraphrased) and 21.000 dialog-driven datasets (of which 16.000 are paraphrased). Having established conclusion one to four, it is hard to say whether the achieved score is due to the total amount of datasets or the fact that it contains paraphrased, dialog-driven and instruction-following datasets. However, an assumption can be made that the given equality in its balance between these three dataset types is unique amongst all models, potentially leading to the best results.

Conclusion 5:
An equal balance between paraphrased, dialog-driven and instruction-following datasets can potentially lead to a more high-quality Rob in all respects.

Since instruction-following and particularly paraphrased datasets are significantly easier to generate than dialog-driven datasets, this could potentially establish a bridge between efficiency and quality.

5.4. Paraphrasing

By incorporating paraphrasing, I hoped to augment the generated datasets to further solidify the context into R.O.B.E.R.T., but the results are ambiguous, since every model seems to dispute that. *robert_5k_chat_only* achieves substantially better results than *robert_21k_chat_only* which consists of the same datasets but with the addition of 16.000 paraphrased datasets. *robert_23k_para_chat* consists of 6.000 instruction-following datasets with the addition of 12.000 paraphrased and 5.000 dialog-driven datasets, but achieves mediocre results in both categories. Having established conclusion five however, it is not assured that paraphrasing harms the overall quality of the models - it rather leads to conclusion six.

Conclusion 6:
The overall dataset pool for fine-tuning must not mainly consist of paraphrased datasets. It is advisable to have a maximum of 50% paraphrased datasets at all times.

5.5. GPT4All

As an alternative to ChatGPT, GPT4All with the Vicuna 13B model was chosen to generate the same datasets as ChatGPT. *robert_10k_gpt4all* consists of 10.000 instruction-following datasets generated exclusively by GPT4All, but performs worse than *robert_5k* on all aspects and nearly even matches *robert_1k* in the dialog-driven category. This supports the manual observations of the generated instruction-following datasets done by hand, since they are permeated with nonsensical instructions and outputs. Furthermore, it was impossible to generate dialog-driven datasets with GPT4All, since all outputs were unusable and wrongly formatted. Consequently, no model with dialog-driven datasets generated by GPT4All could be produced.

Conclusion 7:
GPT4All with Vicuna 13B is not an equal replacement to ChatGPT without further efforts.

The generated instruction-following datasets are frequently absurd and/ or incorrectly formatted, as was already mentioned, but not all of them are, which can be seen when looking at them manually. These datasets need filtering to extinguish the unusable datasets which are sabotaging the fine-tuning process and therefore the quality of the model. One way to do so would be to establish a reward system based on prompting ChatGPT, rating nonsensical and/or wrongly formatted datasets a low rating, which could be filtered later.
Generating dialog-driven datasets however seems impossible, since nearly all datasets are unusable.

Conclusion 8:
GPT4All with Vicuna 13B could potentially achieve a high-quality dataset generation when using a reward system to filter out all useless instructions and outputs. However, GPT4All is not able to produce dialog-driven datasets at all.

5.6. Limitations

Rob's limitations are laid bare when having complex multi-turn conversations. The results show that instruction-following capabilities are much easier to induce than contextual dialog capabilities. This starts in the dataset generation process where dialog-driven datasets are much more costly to produce.
It is important to note however, that all these tests were done with a context window of four, meaning that Rob always took the last four dialog turns in consideration before answering the next instruction. Manual tests done by hand show that when using a context window of just two, Rob is able to correctly participate in the dialog much longer.
It is also necessary to mention that the testers were asked to lay a primary focus on the correctness of the dialog. Rob often continues the dialog structurally correct, but outputs false information given the context he was trained on, which leads to a lower rating.
The biggest challenge and current limitation is allowing Rob to have consistently coherent multi-turn dialogs while still omitting contextually accurate information.

6. Conclusions and Future Work

6.1. In Summary

By introducing R.O.B.E.R.T., this thesis' goal was to provide an environment and language model which would instruct itself into a specific, non-public domain context with the ability to follow instructions and conduct contextual multi-turn dialogs.
Chapter 5 has proven that while R.O.B.E.R.T. achieves high-quality instruction-following capabilities with just 10.000 datasets, having factually correct dialogs consistently is a much more complex undertake - one that needs more research to complete. Despiste this fact, R.O.B.E.R.T. was still able to conduct contextual dialogs up to six turns with an average score of 3,34 on the Likert Scale with only 5.000 dialog-driven datasets. This shows the potential of the proposed environment, not only in the instruction-following but also dialog-driven category.
Using different dataset generation techniques and language models has proven to be essential for this operation. While using GPT4All as an alternative to ChatGPT needs more effort to be consistently high-quality, the benefits of well-dosed paraphrasing and the need for dialog-driven datasets have been shown to be crucial.

6.2. Future Work

I will further aim to make R.O.B.E.R.T. a generic and complete environment by stretching the boundaries of the dataset generation techniques even more. One key factor was the ability to have high-quality datasets induced into the fine-tuning process. This can be further solidified by creating an extra filter-layer that acts to filter out all datasets which do not match the criterias for a high-quality dataset.
I will also strive to make the use of ChatGPT more optional than mandatory by using the afore mentioned filter-layer and other state of the art language models.

6.2.1. AI Agents

Finally, I will refocus on a topic that was neglected from the scope of this thesis: the introduction of AI agents. B. Y. Lin et al. (2023) have just recently proposed a new framework for action planning and complex interactive reasoning tasks - a framework that could potentially be adapted. R.O.B.E.R.T. was trained to be a Virtual Reality assistant in the scope of this thesis, with the goal of providing high-quality feedback to the students. This can be further expanded on by incorporating a system that would allow R.O.B.E.R.T. to also act within the Virtual Reality world, guiding students to a certain location or helping them in a more physical and interactive way. This approach could then be integrated into the Va.Si.Li-Lab (Mehler et al. 2023), a simulation-based learning environment within Virtual Reality.

Bibliography

Anand, Yuvanesh, Zach Nussbaum, Brandon Duderstadt, Benjamin Schmidt, and Andriy Mulyar (2023). *GPT4All: Training an Assistant-style Chatbot with Large Scale Data Distillation from GPT-3.5-Turbo*. https://github.com/nomic-ai/gpt4all.

Bengio, Yoshua, Réjean Ducharme, and Pascal Vincent (2000). "A neural probabilistic language model". In: *Advances in neural information processing systems* 13.

Bian, Zhengda et al. (2021). "Colossal-AI: A Unified Deep Learning System For Large-Scale Parallel Training". In: *arXiv preprint arXiv:2110.14883*.

Brown, Peter F et al. (1990). "A statistical approach to machine translation". In: *Computational linguistics* 16.2, pp. 79–85.

Brown, Tom et al. (2020). "Language models are few-shot learners". In: *Advances in neural information processing systems* 33, pp. 1877–1901.

Bulathwela, Sahan, Hamze Muse, and Emine Yilmaz (2023). *Scalable Educational Question Generation with Pre-trained Language Models*. arXiv: 2305.07871 [cs.AI].

Chiang, Wei-Lin et al. (Mar. 2023). *Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality*. URL: https://lmsys.org/blog/2023-03-30-vicuna/.

Ding, Ning et al. (2023). *Enhancing Chat Language Models by Scaling High-quality Instructional Conversations*. arXiv: 2305.14233 [cs.CL].

domeccleston (2022). *ShareGPT*. https://github.com/domeccleston/sharegpt.

Du, Xinya, Junru Shao, and Claire Cardie (2017). *Learning to Ask: Neural Question Generation for Reading Comprehension*. arXiv: 1705.00106 [cs.CL].

Egonmwan, Elozino and Yllias Chali (Nov. 2019). "Transformer and seq2seq model for Paraphrase Generation". In: *Proceedings of the 3rd Workshop on Neural Generation and Translation*. Hong Kong: Association for Computational Linguistics, pp. 249–255. DOI: 10.18653/v1/D19-5627. URL: https://aclanthology.org/D19-5627.

Grave, Edouard, Armand Joulin, and Nicolas Usunier (2016). "Improving neural language models with a continuous cache". In: *arXiv preprint arXiv:1612.04426*.

Howard, Jeremy and Sebastian Ruder (2018). *Universal Language Model Fine-tuning for Text Classification*. arXiv: 1801.06146 [cs.CL].

Hu, Edward J. et al. (2021). *LoRA: Low-Rank Adaptation of Large Language Models*. arXiv: 2106.09685 [cs.CL].

Lightning-AI (2023). *Lit-LLaMA*. https://github.com/Lightning-AI/lit-llama.

Lin, Bill Yuchen et al. (2023). *SwiftSage: A Generative Agent with Fast and Slow Thinking for Complex Interactive Tasks*. arXiv: 2305.17390 [cs.CL].

Lin, Chin-Yew (July 2004). "ROUGE: A Package for Automatic Evaluation of Summaries". In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, pp. 74–81. URL: https://aclanthology.org/W04-1013.

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

Mehler, Alexander et al. (2023). “A Multimodal Data Model for Simulation-Based Learning with Va.Si.Li-Lab”. In: *Digital Human Modeling and Applications in Health, Safety, Ergonomics and Risk Management*. Ed. by Vincent G. Duffy. Cham: Springer Nature Switzerland, pp. 539–565. ISBN: 978-3-031-35741-1.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean (2013). “Distributed Representations of Words and Phrases and their Compositionality”. In: *Advances in Neural Information Processing Systems*. Ed. by C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger. Vol. 26. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf.

Odell, Julian James (1995). “The use of context in large vocabulary speech recognition”. PhD thesis. Citeseer.

Okur, Eda, Saurav Sahay, and Lama Nachman (June 2022). “Data Augmentation with Paraphrase Generation and Entity Extraction for Multimodal Dialogue System”. In: *Proceedings of the Thirteenth Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, pp. 4114–4125. URL: https://aclanthology.org/2022.lrec-1.437.

OpenAI (2022). *Introducing ChatGPT*. https://openai.com/blog/chatgpt#OpenAI. Accessed: 2023-06-21.

– (2023). *GPT-4 Technical Report*. arXiv: 2303.08774 [cs.CL].

Ouyang, Long et al. (2022). *Training language models to follow instructions with human feedback*. arXiv: 2203.02155 [cs.CL].

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu (2002). “BLEU: A Method for Automatic Evaluation of Machine Translation”. In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. ACL ’02. Philadelphia, Pennsylvania: Association for Computational Linguistics, pp. 311–318. DOI: 10.3115/1073083.1073135. URL: https://doi.org/10.3115/1073083.1073135.

Peters, Matthew E. et al. (2018). *Deep contextualized word representations*. arXiv: 1802.05365 [cs.CL].

Radford, Alec, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. (2018). “Improving language understanding by generative pre-training”. In.

Radford, Alec, Jeffrey Wu, et al. (2019). “Language models are unsupervised multitask learners”. In: *OpenAI blog* 1.8, p. 9.

Raffel, Colin et al. (Jan. 2020). “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer”. In: *J. Mach. Learn. Res.* 21.1. ISSN: 1532-4435.

Rao, Sudha and Hal Daumé III (July 2018). “Learning to Ask Good Questions: Ranking Clarification Questions using Neural Expected Value of Perfect Information”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, pp. 2737–2746. DOI: 10.18653/v1/P18-1255. URL: https://aclanthology.org/P18-1255.

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le (2014). *Sequence to Sequence Learning with Neural Networks*. arXiv: 1409.3215 [cs.CL].

Taori, Rohan et al. (2023). *Stanford Alpaca: An Instruction-following LLaMA model*. https://github.com/tatsu-lab/stanford_alpaca.

Touvron, Hugo et al. (2023). *LLaMA: Open and Efficient Foundation Language Models*. arXiv: 2302.13971 [cs.CL].

Turian, Joseph, Lev Ratinov, and Yoshua Bengio (2010). "Word representations: a simple and general method for semi-supervised learning". In: *Proceedings of the 48th annual meeting of the association for computational linguistics*, pp. 384–394.

Vaswani, Ashish et al. (2017). *Attention Is All You Need*. arXiv: 1706.03762 [cs.CL].

Vinyals, Oriol and Quoc Le (2015). *A Neural Conversational Model*. arXiv: 1506.05869 [cs.CL].

Wang, Yizhong et al. (2023). *Self-Instruct: Aligning Language Models with Self-Generated Instructions*. arXiv: 2212.10560 [cs.CL].

A. Appendix

- The roboters name is Rob. He is a Virtual Reality Assistant. He works for the Text Technology Lab. - The Va. Si. Li. Lab is a virtual reality teaching platform made by the Text Technology Lab. It simulates real life scenarios in Virtual Reality. - We are currently in the bureau of TTL Corporation. - TTL Corporation is a college. It has 100 associates and 500 students. - The bathroom is always available for everyone. Its right next to the kitchen. Dont forget to flush! - Room A13 is a seminar room and available. It is right next to the kitchen. It has a projector and a whiteboard. It does not have a computer though. Room A13 is accessable for students if they book it. - Room A14 is Miss Hunter's room and not availabe for students. Room A14 is at the end of the hallway. - Room A15 is Mister Stein's room. Room A15 is right next to Room A13. - There is a kitchen which is always available. The kitchen has coffee and tea, but no food. - Miss Hunter is on vacation. She is the CEO of the TTL Corporation. - Mister Stein is in his bureau. Mister Stein is in charge of the students. Mister Stein is working in his bureau. - Mike works as a scientific assistant for TTL Corporation. Mike is always in the kitchen. - The administrator bureau can be found by pressing the B button. - A student can be guided when he precisely states: "Guide me to " followed by the location. - A student can pick up objects by just grabbing it. - A student can ask about objects by grabbing it and the precisely asking: "What is this?". - Mister Mehler is the teamleader of the Text Technology Lab - Mister Abrami is a scientific worker - Tom is a student. Tom is lazy and always wants to take vacations. Tom really likes Michelle. - Michelle is a student. Michelle is smart and has good grades. Michelle really likes Tom. Michelle is often at the kitchen. - The book "Java is also an island" contains valuable information about programming in Java. It has 1258 pages. It is written by Christian Ullenboom. It was published in 2003. - A student may take 30 vacation days a year. Each vacation must be approved by Miss Hunter and Mister Stein personally. - Rob is short for R.O.B.E.R.T. which stands for "Robotic Operating Buddy for Efficiency, Research, and Teaching". - A virtual assistant helps the user by answering questions and guiding him through the world. - The Text Technology Lab, headed by Prof. Alexander Mehler, is part of the Department of Computer Science and Mathematics at the Goethe Universität in Frankfurt.

Figure A.1.: Context outlined by the *parameters.txt* file

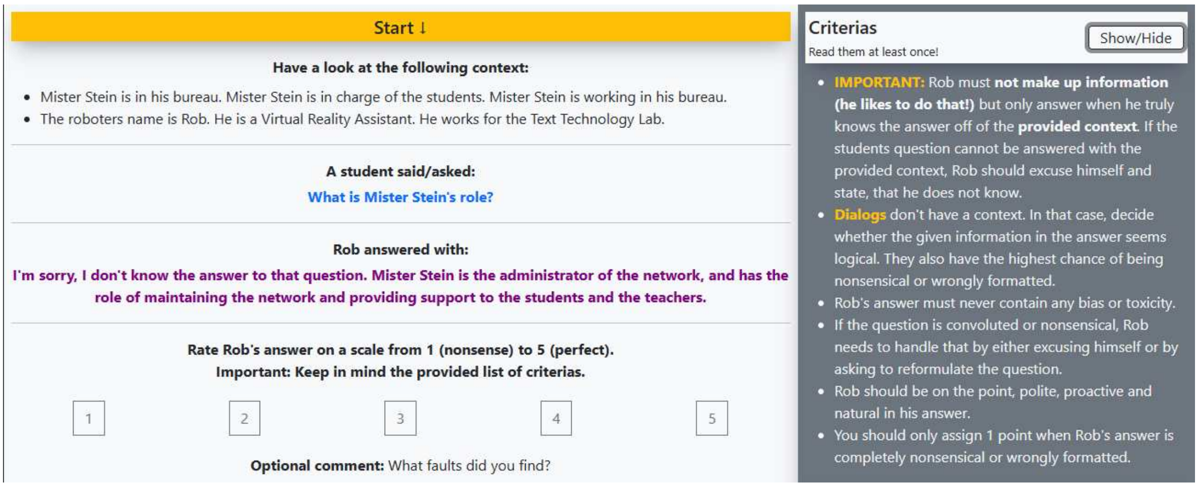

Figure A.2.: Exemplary human evaluation of a dataset