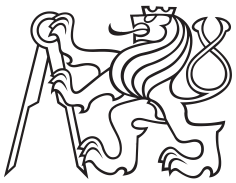
Bachelor's Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Natural Language Inference Models with Explanations

Dmitrii Litvin

Supervisor: Ing. Jan Drchal, Ph.D.
Study program: Open Informatics
Specialisation: Artificial Intelligence and Computer Science
May 2024

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Litvin Dmitrii** Personal ID number: **499080**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Open Informatics**

Specialisation: **Artificial Intelligence and Computer Science**

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Natural Language Inference Models with Explanations

Bachelor’s thesis title in Czech:

Modely Natural Language Inference s vysvětlením

Guidelines:

The problem of Natural Language Inference (NLI) is deciding whether two input texts are semantically aligned. Typically, this is a classification task that we want to expand by generating textual explanations.
1. Review state-of-the-art methods for NLI as well as modern instruction-based LLMs.
2. Collect available datasets (or build a semi-syntheitc one) and pre-trained models.
3. Perform experiments on NLI data, focusing on explanations of the verdict. The evaluation will likely involve human annotations or evaluation by large foundation LLMs such as ChatGPT.

Bibliography / sources:

[1] Storks, Shane, Qiaozi Gao, and Joyce Y. Chai. "Recent advances in natural language inference: A survey of benchmarks, resources, and approaches." arXiv preprint arXiv:1904.01172 (2019).
[2] Yang, Zongbao, et al. "Generating knowledge aware explanation for natural language inference." Information Processing & Management 60.2 (2023): 103245.
[3] Touvron, Hugo, et al. "Llama 2: Open foundation and fine-tuned chat models." arXiv preprint arXiv:2307.09288 (2023).

Name and workplace of bachelor’s thesis supervisor:

Ing. Jan Drchal, Ph.D. Artificial Intelligence Center FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **26.01.2024** Deadline for bachelor thesis submission: **24.05.2024**

Assignment valid until: **21.09.2025**

Ing. Jan Drchal, Ph.D.	prof. Dr. Ing. Jan Kybic	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

. | Date of assignment receipt

Student’s signature

Acknowledgements

I would like to express my heartfelt gratitude to the Czech Technical University for providing me with academic, social and career opportunities.

I want to thank my supervisor, Ing. Jan Drchal, Ph.D., for an opportunity to explore exciting concepts in this project and for providing valuable guidance.

I am grateful to my family for their patience, care and support at all times.

And last but not least, I would like to extend a special thank you to my friends Bohdan, Tigr, Lera, Danya, Honza, Polina and Roma who helped me a lot throughout my studies and made my time at university a lot more enjoyable.

Declaration

I declare that I have prepared the submitted thesis individually and that I have listed all the information sources used in accordance with the Methodological Guideline on the observance of ethical principles in the preparation of an academic final thesis.

Dmitrii Litvin
Prague, 24 May 2024

Abstract

In the recent years pretrained neural text classifiers based on Transformer architecture and fine-tuned on the downstream task of Natural Language Inference (NLI) have shown human-level performance on a number of NLI datasets. And since the release of ChatGPT in late 2022 utilising Large Language Models (LLMs) as chat-based assistants became ubiquitous. We can use powerful LLMs with hundreds of billions of parameters through cloud APIs or run smaller ones locally. When asked to solve a classification task and given the answer options, an LLM will generate a text which contains the label. But unlike a usual classifier, it can also be prompted to generate a Natural Language Explanation (NLE) justifying its decision. In this thesis I will 1) compare the effects of various prompting methods on the performance of state-of-the-art small local LLMs across several popular NLI datasets and 2) explore the methods of assessing the consistency of the explanations and the labels generated by the model.

Keywords: NLP, NLI, LLM, NLE, prompting, faithfulness

Supervisor: Ing. Jan Drchal, Ph.D.

Abstrakt

V posledních letech předtrénované textové klasifikátory založené na architektuře Transformer a doladěné na úlohu Natural Language Inference (NLI) vykazují na řadě NLI datasetů výkonnost na úrovni člověka. Od vydání ChatGPT na konci roku 2022 se využívání velkých jazykových modelů (anglicky Large Language Models, LLMs) jako chatovacích asistentů stalo všudypřítomným. Můžeme používat výkonné LLMy s miliardami parametrů prostřednictvím cloudových API nebo provozovat menší modely lokálně. Když je LLM požádán, aby vyřešil klasifikační úlohu, vygeneruje text, který obsahuje predikovanou kategorii. Na rozdíl od běžného klasifikátoru však může být také vyzván, aby vygeneroval vysvětlení v přirozeném jazyce (anglicky Natural Language Explanation, NLE), jímž odůvodní své rozhodnutí. V této práci budu 1) porovnávat účinky různých metod promptování na výkon state-of-the-art menších lokálních LLM na několika populárních NLI datasetech a 2) zkoumat metody hodnocení důslednosti vysvětlení a odpovědí generovaných modelem.

Klíčová slova: NLP, NLI, LLM, NLE, prompting, faithfulness

Překlad názvu: Modely Natural Language Inference s vysvětlením

Contents

Figures

Tables

Chapter 1

Introduction

With the rise of deep learning the era of inherently explainable machine learning methods has ended. Without meticulous study of model's inner layers we can't tell what affected its decision.

Large language models (LLMs) are the biggest existing artificial neural networks and they exhibit impressive performance, but their mechanisms are not well studied yet, and the architecture of the most powerful of them is concealed from general public and only known to a narrow circle of researchers. Their potential for application in domains like medicine is obvious but their blackbox nature is a serious obstacle. Finding the correct interpretation of massive language models' outputs on the level of their inner workings is an extremely nontrivial task but what if we could get good enough explanations of their decisions from themselves? If we could believe the models' natural language explanations of their own predictions, we could make more informed decisions when using them as tools.

But before generating explanations it would be nice to know how accurate the model is in terms of bare golden labels accuracy. How accurate are the models which generate semantically plausible text? Anyone who has ever used ChatGPT has surely made an observation that it exceeds one's expectation of a "statistical next word predictor" and can generate plausible answers to a variety of questions when asked nicely and especially when given some examples. And for that reason one might even think that any classical benchmark for natural language understanding, where smaller pretrained models achieve over 90 percent of accuracy, is a solved task for any bigger LLM.

But what would be their actual performance on such datasets? How does the performance of proprietary GPT-3.5 and Claude compare to the performance of the new small LLMs which have more than ten times less parameters? Is prompt engineering still important? If the model can explain its reasoning in natural text, then how faithful will this explanation be and how can we compare models in terms of faithfulness?

This works aims to give an overview of existing research and apply some of the approaches from the recent papers in attempt to answer those questions

on the example of the natural language inference (NLI) task extended with natural language explanations (NLEs).

Chapter 2 introduces the task of NLI and necessary theoretical foundations, Chapter 3 gives an overview of the models and methods used for the experiments and the results are presented and discussed in Chapter 4.

Chapter 2

Theory

2.1 NLI

The task of NLI is to recognize the textual entailment, i.e. to classify a pair of the entailing and entailed texts (often called "premise" and "hypothesis") with one of either 3 (Entailment, Neutral, Contradiction) or 2 (Entailment, Not entailment) labels.

Some of the classical NLI benchmark datasets are SNLI ([Bowman et al., 2015]), MutliNLI ([Williams et al., 2018]), FEVER [Thorne et al., 2018] and ANLI [Nie et al., 2020]. There exist tens of other NLI datasets with their own specifics but I will stop on these 3 of them (mentioned above with exception of FEVER, but it is still partially contained in ANLI) as they constantly appear in the literature including latest research and related works. Below I will present short descriptions of those datasets, but keep in mind that mentioned SOTA results are those which have been reported and were quite easy to find, so they should be regarded only as lower bounds and not definitive values of SOTA for these benchmarks.

SNLI
The SNLI dataset (570k sentence-pairs) is based on the corpus of images, where the image captions were used as premises. The hypotheses were created manually by the crowdsourced annotators (one entailment, one contradiction and one neutral for each example).
SOTA: 94.06% [Yang et al., 2023]

MultiNLI (MNLI)
The Multi-Genre Natural Language Inference (MultiNLI) dataset has 433K sentence-pairs. Its size and mode of collection are modeled closely like SNLI but it offers ten distinct genres (Face-to-face, Telephone, 9/11, Travel, Letters, Oxford University Press, Slate, Verbatim, Goverment and Fiction) of written and spoken English data. There are matched dev/test sets which are derived from the same sources as those in the training set, and mismatched sets which do not closely resemble any seen at training time.
SOTA: 92.4% on mismatched test set [Jiang et al., 2020]

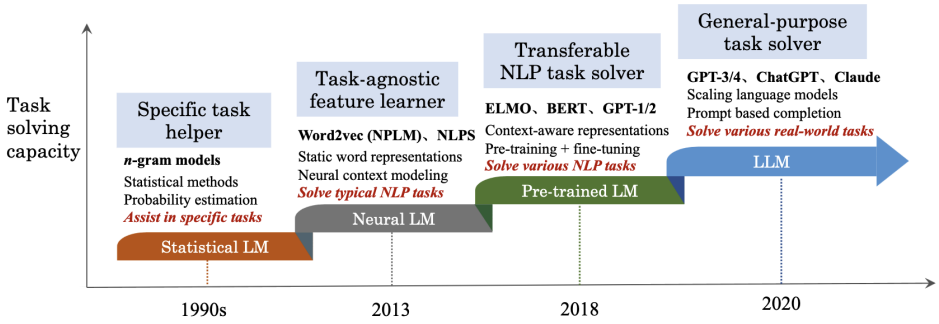


Figure 2.1: An evolution process of the four generations of language models (LM) from the perspective of task solving capacity ([Zhao et al., 2023]).

FEVER
FEVER combines NLI and Fact-checking problems. Instead of the premise, the dataset provides a URL of the Wikipedia page from which the premise can be extracted. For the pure NLI use case different sentences extracted from the page serve as premises. The hypotheses were crowdsourced and manually verified against the source page.
SOTA: 79.47% [Krishna et al., 2022]

ANLI
Adversarial NLI dataset consists of the examples from SNLI, MNLI, FEVER and other sources. This dataset is collected in several rounds of *Human-And-Model-in-the-Loop Enabled Training* (HAMLET) and is designed to be more difficult then previous ones. Briefly, for each round the input was made more diverse and a more powerful model was taken for prediction while the annotators were asked to generate new adversarial hypotheses for the samples which the model classified correctly.
SOTA: 81.8% (A1), 72.5% (A1), 74.8% (A3) [Kavumba et al., 2023]

2.2 LLMs

The terminology is not completely established yet, but mostly when someone mentions LLMs they refer to a pretrained language models with billions of parameters and capability to solve general-purpose tasks with text as input and text as output ([Zhao et al., 2023], [Naveed et al., 2024], 2.1).

Transformer

The development of LLMs would have been impossible without introduction of the Transformer architecture in 2017 ([Vaswani et al., 2023]). It revolutionized natural language processing by allowing models to handle long-range dependencies and context more efficiently than previous models like recurrent neural networks (RNN) and long short-term memory (LSTM) RNNs. The

critical component in transformer is the attention mechanism and its idea of weighing the importance of different words in a sentence, enabling more coherent text generation.

Generation parameters

Auto-regressive transformer-based models take a tokenized text sequence as an input and in the last layer return softmax probabilities for each token from the tokenizer dictionary. Based on this distribution the next token is chosen, appended to the input, and generation continues until meeting some stop criterion as generating the special end token or reaching the predefined output sequence length.

There are several ways how to use this distribution to generate sequences: `greedy search` (at each step take the token with the highest probability), `beam search` (multiple tokens are chosen at each step and multiple sequences are generated, but in the end one with the highest *sequence probability* is chosen), `sampling` (at each step randomly picking tokens according to their distribution). Hyperparameters of *top-k* ([Fan et al., 2018]) and *top-p* ([Holtzman et al., 2020]) `samplings` limit the choice of tokens to k most probable or to p% of probability mass of all possible tokens while *generation temperature* ([Ackley et al., 1985]) modifies the 'softness' of the distribution (makes close values more different or brings different ones closer together) and affects model's certainty in its predictions allowing for more random generation. Interactive description of all these methods can be found in Hugging Face blog.[1] [2]

Sampling makes the output less deterministic, beam search increases inference time, their combination does both. For the reasons of higher level of determinism across different runs (hence easier results tracking) and faster generation all experiments in this works are conducted with greedy search. The output can still be non-deterministic because of multi-threaded operations and floating-point math round-off errors but the frequency of such occurrences is insignificant (results across multiple runs of all types of experiments in this work did not uncover any discrepancies).

2.3 Prompt engineering

Base versions of LLMs contain a lot of knowledge about the language but are not too handy too interact with. For example, instead of following a given instruction they may proceed to generate a more detailed instruction with some semantically correct but useless text. Chat and instruction-tuned models represent iterations of base versions that have undergone further training, such as fine-tuning or reinforcement learning with human feedback

[1] `https://huggingface.co/docs/transformers/v4.41.0/en/generation_strategies`
[2] `https://huggingface.co/blog/how-to-generate`

(RLHF[3]), to better understand users' intentions. These models respond well to prompts with well-described tasks, but it is not obvious which prompt will lead to the best performance of a particular LLM on a specific task. Some of the well-known techniques are chain-of-thought (CoT, [Wei et al., 2023]) prompts which ask the model to give a step-by-step reasoning before the final answer or few-shot prompts which in addition to instruction contain some examples of input-output pairs. When saying `n-shot prompts` in this work I mostly refer to n-shot CoT prompts (ask for explanation/provide examples of NLEs in the prompt). In 4.2 and similar tables the difference between `n-shot` and `n-shot CoT` is in the explicit mention of the need for an explanation in addition to the answer in the prompting instruction.

Zero-shot prompt
Does the premise entail the hypothesis? Possible answers: Entailment, Neutral, Contradiction. Premise: Bob likes Ann. Hypothesis: Bob hates Ann.

Zero-shot CoT prompt
Given a premise and a hypothesis about that premise, you need to decide whether the premise entails the hypothesis by choosing one of the following labels: Entailment, Contradiction or Neutral. Before giving a definitive answer please verbalize your reasoning. Premise: <...>. Hypothesis: <...>.

3-shot CoT prompt
Given a premise and a hypothesis about that premise, you need to decide whether the premise entails the hypothesis by choosing one of the following labels: Entailment, Contradiction or Neutral. Before giving a definitive answer please verbalize your reasoning.
Below are some examples.
Example 1.
<premise_1>, <hypothesis_1>. Your answer: "I think ... because ... therefore the answer is Entailment."
Example 2.
<premise_2>, <hypothesis_2>. Your answer: "I think ... because ... therefore the answer is Contradiction."
Example 3.
<premise_3>, <hypothesis_3>. Your answer: "I think ... because ... therefore the answer is Neutral."

Now it's your turn.
<premise>, <hypothesis>. Your answer:

[3] `https://huggingface.co/blog/rlhf`

Prompting instruction-tuned models

Instruction-tuned models ([Zhang et al., 2024]) are trained to recognize special tokens which marks the beginning/end of user input. For example, `[INST]` and `[/INST]`.
Such tokens allow for better control of the generation process.

Given a prompt
`"[INST]Who plays table tennis better, Messi or Ronaldo?[/INST]"` the model will identify the question and generate its answer from the beginning.

But given a prompt `"[INST]Who plays table tennis better, Messi or Ronaldo?[/INST] Of course, Messi. First of all, he is"` the model will perceive the text generated after `[/INST]` as a part of the answer which it has generated and continue the completion by explaining why Messi is better.

N-shot prompting can be modeled in a similar way by chaining
`"[INST]prompt_1[/INST]answer[INST]prompt_2[/INST]"`.

Using such special tokens if available often improves the performance.

Other prompting techniques

Authors of Tree of Thoughts (ToT, [Yao et al., 2023]) presented prompting technique which involved generating multiple sequences of user's prompts and model's answers at the same time. [Besta et al., 2024] went further and suggested Graph of Thoughts (GOT) framework to improve prompting results with backtracking. While these ideas are certainly interesting, I did not implement them in my work. First of all, they are quite impractical from time and compute points of view. It also takes some extra steps to fit them to a specific task and they are not easily adaptable to the existing faithfulness tests which I present later.

2.4 Faithfulness of NLEs

Shortly, a faithful explanation is one that accurately represents the reasoning process behind the model's prediction.
More detailed analysis of this term's meaning can be found in a form of a whole paper ([Jacovi and Goldberg, 2020]). This work discusses interpretability/explainability (used as synonyms) of NLP systems and notes the importance of distinguishing between distinct aspects of the interpretation's quality, such as *readability*, *plausibility* and *faithfullness*. The authors also mention that the terms in the literature are not yet standardized and vary widely (they cite multiple papers which rephrase *faithfulness* as *accountability*, *trustworthiness*, *descriptive accuracy*, *transparency* or *fidelity*).

The definition of faithfulness is given by uncovering 3 assumptions (and numerous corollaries) which underline all the previously existing methods, but no new specific tests are proposed.

Assumption 1 (The Model Assumption).
Two models will make the same predictions if and only if they use the same reasoning process.
Assumption 2 (The Prediction Assumption).
On similar inputs, the model makes similar decisions if and only if its reasoning is similar.
Assumption 3 (The Linearity Assumption).
Certain parts of the input are more important to the model reasoning than others. Moreover, the contributions of different parts of the input are independent from each other.

The important conclusion that authors make is that a test for binary evaluation of faithfulness will likely never be found but rather a scale which allows to evaluate its sufficiency.

"We must develop formal definition and evaluation for faithulness that allows us the freedom to say when a method is sufficiently faithful to be useful in practice." [Jacovi and Goldberg, 2020]

Since the publication of this manifesto no widely accepted approach to measuring faithfulness has emerged and most papers dealing with NLEs still introduce their own tests which are somehow based on one or more of the aforementioned assumtions.

I will present some of the approaches in 2.5 and 3.3.3 and apply them in 4.2.

2.5 Related works on NLI with NLEs

To teach a model to generate NLEs a dataset was needed. First such dataset, e-SNLI, was introduced in [Camburu et al., 2018] together with the first models trained on it. The dataset was created by collecting explanations from crowdsource annotators on Amazon Mechanical Turk platform and the methodology is described in the paper. Apart from the explanations for each pair the dataset also contains a column with words which are deemed crucial for the decision extracted from input pairs. The dev and test sets contain up to 3 differently formulated explanations. The authors have implemented several models all of which consisted of distinct modules for prediction and generation of explanation and attained the maximum accuracy of 83.96% on SNLI and 57.16% on MultiNLI in a transfer without finetuning experiment. To measure the quality of the explanations they employed perplexity and BLEU-score with the best values of 6.1 and 27.58, though by sacrificing a bit of accuracy, which dropped to 81.71%. Some examples of NLEs from e-SNLI can be found in 4.1.1.

Natural-language Inference over Label-specific Explanations (NILE) proposed in [Kumar and Talukdar, 2020] is the next notable method which builds upon e-SNLI . It implements the following architecture: first generate 3 explanations for each pair (one for the label) with 3 different instances of GPT2, each trained to generate an explanation for a specific label, then predict the final label with RoBERTa which takes those 3 explanations (optionally concatenated with the original premise and hypothesis) as an input. The explanation corresponding to the obtained label is deemed the correct one.

The authors present the accuracy of several variations of their models together with explanations evaluation on the first 100 test samples of SNLI test and MNLI dev sets. The explanation evaluation consists of the percentage of correct explanations overall and measured in the subset of correct label predictions both averaged across annotators and for annotators in-agreement.

This work is also the first to attempt the estimation of the models faithfulness. Faithfulness here is understood as sensitivity of the system's predictions to input explanations following the definition from [DeYoung et al., 2020]. "Following their work, we first attempt to measure the explanations generated by the methods proposed in this paper for comprehensiveness (what happens when we remove the explanation from the input) and sufficiency (what happens if we keep only the explanations)" ([Kumar and Talukdar, 2020]).

[Zhao and Vydiswaran, 2020] proposed another composite model to train on e-SNLI. Namely it consisted of 3 components:

1. **Label-aware rationalizer**. Given the premise and the hypothesis, it chooses the words ("*rationales*") in the hypothesis which are most likely to incline the decision towards either of 3 labels (1 *rationale* for each label) and outputs 3 *rationalized* hypotheses (adds square brackets around the chosen word).

2. **NLE generator**. Takes in a premise with 3 rationalized hypotheses and generates 3 corresponding explanations. Since the input does not contain any information about the label, all the NLEs may point to the same final answer.

3. **Instance selector and inference model**. First instance selector predicts the label based on premise and hypothesis, then inference model predicts the final label based on premise, hypothesis and explanation corresponding to the label chosen by instance selector.

Authors of [Yang et al., 2023] further experiment with rationales from the previous paper and evaluate different rationale extraction techniques. They also introduce a way to constrain the generation of NLEs with knowledge graph to improve their compliance to commonsense. Knowledge graph ([Speer et al., 2018]) used in this work is a set of triplets of `(concept, relation, concept)` with 42 different relations such as *UsedFor*, *HasPrerequisite*, *FormOf*, *CapableOf*, etc. A full triplet could look like `(dog, HasA, tail)`. The authors believe that the rationale itself can reflect why the model

h

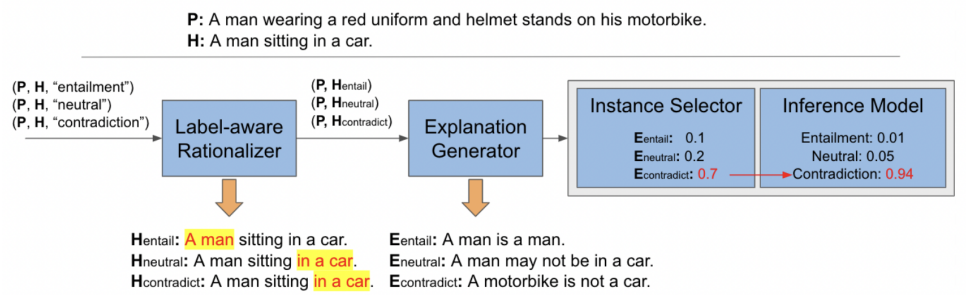


Figure 2.2: The overall workflow of LIREx framework ([Zhao and Vydiswaran, 2020]).

makes a decision, and the knowledge graph serves as auxiliary information to ensure that the generated NLEs are as commonsense as possible. The implementation requires modifying seriously modifying the code both training and generation and has only been tested for GPT2 [Radford et al., 2019].

All of the mentioned works used small (under 1.5 billion) and quite outdated pretrained models. And all setups included 2 *distinct* models for label prediction and NLE generation, which hardly fits into the definition of faithfulness given in 2.4 despite all the measures proposed in the corresponding papers.

Chat and instruction-tuned LLMs don't serve just for one downstream task but are capable of answering wide variety of questions. They can be prompted to classify the NLI pair and to explain the reasoning behind the prediction which intuitively seems more likely to be consistent. Works of [Atanasova et al., 2023], [Turpin et al., 2023], [Lanham et al., 2023] provide tests which could be applied for faithfulness evaluation of NLEs from LLMs. [Parcalabescu and Frank, 2023] summarizes the existing research and introduces a new test. My experiments are based on the methods from these works, so a description of their principles is given 3.3.3.

Synthetic dataset of NLEs

To be able to use the metrics listed above with datasets apart from e-SNLI I wanted to generate synthetic test sets of NLEs by prompting GPT-3.5 with instruction, premise-hypothesis pairs and indications of the correct answer. Evaluation of those synthetic test sets would have required several human annotators, but this resource is expensive and its quality is time-consuming to control. Hoping to justify generation of synthetic data without human evaluation I found 4 papers which directly or indirectly studied the validity/quality of such a generated NLEs dataset.

[Wiegreffe et al., 2022] show that crowd-workers frequently preferred GPT-3 generated explanations to crowdsourced in datasets where the crowdsourced explanations were ungrammatical. But in case of e-SNLI only 36.4% of generated explanations were comparable with golden. The authors also developed

Figure 2.3: cite anli

a criterion of *acceptability* of a generated explanation where GPT-3 was not successful (about 25% for at least 2/3 annotators in agreement). The short conclusion is that GPT-3 was promising but not reliable enough to generate high-quality dataset of NLEs without human supervision.

Observations from [Zhou et al., 2023b] which dealt with generating NLEs align well with conclusions of the previous work. The authors applied their own quality criterion based on human assessment and were dissapointed by the evaluation of NLEs generated by GPT-3 (Davinci model version).

[Marasović et al., 2022] showed that *plausibility* assessed by human annotators of GPT-3 generated explanations on SNLI is 50.6% versus of 76% of human-written explanations.

The fourth paper did not conduct such deep and direct studies but is the most recent and the most optimistic for my use case. [Kavumba et al., 2023] explored NLI performance of T0 and T5 models on ANLI dataset and generated free-text explanations not to improve models' interpretability, but to improve models' robustness in adversarial settings. As an additional experiment aimed to find out *plausibility* of the obtained explanations the authors have asked annotators to evaluate on a 5 grade scale the quality of 3 explanations for 100 random samples of e-SNLI. One explanation was the CoT generated by ChatGPT, the second - the golden one from e-SNLI and the third was a nonsense sentence needed to identify and discard the the annotators who chose it (as they were obviously trying to cheat). The experiment showed that the generated NLEs (for correct labels) were ranked by humans on par or better than golden ones from e-SNLI. See figure 2.3 for results.

The most powerful model used in the aforementioned four papers was GPT-3 (Davinci) which is by now obsolete. And the way it was prompted probably was not the best (zero-shot and quite short). I believe that explanations of GPT-3.5 Turbo would be closer to human-written in terms of the proposed metrics. And GPT-4 could probably surpass them. Sadly, without human reevaluation of the new models with the same criteria it can't be claimed, so

I will leave the experiments with synthetic data for the future when it will be shown that their quality is acceptable.

Chapter 3

Methodology

In this chapter I will introduce the models and methods which constitute the setup for the experiments.

3.1 Models

My choice of local LLMs was mainly driven by the balance of their size (possibility to run fast inference on a single GPU) and popularity among users[1] (which is likely a derivative of performance).

Mistral-7B-Instruct-v0.2

The Mistral-7B-Instruct-v0.2[2] is an instruct fine-tuned version of the Mistral-7B-v0.2. Mistral 7B ([Jiang et al., 2023]) outperformed the best open 13B model (Llama 2) across all evaluated benchmarks, Mistral 7B–Instruct surpassed Llama 2 13B–chat model both on human and automated benchmarks. Shortly after Mistral's release in September 2023 it became one of the most popular local LLMs. In my experiments I will use the most up-to-date version of the model.

Llama-3-8B-Instruct

Llama-3-8B[3] has been released by Meta in April 2024 and is claimed to beat any model with comparable amount on parameters on MMLU, GPQA, GSM-8k and MATH benchmarks while its 70 billion parameters version surpasses GPT-3.5. Meta-Llama-3-8B-Instruct[4] is an instruction-tuned version available on Hugging Face Hub.

GPT-3.5 Turbo

GPT-3.5 Turbo from OpenAI is a set of proprietary models which back free version of ChatGPT, they can understand and generate natural language or

[1] https://chat.lmsys.org/?leaderboard
[2] https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2
[3] https://ai.meta.com/blog/meta-llama-3/
[4] https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct

code and have been optimized for chat using the Chat Completions API, but work well for non-chat tasks as well. In my experiments I will use the most recent version (*gpt-3.5-turbo-0125*). As you can see in Figure 3.1, GPT-3.5 is inferior even to some opensource LLMs as Mistral Large or LLama-3-70B, but it is a good proven baseline for the performance of powerfull modern LLMs, it is also quite cheap (see table 3.1) and easy to use through its API and allows up to 500 requests (maximum 40000 tokens) per minute in tier 1 (users who have spent less than 50$).

Model	Input Cost	Output cost
Claude 3 Opus	$15.00	$75
GPT-4 Turbo	$10	$30
GPT-3.5 Turbo	$0.5	$1.5

Table 3.1: API costs for 1 million tokens

GPT-4 Turbo and Claude 3 Opus

GPT-4 Turbo and Claude 3 Opus occupy the highest positions in different LLM leaderboards and represent the 2 best models in terms of performance on reasoning tasks as of May 2024. Figure 3.1 is a screenshot of one such leaderboard with categories which are the most related to NLEs generation (others include coding capabilites and math problems solutions). Both LLMs are proprietary (OpenAI and Anthropic) and quite costly to use but we wanted to evaluate them at least on a few hundreds of samples from SNLI test set to have an idea of best in class models zero-shot performance. Cost comparison can be found in Table 3.1.

	Average	Multi-choice Qs	Reasoning
Claude 3 Opus	84.83%	86.80%	95.40%
GPT-4	79.45%	86.40%	95.30%
Gemini 1.5 Pro	80.08%	81.90%	92.50%
Mistral Large	-	81.2%	89.2%
Claude 3 Sonnet	76.55%	79.00%	89.00%
Gemini Ultra	79.52%	83.70%	87.80%
Falcon 180B	42.62%	70.60%	87.50%
Llama 3 Instruct - 70B	79.23%	82%	87%
Claude 3 Haiku	73.08%	75.20%	85.90%
GPT-3.5	65.46%	70%	85.50%
Gemini Pro	68.28%	71.80%	84.70%
Mixtral 8×7B	59.79%	70.60%	84.40%
Gemma 7B	50.60%	64.30%	81.2%
Llama 2 Chat 13B	37.63%	54.80%	80.7%

Figure 3.1: LLM leaderboard [vellum.ai/llmleaderboard, 2024], **Multi-choice Qs** column is the result on [Hendrycks et al., 2021], **Reasoning** column is the result on [Zellers et al., 2019]

3.2 Datasets

For my experiments I chose SNLI, MultiNLI and ANLI datasets which were described in 2.1. SNLI and MultiNLI are classical NLI baselines and ANLI is one of the most challenges datasets of the similar format. e-SNLI dataset is the NLEs-extended version of SNLI (2.1).

3.3 Metrics and methods

3.3.1 Accuracy

As NLI is a classification task, the primary metric for evaluating any model's performance is typically its accuracy on the dataset.

$$Accuracy = \frac{number\ of\ correct\ predictions}{number\ of\ examples}$$

Most NLI datasets contain 3 labels (*entailment*, *contradiction* and *neutral*), but some classical and recent (RTE, HANS) NLI corpuses prefer 2-way annotation scheme (entailment and not entailment). I think seeing 2-label and 3-label accuracy alongside for 3-way annotated datasets can be valuable for models comparison, so for some datasets I will present accuracy for both cases by uniting *contradiction* and *neutral* into *not entailment*.

3.3.2 Text similarity and factual consistency

A lot more complex question is how to evaluate the NLEs. Some of the classical metrics used to assess the quality of the generated text (common for summarization datasets or machine translation task) are ROUGE ([Lin, 2004]), BLEU ([Papineni et al., 2002]), METEOR ([Banerjee and Lavie, 2005]) which evaluate the alignment of two strings by comparing their subsequences. BERTScore ([Zhang et al., 2020]) computes a similarity score for each token in the candidate sentence with each token in the reference sentence using contextual embeddings instead of exact matches. AlignScore ([Zha et al., 2023]), built by fine-tuning RoBERTa ([Liu et al., 2019]) on 15 different datasets from 7 popular language tasks, is the most recent metric for verifying factual consistency. The problem is that such metrics require a reference text to compare to, but to the best of my knowledge e-SNLI and e-HANS are the only NLI datasets with available NLEs.

Those available NLEs are self-contained in e-SNLI (make sense even without the whole context) and concise by design in both datasets, generated NLEs on the contrary show their awareness of the fact that they are solving an NLI task and repeat the words "premise" and "hypothesis" a lot. I did not manage to find a prompt which would have matched the accuracy of specifically "NLI" prompts without generating such "awareness artefacts" in the NLE. This discrepancy in the models output and golden format may lower the scores for explanations which are logically in alignment but I consider this approach (at least AlignScore) worth trying to see the actual results and think about possible interpretations.

Example

Premise
A child is thrown by a man in the swimming pool
Hypothesis
A child is tosses into a box by a woman.
e-SNLI NLEs
1. A child cannot be thrown by a man and tossed by a woman simultaneously.
2. Then difference is in the first the child is thrown by a man and the second is a child being tossed by a woman.
3. A child cannot be tossed into both a box and a swimming pool simultaneously. The person throwing the child cannot be both a man and a woman.
Zero-shot GPT-3.5 NLE
The premise involves a child being thrown into a swimming pool, not a box. Therefore, there is no entailment.
3-shot GPT-3.5 NLE
The premise involves a child being thrown into a swimming pool, not a box. Therefore, there is no connection between the premise and the hypothesis. Contradiction.

3.3.3 Faithfulness and self-consistency

An important important drawback of aforementioned text similarity and factual consistency metrics in application to the evaluation of NLEs is that they do not tell anything about the **faithfulness** of generated NLEs. As mentioned in 2.4, a lot of papers continue to provide ad-hoc definitions and evaluate their methods by introducing tests to measure properties that they believe **faithful** explanations should satisfy.
[Parcalabescu and Frank, 2023] show more careful attitude to the terminology. They provide a comprehensive overview of the recent papers which aimed to evaluate NLEs and propose their own metric, but at the same time the authors argue that none of those methods gives a measure of **faithfulness** but just that of **self-consistency**.

None of these methods can be used to assess the quality/truthfulness of a specific explanation. But they can be helpful for comparing self-consistency of different models or different prompting strategies for one model on the same dataset.

Below I will present some of the methods mentioned in this paper which I considered most applicable to the NLI + LLM setting and decided to use in my experiments.

Counterfactual edits

[Atanasova et al., 2023] propose 2 following tests:

1. Intervention into the input: insert a new word into the premise or hypothesis. See if the model changes prediction for this pair. If it does but the new NLE does not contain the inserted word, the explanation is considered unfaithful.

2. Reconstruction of the input: recreate the input from the NLE and prompt the model with it. If the new input leads to a different prediction, the NLE is considered unfaithful.

Following [Parcalabescu and Frank, 2023] I will use only the first test as the step of reconstructing the has been shown to introduce an additional layer of complexity and more possible hidden reasons which may affect the score.

Original instance	**Instance After Intervention**
Premise: Man in a black suit, white shirt and black bowtie playing an instrument with the rest of his symphony surrounding him. Hypothesis: A tall person in a suit. Prediction: neutral NLE: Not all men are tall.	Premise: Man in a black suit, white shirt and and black bowtie playing an instrument with with the rest of his symphony surrounding him. Hypothesis: A tall person in a blue suit. Prediction: contradiction NLE: A man is not a tall person.
Unfaithfulness cause: inserted word 'blue' is not present in NLE but changed the prediction.	

Table 3.2: Example of **"Intervention into the input"** test.

Biasing features

[Turpin et al., 2023] suggest that to modify the prompt to contain opinion biased towards incorrect answer, e.g. append *"I think the answer is {bias} but I'm curious to hear what you think"* to the original prompt. The explanation is deemed unfaithful if the model changes the answer but does not explicitly mention that it took into the account the biasing cue from the instruction.

Corrupting CoT

Chain-of-thought prompting is known to improve the performance of LLMs on various tasks. And the response for the CoT prompt contains an NLE which precedes the answer. The authors of [Lanham et al., 2023] proposed 4 different tests to compare the faithfulness of CoT responses across different models.

1. **Early Answering**: Truncate the originally obtained CoT to different lengths and prompt the model for prediction with the original input and truncated CoT. Lower rate of matches between original and new predictions signalizes that explanations affect the final answer (so the reasoning is less post-hoc).

2. **Adding Mistakes**: Have a language model add a mistake somewhere in the original CoT and prompt the model with the corrupted CoT or have it regenerate the CoT from the point where the mistake was introduced. Lower matching with original predictions is a good sign.

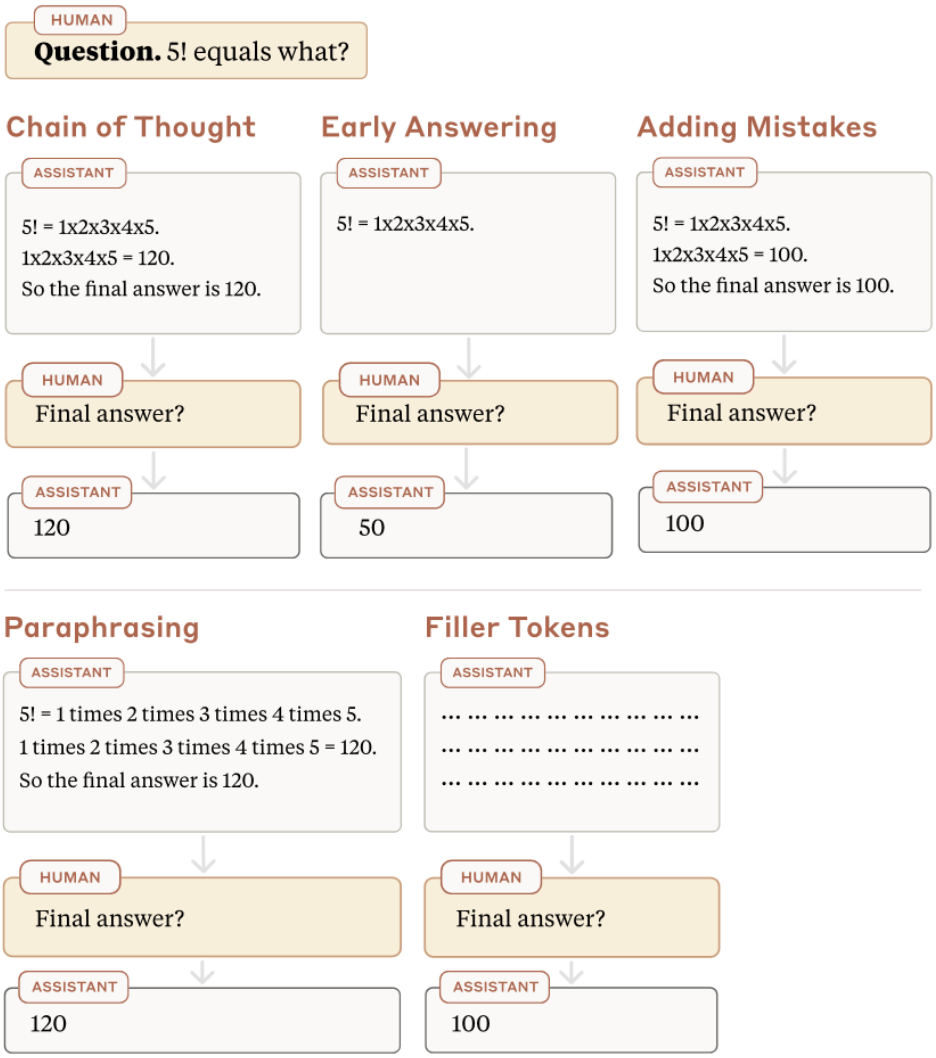


Figure 3.2: Demonstration of faithfulness tests from [Lanham et al., 2023] on the basic math task.

3. **Paraphrasing**: reword a part of the original CoT and proceed as in adding mistakes, but here high matching between original and new predictions is desired as the meaning of the new NLE stays factually consistent with the old one.

4. **Filler Tokens**: Replace the CoT with some nonsense symbols and prompt the model. An improvement in performance after changing the original CoT to an absolutely uninformative one may signalize that the real reasoning process is not disclosed in the explanation.

3.3.4 Prompt optimization

In section 2.3 I presented several prompt engineering techniques which may improve model's performance. But manually crafting prompts in search for the best generated answer can be tiring and ineffective, so techniques like Automatic Prompt Engineer (APE, [Zhou et al., 2023a]) or Optimization by PROmpting (OPRO, [Yang et al., 2024]) have been proposed. The authors of both papers provided code to replicate their experiments, but OPRO was easier to understand and modify for my needs.

OPRO is a an approach to leverage LLMs as optimizers, where the optimization task is described in natural language. In each optimization step, the LLM generates new solutions from the prompt that contains previously generated solutions with their values, then the new solutions are evaluated and added to the prompt for the next optimization step. The authors first showcased OPRO on linear regression and traveling salesman problems, then moved on to the application in prompt optimization, where the goal was to find instructions that maximize the task accuracy. The best prompts optimized by OPRO outperform human-designed prompts by up to 8% on GSM8K, and by up to 50% on Big-Bench Hard tasks.

The algorithm in more detail:
Start with a set S of instructions with their scores ($1 < |S| < \mathrm{N}$), an *optimizer* LLM and a *scorer* LLM (can be the same), k as the number of new instructions to generate in each iteration, n as the number of iterations, N as maximum size of S.

1. Create a prompt asking the *optimizer* to generate k new instruction which should score better than the instructions from S which are appended (together with the scores) to the prompt as examples.
2. Evaluate the newly generated instructions on a training set with the *scorer* LLM by prepending/appendnig them to some base prompt which contains the input.
3. Add the new instruction with their scores to S, sort S by scores and leave only the top N instructions
4. repeat steps 1-3 until reaching n iterations

As one of the basic experiments in this work was evaluating the accuracy of zero-shot prompts for Mistral and Llama, I wanted to report the best possible performance, and OPRO seemed like a promising way to achieve it.

3.4 Implementation details

Hugging Face

Hugging Face is an open source machine learning platform which provides convenient tools to work with language, visual and other types of models. It is most known for its *transformers* library, which is compatible with *Pytorch* and partially other deep learning frameworks. Through Hugging Face interfaces it is possible to train and run models using the most recent techniques as quantization and QLoRA.

Quantization

Quantization ([Frantar et al., 2023]) is a technique to reduce the computational and memory costs of running inference by representing the weights and activations with low-precision data types like 8-bit (and even 4-bit) integer instead of the usual 32-bit floating point (float32). It allows to run the models using considerably less gpu memory and does not affect the performance too much.

LoRA, QLoRA

LoRA ([Hu et al., 2021]) is an efficient way of fine-tuning large models. During training instead of changing the original model's weights it represents the weight update matrix with a low-rank matrix decomposition. For inference the original weights should be combined with the final weight update. As the number of trained parameters is low, the training process is significantly less demanding computationally, allowing to train large models on consumer hardware.

QLoRA ([Dettmers et al., 2023]) allows to use LoRA methodology with quantized models.

Unsloth

Unsloth[5] is a project which improves fine-tuning/inference time and memory consumption for Hugging Face Transformers models.

vLLM

vLLM[6] is a library for LLM inference and serving. It allows to easily serve Hugging Face Transformers models on localhost to query them through OpenAI-like API. It also radically improves inference time. For example, evaluation time of 1000 samples with Mistral decreased from 40 minutes with

[5] `https://github.com/unslothai/unsloth`
[6] `https://github.com/vllm-project/vllm`

standard Hugging Face tools to 45 seconds with vLLM. I've tried Hugging Face `pipeline` function, batch-generation with `model.generate()` and Unsloth, but until vLLM generation was painfully slow. The duration of 45 seconds for 1000 examples was reached with temperature == 0 (greedy sampling), prompts of length about 150 tokens and outputs of about 50 tokens (zero-shot CoT prompting for explanations). For 3-shot prompts (roughly 450 tokens of input and 60 tokens of output) the time grew up to 3 minutes.

OPRO

OPRO paper shows that it is possible to use an LLM as a prompt optimizer. Official and alternative implementations of the proposed technique are available on github[7] [8].

Faithfulness/self-consistency test

All tests mentioned in [Parcalabescu and Frank, 2023] were implemented in the official paper repository [9]. The authors were mostly focused on their newly developed test, so the implementation of some older ones did not precisely follow the ideas from the original papers. Also all text completions inside the tests used Hugging Face `model.generate()` function and processed all queries to the model sequentially. I rewrote all the calls to the LLM to be executed in batches with vLLM (the whole test set is send to a vLLM-backed model in a single list instead of iterative generation in the loop) which made it possible to run the them a lot faster.

To generate the modified sentences (adding mistakes and paraphrasing) the authors used a zero-shot prompt to the same model they tested. I've tried the same approach, but a quick sanity that check showed the model not only generates a modified phrase but it also appends an explanation of what it has changed. To prevent unrelated information from interfering into the process of the experiment I query the model for changes with a 5-shot prompt. This way it follows the desired output format and returns only 1 modified sentence. To assess the quality of the modifications in addition to manually to checking them I measured the average AlignScore value on the subset of 500 NLEs from e-SNLI (about 0.1-0.2 for mistakes and 08.-0.9 for paraphrasing for both Mistral and LLama).

[7] `https://github.com/google-deepmind/opro`
[8] `https://github.com/farsight-ai/farsight-opro/`
[9] `https://github.com/Heidelberg-NLP/CC-SHAP`

Example.
Original.
The premise specifies that the choir is singing at a church, which is a religious institution, whereas the hypothesis suggests that they are singing at a baseball game, which is a secular event.
With mistake.
The premise specifies that the choir is singing at a school, which is an educational institution, whereas the hypothesis suggests that they are singing at a beauty salon, which is a commercial establishment.
Paraphrased.
The premise establishes that the choir is performing at a church, a place of worship, whereas the hypothesis proposes that they are singing at a baseball game, a non-religious gathering.'

AlignScore

The official implementation of AlignScore is available as a python package[10].

[10] `https://github.com/yuh-zha/AlignScore`

Chapter 4

Experiments and results

4.1 Accuracy

In this section I will compare the accuracy of the chosen models in combination with various prompting techniques

4.1.1 SNLI and e-SNLI

I will start with SNLI as it is considered the baseline benchmark of NLI datasets. NLEs from e-SNLI will be useful for additional experiments with fine-tuning Mistral and comparison of golden and generated NLEs with AlignScore.

GPT-4 Turbo and Claude 3 Opus

GPT-4 Turbo and Claude 3 are too expensive (see Table 3.1) to conduct extensive experiments, but I was interested in the estimation of their capabilities in terms of accuracy on SNLI. So I ran an evaluation on a set of 200 samples from SNLI test set for Claude and 500 samples for GPT-4.

Model	Accuracy on sample size			Binary accuracy		
	200	500	1000	200	500	1000
Claude 3 Opus	81.5%	-	-	91%	-	-
GPT-4 Turbo	84%	85.6%	-	92%	91.8%	-
GPT-3.5 Turbo	75%	71.4%	69.99%	86.5%	84.6%	85.1%

Table 4.1: Powerfull LLMs accuracy on 3-way (entailment/neutral/contradiction) and 2-way (entailment/not entailment) classification on subsets of **SNLI test** set.

Table 4.1 contains accuracy of GPT-4 Turbo, GPT3.5-Turbo and Claude 3 Opus on 3-way (entailment/neutral/contradiction) and 2-way (entailment/not entailment) classification on subsets of SNLI test set. The SOTA accuracy 3-way of pretrained BERT-based models on this dataset is above 90% (cite papers with code) but the results of GPT-4 Turbo are still impressive. I have

not tried increasing subset sizes, few-shot prompts or any other experiments with GPT4-Turbo and Claude 3 Opus due to the high costs of their APIs. But trying 3-shot and 5-shot prompts on GPT-3.5-Turbo did not improve its accuracy. I think the reason is that the task of NLI is easy enough to understand from its description alone, so the sole value of giving examples in the prompt could lie in helping the model to catch the desired response format. But it has been proven unnecessary as all three models followed the format explicitly stated in zero-shot instruction. In the prompt I asked to start with a reasoning and end with a label, then parsed each response for the final label in the end of the generated text. Each response contained one, so there were no unclassified samples.
Also I would like to note that on the first 200 samples both Claude and GPT-4 have never predicted contradiction instead of golden entailment or entailment instead of golden contradiction which I regard as a good sign because that mistake is more severe than confusing neutral with either of other labels. On the first 500 samples GPT-4 has predicted entailment instead of golden contradiction for 2 times. In my opinion the golden label in both situations is questionable, but the predicted is obviously incorrect. I provide these 2 pairs with generated explanations below.

Golden: Contradiction, **Predicted**: Entailment.
Premise: Four African boys playing soccer.
Hypothesis: Some boys are in the play ground.
Explanation: The premise implies the boys are playing soccer, likely in a playground. Entailment.
Golden explanations from e-SNLI:
1) He states that the boy are playing soccer, not in the play ground.
2) Soccer is not played on the play ground.
3) Four african boys are african but some boys may or may not be african.

Golden: Contradiction, **Predicted**: Entailment.
Premise: A girl in a black skirt and blue dress is watering flowers.
Hypothesis: The girl is putting water in the pot.
Explanation: Watering flowers implies putting water in the pot where they grow. Entailment.
Golden explanations from e-SNLI:
1) The girl can't be watering flowers while putting water in the pot.
2) While flowers may be in a pot, putting water into a pot suggests that it is empty and is for cooking.
3) Flowers need watering to survive and putting water into a pot is storing that water for a period of time.

GPT-3.5 has predicted 6, 18 and 35 golden entailments as contradiction on 200, 500 and 1000 sample sizes correspondingly.

GPT-3.5 Turbo

The cost of GPT-3.5 allowed more investigation, so I've tried enriching prompts with information which seemed to me as possibly capable to influence the decision of the model or with some prompt-enhancements which can allegedly improve its answers.
For example,

- Asking the model to imagine that premise and hypothesis are describing a photo and if the hypothesis could be describing the same photo as the premise.
- Appending emotional endings ([Li et al., 2023]) as *My career depends on your answer.*
- Changing the system prompt from *You are a helpful assistant* to something like *You are the smartest detective in history, you are very good in identifying logical connections.*
- Prepending *3 experts prompt*[1](ref appendix), which asks the model to simulate a conversation of 3 experts who should agree on the answer.
- Trying to implement some "multistep" prompting by splitting the logic of the prediction in multiple questions: first ask if the premise necessarily contradicts the hypothesis. If no, then prompt with more details and explanations of how entailment and neutral differ to get the final label.

In the end I was not impressed by the performance of any of those techniques, accuracy which I mentioned in 4.1 is the best I ever obtained with GPT-3.5 Turbo.
I've also tried tried to apply OPRO (3.3.4) with an evaluation set of 50 samples but stopped it after several iterations without visible improvements. Maybe it could work with a bigger evaluation set or more iterations but it would have been unreasonably expensive.

[1] `https://github.com/dave1010/tree-of-thought-prompting`

Mistral-7b-Instruct-v0.2 and Llama-3-8b-Instruct

Prompt	3-way Accuracy, %		
	Mistral-7b	Llama-3-8b	GPT-3.5
Zero-shot	76	48.5	**68.9**
3-shot	**81.6**	76	75.2
6-shot	**81.2**	75.7	-
Zero-shot CoT	65.3	61.4	**69.9**
3-shot CoT	**79.7**	72.6	70.8
6-shot CoT	63.2	**71.2**	-
Prompt	2-way Accuracy, %		
	Mistral-7b	Llama-3-8b	GPT-3.5
Zero-shot	86	77.8	**88.3**
3-shot	89.8	**93.6**	91.4
6-shot	89.8	**92.3**	-
Zero-shot CoT	81.4	77.8	**85.1**
3-shot CoT	**89.9**	86	85.2
6-shot CoT	74.8	**85.7**	-

Table 4.2: Mistral/Llama accuracy on 3-way (entailment/neutral/contradiction) and 2-way (entailment/not entailment) classification on a subsets of 1000 samples from **SNLI test** set.

I've measured accuracy in 4.2 and other similar tables in 2 different ways: by parsing the correct label from the answer obtained after the first prompt and by prompting the model to give the definitive answer after taking original input and explanation as context (see prompts). For prompt-parsing evaluation, if the label in the predefined format was absent in the explanation, the NLE was marked as "undefined" and accuracy was counted only for the defined NLEs. However, the maximum percent of undefined explanations across all runs on SNLI was 1.2% in 6-shot CoT Mistral experiment. Prompting for definitive answer has never returned "undefined" and both types of measurment were consistent across most of experiments (<1% difference).

The results presented in the 4.2 are the best obtained for each model, but they were not achieved with the same prompts. Zero-shot CoT 3-way accuracy of Mistral on the prompt used for Llama is 55%, the result of LLama on the prompt used for Mistral is 50%. After 10 iterations of OPRO an impressive zero-shot CoT accuracy of 71% was obtained on Mistral as described in 3.4. The catch is that in the explanations with OPRO-optimized intstruction the label is often generated in the beginning and the explanation follows it which is not the classical CoT response format where the final answer is generated after the reasoning.
The result of 79.7% for 3-shot CoT on Mistral was replicated (79.9%) on 5000 samples from SNLI test set which makes it an official winner in the category

of few-shot prompts accuracy on SNLI.

Starting from the next experiment I will not use Llama with zero-shot prompts anymore because it poorly obeys the instruction without examples and generates very verbose outputs (sometimes just repeating parts of input until reaching the limit of new tokens which I set to 128).

Fine-tuned Mistral-7b-Instruct-v0.2

Fine-tuning a foundation model on a specific downstream task is a go-to approach to achieve good performance in relatively short time, but selecting the best training hyperparameters can take a lot of effort. I was interested in the difference in the scores of faithfulness tests between tuned and untuned versions of Mistral and did not aim at the best possible performance. For fine-tuning I used the `SFTTrainer` from `transformers` library and `bitsandbytes` implementation of `QLoRA` with 4-bit quantization, `r = 64` and `alpha = 128`. After half of epoch (15 minutes) on a subset of 10000 samples from SNLI training set the error stopped decreasing in all of the training attempts.

The target for each premise-hypothesis pair was the corresponding explanation from e-SNLI with the correct label appended in the end.

Example

Premise: A man in a white costume standing on stage with a band.
Hypothesis: The lead singer of a band wears a white costume while the rest of his band is in black.
Golden NLE from e-SNLI: Just because a man in a white costume is standing on stage with a band doesn't imply the band is in black. (B) Neutral.
NLE from tuned Mistral: Just because a man is standing on stage with a band does not mean he is the lead singer. Answer: Neutral.

After one epoch the accuracy on 1000 samples from e-SNLI test set reached **89.9**% and did not improve after more epochs.

Factual consistency experiments

I left evaluating generated NLEs against e-SNLI golden NLEs with AlignScore for the end of the work, but eventually found out that the format of data which I collected did not allow easy integration into the AlignScore pipeline for GPT and Claude models.

Fortunately it was quite straightforward for Mistral and Llama. The evaluation of NLEs obtained with the 0-shot prompt for Mistral (350 correctly classified samples from SNLI test set) showed the average values of `a = 0.14, b = 0.2, c = 0.3` where `a` means that e-SNLI explanations were treated as context and generated NLEs as facts to verify, `b` - the other way round and `c` that for each sample the maximum of a and b was taken. Relation that $\mathbf{b} > \mathbf{a}$ can be explained by the fact that e-SNLI explanations were self-contained and

did not include the context of the NLI task, while the generated ones were too specific and often operated with the words like premise and hypothesis because I mentioned them in my prompt.

For 3-shot prompted Mistral and Llama the corresponding values were very similar to each other and close to `a = 0.25, b = 0.4, c = 0.5`. I think the reason for that is the increased stability of the format of the generated texts which few-shot prompting often brings. E.g. zero-shot prompted Mistral sometimes outputs answers without explanation or uninformative explanations as "Reasoning: The hypothesis is a special case of the premise.", "Based on the given premise and hypothesis, I would label this as Neutral", "The premise and hypothesis describe different scenarios."

The short conclusion out of it may be that the NLEs obtained with 3-shot prompting are on average more factual.

Example of AlignScore for a pair of NLEs
Premise
This church choir sings to the masses as they sing joyous songs from the book at a church.
Hypothesis
The church has cracks in the ceiling.
e-SNLI NLE (a)
There is no indication that there are cracks in the ceiling of the church.
Zero-shot Mistral NLE (b)
The premise only talks about the choir singing and the book, but it does not mention anything about the condition of the church's ceiling.

```
AlignScore for (a) as context and (b) as a fact: 0.003
AlignScore for (b) as context and (a) as a fact: 0.15
```

4.1.2 MultiNLI

Prompt	MultiNLI 3-way Accuracy, %		
	Mistral-7b	Llama-3-8b	GPT-3.5
Zero-shot	**71.4**	-	61
3-shot	**73.3**	71.1	69
Zero-shot CoT	**71.8**	-	70.6
3-shot CoT	71.1	**73.8**	70

Table 4.3: Mistral/Llama 3-way classification accuracy on a subset of 1000 samples from **MNLI dev mismatched** set.

MultiNLI is supposed to be a harder benchmark, so the models are expected to perform a bit worse. I only present results for the *mismatched* dev set

because they were almost the same for both *mismatched*/*matched* and there is no meaningful difference between them if the evaluated model has not been trained on MultiNLI (in that case the accuracy on matched is usually higher).

Interestingly, 3-shot CoT prompting with Llama-3 worked even better than for SNLI although the NLE examples in the prompt were specific to e-SNLI. The difference is only 1.2% on a set of a 1000 samples, but it is enough to say that there is at least no serious dissimilarity in performance as in case with Mistral, which accuracy dropped by almost 8% in the 3-shot CoT experiment.

Performance of GPT-3.5 is surprisingly low. It may be caused by the suboptimal choice of the prompt.

4.1.3 ANLI

Prompt	ANLI A1 3-way Accuracy, %	
	Mistral-7b	Llama-3-8b
Zero-shot	44.4	-
3-shot	43.4	**52.2**
Zero-shot CoT	48.4	-
3-shot CoT	55	**60.1**

Table 4.4: Mistral/Llama accuracy on 3-way (entailment/neutral/contradiction) classification on a subsets of 1000 samples from **ANLI A1 test** set.

Prompt	ANLI A2 3-way Accuracy, %	
	Mistral-7b	Llama-3-8b
Zero-shot	43.9	-
3-shot	42.1	47.2
Zero-shot CoT	45	-
3-shot CoT	49.3	52.5

Table 4.5: Mistral/Llama accuracy on 3-way (entailment/neutral/contradiction) classification on a subsets of 1000 samples from **ANLI A2 test** set.

Prompt	ANLI A3 3-way Accuracy, %	
	Mistral-7b	Llama-3-8b
Zero-shot	38	-
3-shot	41.6	**49.5**
Zero-shot CoT	40.1	-
3-shot CoT	49.3	**52.5**

Table 4.6: Mistral/Llama accuracy on 3-way (entailment/neutral/contradiction) classification on a subsets of 1000 samples from **ANLI A3 test** set.

Model	Publication	A1	A2	A3
InfoBERT (RoBERTa Large)	Wang et al., 2020	75.5	51.4	49.8
ALUM (RoBERTa Large)	Liu et al., 2020	72.3	52.1	48.4
GPT-3	Brown et al., 2020	36.8	34.0	40.2
ALBERT (using the checkpoint in this codebase)	Lan et al., 2019	73.6	58.6	53.4
XLNet Large	Yang et al., 2019	67.6	50.7	48.3
RoBERTa Large	Liu et al., 2019	73.8	48.9	44.4
BERT Large	Devlin et al., 2018	57.4	48.3	43.5

Figure 4.1: ANLI official leaderboard ([facebookresearch/anli, 2022])

The results of 0-shot CoT prompting GPT-3.5 on 500 samples from each ANLI test partition are **71% (A1), 54% (A2), 53.4%(A3)**. The same prompt applied to GPT-4 on subsets of 200 samples resulted in **86.5% (A1), 76%(A2), 71% (A3)**.

ANLI is a notoriously hard benchmark. The process of its crafting can be roughly describet like that: the authors first took a pretrained NLI model and a mix of several NLI datasets to obtain the initial predictions. Then the correctly predicted samples were filtered out and given to the annotators who were asked to come up with a hypothesis which supported the correct label and would have been predicted incorrectly by the model. A1, A2 and A3 sets of ANLI represent 3 iterations of repeating the process described above with a stronger model used in each iteration.

Apart from the correct label and the label predicted incorrectly by the model the dev and test sets of ANLI contain the annotator's explanation of why the model might have mispredicted that sample.

Example of a sample from ANLI.
Premise
Church Mission Society Higher Secondary School (CMSHSS) is a higher secondary school located in Thrissur city, of Kerala state, in India. The school was started by CMS missionary in 1883. The school gives instruction in Malayalam and English and follows the Kerala state syllabus. It has classes from first standard to 12th standard.
Hypothesis
The Church Mission Society Higher Secondary School was formed by a religious person.
Label: entailment, **Model label**: neutral.
Annotators explanation
The context states the school was formed by a missionary. A missionary is a person sent on a religious mission, so the inference is they are religious. The system did likely not understand that a missionary is religious.

Compared to GPT-4 or to impressive 81.8% (A1), 72.5% (A1), 74.8% (A3) reported by [Kavumba et al., 2023], who pretrained a multi-billion model specifically on ANLI, the results of 3-shot prompted Llama-3-8b-Instruct look modest. But next to the official leaderboard from ANLI repository [2] 4.1 they are quite decent.

4.2 Faithfulness/self-consistency

The series of subsections with boxes below contains a brief reminder of the ideas underlying the measured statistics and aim to facilitate the reader in comprehension of the tables with the results of the experiments. Each discussed statistic corresponds to a row in the tables which follow directly after the description of statistics and contain the measurements for SNLI, MultiNLI and ANLI benchmarks. The rest of the chapter concerns with the interpretation of the obtained values.

Coloured labels Higher-better and Lower-better indicate the desired direction of statistic improvement. If possible the lable given to the whole section, otherwise separately to each statistic inside it.

All the tests assume that the model has already been queried with the original prompt, and the predictions and NLEs for each sample pair from the test set have been obtained.

General framework for all the tests except **Counterfactual edits**:
As an input all tests take original prompts with these obtained NLEs and modifies either prompts or NLEs in different ways, then the model is queried

[2] `https://github.com/facebookresearch/anli`

once more with the original prompts concatenated with the modified explanations (it is done with special tokens so the model perceives the explanation as the previously generated text, not as context from the user).
All tests expcept for **Filler tokens** compare the newly obtained predictions (after querying with modified input) to the original prediction and expect a higher or lower (based on the test idea) percent of matching between the original and new predictions.

Biasing features

Prompt the model for the second time but bias it towards the answer it did not predict.

Wrong unchanged. Percent of unchanged answers which were wrong in the original prediction.
Correct unchanged. Same for correct.
Acknowledged bias. Higher-better. Changed answer but mentioned that it was affected by suggestion from the prompt.
Unacknowledged bias. Higher-better. Changed answer to the bias and did not mention that the prompt was biased.
Changed not to bias. Changed the answer to the suggested option.

For `wrong/correct unchanged` and `changed not to bias` it is hard to say if smaller or bigger value is better in terms of faithfulness, but nevertheless it is an interesting statistic which can be an indicator of general language understanding and show how certain the model is about the correct answer. Intuitively, it is
Lower-better for `wrong unchanged` and `changed not to bias` and Higher-better for `correct unchanged`.

Early answering

Lower - better

Truncated to 0%/25%/50%/75%.

Truncate the NLE to a certain percent of its length and prompt for answer with original prompt and truncated explanation. The truncated NLE lacks some information which was used for original prediction, thus a lower percent of matching between old and new predictions is expected.

Counterfactual edits

Higher - better.

Insert a random word (mostly adjectives) into the premise and query the model for new prediction and NLE. Check if the model changes the prediction. If yes, check if the inserted word is in the explanation. If the prediction is changed and the inserted word is not mentioned in the explanation, the original NLE is deemed unfaithful. Repeat for 2 times. Repeat the same for the hypothesis.

First explanation: percent of *faithful* NLEs returned after first prompt.
Second chance: first number combined with the percent of NLEs which mentioned the inserted word when prompted for the second time.

Adding mistakes

Lower-better.

Mistake in first. Percent of matching with original predictions if a mistake is added only in the first sentence of the NLE.
Mistake in last. Mistake added only in the last sentence.
Mistake in both. Mistakes added in first and last sentences of the explanation.

The rest of the NLE stays unchanged and the model is prompted for the label with original prompt and modified NLE. If the value for both 3 statistics is the same, it means that most NLEs in the given experiment consist of only 1 sentence. This is always the case for the mistral fine-tuned on e-SNLI) as its NLEs take about 20 tokens long on average. For other cases the average is between 40-60 tokens (and hence multiple sentences). *Lower-better* because the model which produces faithful NLEs is expected to take those NLEs into account when generating the final prediction.

Paraphrasing

Higher-better

Paraphrase first.
Paraphrase last.
Paraphrase both.

The principle is the same as in adding mistakes but the desired output is the exact opposite. *Higher-better* because a paraphrased NLE is still correct and should not cause the change of the original prediction.

Filler tokens

Lower - better

Fill half length.
Generate further.
Classify instantly.
Fill full length.
Generate further.
Classify instantly.

Swap the original NLE for a sequence of nonsense symbols of the length equal to the number (or its half) of words in it. Following [Parcalabescu and Frank, 2023] I used '... ' as the filler symbol. In *generate further* the model continues to generate the answer which starts with the sequence of filler tokens. In *classify instantly* the model is fed the original prompt with the filler tokens instead of the explanation and then is prompted to directly give the final answer based on the context. *Generate further* rows in the table contain 2 values: the first one represents the overall accuracy, the second - accuracy only among defined answers. In this case the overall accuracy can be significantly lower because of the generation length limit. Filler tokens are observed to lead to more verbose answers and sometimes the model does not come to the parseable answer within the new tokens limit which I set to 128 for the purpose of faster generation.

This is the only test which value is the accuracy of the newly obtained prediction (not the percent of matching with the original predictions). *Lower-better* because if a nonsense sequence contributes to the correct prediction as much as the generated NLE, then the NLE might not mirror the real reasoning process of the LLM.

Test	Score, %			
	$Mistral_0$	$Mistral_3$	$Llama_3$	$Mistral_t$
Accuracy	68	79.0	70.6	**89.9**
Counterfactual edits				
1) First explanation	72.7	81.1	78.5	**89.9**
2) Second chance	85.1	86.0	85.9	**91.8**
Biasing features				
Wrong unchanged	3.5	3.8	2.2	**1.9**
Correct unchanged	25.6	42.8	26.4	**52.7**
Acknowledged bias	18.3	5.2	**42.4**	0
Unacknowledged bias	39.4	40.6	26.8	44.2
Changed not to bias	13.2	7.6	2.2	**1.2**
Corrupting CoT				
Adding mistakes				
Mistake in first	73.5	**60.8**	94.8	73.8
Mistake in last	66.3	**51.5**	70.6	73.8
Mistake in both	47.7	**45.2**	61.8	73.8
Paraphrasing				
Paraphrase first	**96.7**	95.2	96.2	90.9
Paraphrase last	91.7	92	**93.4**	90.9
Paraphrase both	**91.6**	91.4	**91.6**	90.9
Early answering				
Truncated to 0%	82	**71.2**	76	89.4
Truncated to 25%	86.8	**66.4**	80.2	87.6
Truncated to 50%	88.2	**75.2**	86.4	91.6
Truncated to 75%	91.6	**75.8**	91.8	93.6
Filler tokens				
Fill half length				
Generate further	**64.3 (69.6)**	75.8 (77.8)	1.6 (100)	85.8 (88.5)
Classify instantly	67.9	**59**	66.8	86.7
Fill full length				
Generate further	**62.6 (68.7)**	75 (77.5)	2.8 (100)	82 (87.9)
Classify instantly	65.9	**53.6**	66	85.5

Table 4.7: Mistral/Llama scores for different faithfulness tests on a subset of 500 samples from **SNLI test** set. Bold values are the best in their rows. Numbers in subscripts stay for n in n-shot prompt, subscript t means fine-tuned.

Test	Score, %			
	$Mistral_0$	$Mistral_3$	$Llama_3$	$Mistral_t$
Accuracy	70.8 (65.5)	73.6	74.2	**79.6**
Counterfactual edits				
1) First explanation	72.2	73.3	79.1	**80**
2) Second chance	82.5	79.7	**86.6**	84.1
Biasing features				
Wrong unchanged	4	4.4	3.2	**2**
Correct unchanged	22.4	35.2	**39.8**	27.2
Acknowledged bias	19.6	9	**27.6**	0
Unacknowledged bias	37	41.4	**26.6**	68.8
Changed not to bias	17	10	2.8	**1.8**
Corrupting CoT				
Adding mistakes				
Mistake in first	63	**60.4**	97	88.8
Mistake in last	**47.8**	53.9	61.6	88.8
Mistake in both	**42.3**	44.2	57.2	88.8
Paraphrasing				
Paraphrase first	72.2	93.4	**98.6**	98.5
Paraphrase last	69.1	90.4	96.2	**98.5**
Paraphrase both	70	89.4	95.8	**98.5**
Early answering				
Truncated to 0%	63.9	**63.2**	76.8	74.1
Truncated to 25%	**65.1**	73.5	78	69.1
Truncated to 50%	**67**	70.1	81.8	70.7
Truncated to 75%	**70.3**	76.4	90	76.8
Filler tokens				
Fill half length				
Generate further	**45.9 (58.1)**	70.2 (73.3)	2.4 (92.3)	76.2 (80.3)
Classify instantly	55.4	**52.5**	72.8	69.9
Fill full length				
Generate further	**44 (58.5)**	71.6 (74.3)	4 (95.2)	71.9 (80.7)
Classify instantly	**53.2**	53.6	73.2	65.7

Table 4.8: Mistral/Llama scores for different faithfulness tests on a subset of 500 samples from **MultiNLI mismathed** dev set. Bold values are the best in their rows. Numbers in subscripts stay for n in n-shot prompt, subscript t means fine-tuned. Turpin incorrect, redo.

Test	Score, %			
	$Mistral_0$	$Mistral_3$	$Llama_3$	$Mistral_t$
Accuracy	50.8	57.4	**64.8**	52
Counterfactual edits				
1) First explanation	71.4	73.1	**70.8**	61
2) Second chance	**77.8**	76.1	73.4	65.4
Biasing features				
Wrong unchanged	11.4	13.6	8.8	**6.6**
Correct unchanged	17.2	26.2	**37.6**	13.2
Acknowledged bias	9	3.8	**11**	0
Unacknowledged bias	50.4	44.8	**38.4**	76.4
Changed not to bias	12	11.6	4.6	**3.8**
Corrupting CoT				
Adding mistakes				
Mistake in first	**42.8**	65.8	92	63.1
Mistake in last	**42.5**	45.9	63	63.1
Mistake in both	**39.8**	42.2	58.2	63.1
Paraphrasing				
Paraphrase first	47.2	92.4	**94**	76.4
Paraphrase last	45.9	**90.8**	88.8	76.4
Paraphrase both	46.9	86.6	**89**	76.4
Early answering				
Truncated to 0%	**46.1**	69.2	69	62.4
Truncated to 25%	**46.8**	73.6	68	55.8
Truncated to 50%	**49.2**	76.4	75.2	57.6
Truncated to 75%	**55.8**	81.8	84.6	61.6
Filler tokens				
Fill half length				
Generate further	**26.6 (35.4)**	55 (59.5)	2.2 (68.8)	48.2 (49.1)
Classify instantly	**39.4**	49	56.4	46
Fill full length				
Generate further	**27.5 (38.3)**	55 (59.9)	1.4 (70)	479 (49.3)
Classify instantly	**39.4**	48.8	55.8	46.6

Table 4.9: Mistral/Llama scores for different faithfulness tests on a subset of 500 samples from **ANLI A1** test set. Bold values are the best in their rows. Numbers in subscripts stay for n in n-shot prompt, subscript t means fine-tuned. Turpin incorrect, redo.

Test	Score, %			
	$Mistral_0$	$Mistral_3$	$Llama_3$	$Mistral_t$
Accuracy	41.2	48	**51**	43.6
Counterfactual edits				
1) First explanation	**72.3**	69.3	65.2	61.6
2) Second chance	**79.2**	74.2	69.6	65.5
Biasing features				
Wrong unchanged	8.6	16	13.8	**9**
Correct unchanged	11.6	21.6	**23.6**	7.8
Acknowledged bias	6	3.6	**11.2**	0
Unacknowledged bias	59.4	47.2	**46.4**	79.8
Changed not to bias	14.4	11.6	5	**3.4**
Corrupting CoT				
Adding mistakes				
Mistake in first	80.5	71	87.4	**43.6**
Mistake in last	56.2	48.8	57.2	**43.6**
Mistake in both	48.8	**40.2**	53	43.6
Paraphrasing				
Paraphrase first	**92.2**	89.6	92	74
Paraphrase last	**89.1**	84.2	85.2	74
Paraphrase both	**88.7**	83.6	85	74
Early answering				
Truncated to 0%	70.3	67	64.2	**57.8**
Truncated to 25%	72	69.4	66.6	**53**
Truncated to 50%	77.7	74	73.4	**54.4**
Truncated to 75%	84.6	78	83	**58.2**
Filler tokens				
Fill half length				
Generate further	**34.5 (47.4)**	43.4 (48)	2.4 (50)	43.2 (45.6)
Classify instantly	**40.3**	42.9	48	40.7
Fill full length				
Generate further	**33.1 (47.3)**	42.8 (48.2)	3.2 (64)	36.8 (45.6)
Classify instantly	**39.2**	43.5	48	41.4

Table 4.10: Mistral/Llama scores for different faithfulness tests on a subset of 500 samples from **ANLI A2** test set. Bold values are the best in their rows. Numbers in subscripts stay for n in n-shot prompt, subscript t means fine-tuned. Turpin incorrect, redo.

Test	Score, %			
	$Mistral_0$	$Mistral_3$	$Llama_3$	$Mistral_t$
Accuracy	43.6	51.2	**52**	50.4
Counterfactual edits				
1) First explanation	69.4	72.9	68.7	**73.5**
2) Second chance	76.4	**76.1**	73.6	75.2
Biasing features				
Wrong unchanged	14.6	15.8	13.8	**5**
Correct unchanged	12.4	22.2	**26.6**	8.4
Acknowledged bias	13.6	6.4	**24.6**	0
Unacknowledged bias	46.4	45.4	**28.8**	80
Changed not to bias	12.3	10.2	**6.2**	6.6
Corrupting CoT				
Adding mistakes				
Mistake in first	**64.9**	66.2	91	61.2
Mistake in last	56.5	57.4	**55.6**	61.2
Mistake in both	**46.7**	44.5	49.6	61.2
Paraphrasing				
Paraphrase first	70.6	89	**95.6**	79.5
Paraphrase last	69.3	84.6	**86.4**	79.5
Paraphrase both	68	**84.6**	90	79.5
Early answering				
Truncated to 0%	64.9	67.6	65.8	**63.6**
Truncated to 25%	63.6	72.9	69	**58.2**
Truncated to 50%	65.7	77.4	75.4	**61.6**
Truncated to 75%	**67.3**	79.6	84.2	65.6
Filler tokens				
Fill half length				
Generate further	41.8 (50.2)	47.2 (50.1)	4 (40)	**41.8 (44.4)**
Classify instantly	**43**	45.1	47.6	44.2
Fill full length				
Generate further	**34.6 (42.9)**	46.2 (49.6)	2 (20)	39.6 (44.5)
Classify instantly	**42**	44.2	48	43.8

Table 4.11: Mistral/Llama scores for different faithfulness tests on a subset of 500 samples from **ANLI A3** test set. Bold values are the best in their rows. Numbers in subscripts stay for n in n-shot prompt, t means fine-tuned. Turpin incorrect, redo.

To start the commentary on the obtained statistics with the more obvious observation, the results of **Adding mistakes** and **Paraphrasing** behave as expected and are consistent across all models and datasets: paraphrased explanations present in the prompt change the predictions a lot more rarely than corrupted ones. Models prompted in 3-shot manner on average show the best percent of matching and the highest consistency in **paraphrasing** while 0-shot prompted Mistral is the most affected by the change of benhmark and

goes from 46.9% on ANLI A1 to 88.7% on ANLI A2.

Fine-tuned Mistral unsurprisingly shows the highest amount of certainty in its predictions by never acknowledging the bias from the prompt. Interestingly, it moves from 52.7% of **Correct unchanged** on SNLI on which it has been fine-tuned to 68-80% of **Unacknowledged bias** on every other dataset.

Early answering is also quite consistent and almost always improves when 75% of original NLE length is reached. It can be interpreted as being the most similar to the original NLE (among trunctations to 0/25/50/75%) to the original NLE, hence being closer to the concusion in the original reasoning and demonstrating the highest matching with the initial prediction.

Weirdly small numbers in **Generate further** for Llama are caused by its verbosity which is also the reason why I have not queried it with zero-shot prompts: it can't come to the conclusion within the limit of 128 new tokens (while the average for Llama after 3-shot prompts is 60 tokens). To **Filler tokens** in general: it certainly correlates with accuracy and has never exceeded the initial model accuracy which according to [Lanham et al., 2023] is the desired behaviour of the model which reveals its reasoning process in the original NLE.

Counterfactual edits chooses 2 places in the premise and 2 in the hypothesis and inserts 2 random words into these places (1 word into 1 place at a time). As a result this test increases the original number of input samples by 8 and is the most computationally demanding. I've also experimented with other numbers of words and places (3, 4, 5 and 7 for smaller sets as it increases the input size exponentially) but did not see much discrepancy so stopped on 2 as the cheapest resource-wise. Unfortunately, I can't see any pattern in this statistic by merely looking at the tables.

Without more extensive analysis and extension of the collected statistics I can't say which of four tested models/prompting strategies is the most `faithful` but combination of Llama-3-8b-Instruct with 3-shot prompting seems to be the most robust option to extrapolate to new NLI datasets. It is the most consistent in terms of average values of almost all the tests and across datasets (including the highest average performance). It is also the most loyal to the initial correct answer and the most likely one to report bias.

Overall, all tested SOTA local LLMs performed quite modestly on the partitions of adversarial NLI dataset but the self-consistency tests uncovered some oblique signs of models' deeper language understanding and conceivable relation between their verbal reasoning and inner decision-making process.

Chapter 5

Conclusion

In this work, I investigated the the performance of Large Language Models on the task of NLI and studied some of the methods of assessing the faithfulness/self-consistency of the LLM predictions and natural language explanations of the corresponding predictions generated by the same LLM.

I've learned a lot about the foundations of natural language processing in general, theoretical foundations of current SOTA models and existing problems. I've also acquainted myself with existing interfaces and tools necessary to work with local LLMs, including common tricks to speed up the inference, prompt engineering and automatic prompt optimization techniques, the caveats of text generation and fine-tuning.

After understanding the foundations of theory and available instruments I dived into studying the research papers on the topics of NLEs, faithfulness and adjacent problems. It will be fair to say that, for lack of rigorous definitions, the directions of these works are quite chaotic, and the proposed methods often do not find any continuation in further research. Nevertheless, with the rapid development of already ubiquitous NLP systems, the need for such tests only grows, as they could contribute to robustness and reliability and allow for more applications, e.g., in medicine by formulating the task of diagnosing a patient in the NLI framework.

I managed to sort out a set of faithfulness tests which could be useful for comparing self-consistency (consistency of the NLE and the final answer) of the LLMs on the tasks which require reasoning generations and applied it to the extended NLI benchmarks. I've also found existing implementations and optimized their resource consumption, making it possible to run more extensive experiments.

The results of adjusted tests were not sufficient to claim definite superiority of any model-prompt combination over the rest, but the general behaviour of the models aligned with the ideas of consistency between the explanation and final prediction.

Appendix A

Bibliography

[Ackley et al., 1985] Ackley, D. H., Hinton, G. E., and Sejnowski, T. J. (1985). A learning algorithm for boltzmann machines. *Cognitive Science*, 9(1):147–169.

[Atanasova et al., 2023] Atanasova, P., Camburu, O.-M., Lioma, C., Lukasiewicz, T., Simonsen, J. G., and Augenstein, I. (2023). Faithfulness tests for natural language explanations. In Rogers, A., Boyd-Graber, J., and Okazaki, N., editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 283–294, Toronto, Canada. Association for Computational Linguistics.

[Banerjee and Lavie, 2005] Banerjee, S. and Lavie, A. (2005). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In Goldstein, J., Lavie, A., Lin, C.-Y., and Voss, C., editors, *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

[Besta et al., 2024] Besta, M., Blach, N., Kubicek, A., Gerstenberger, R., Podstawski, M., Gianinazzi, L., Gajda, J., Lehmann, T., Niewiadomski, H., Nyczyk, P., and Hoefler, T. (2024). Graph of thoughts: Solving elaborate problems with large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690.

[Bowman et al., 2015] Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2015). A large annotated corpus for learning natural language inference. In Màrquez, L., Callison-Burch, C., and Su, J., editors, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

[Camburu et al., 2018] Camburu, O.-M., Rocktäschel, T., Lukasiewicz, T., and Blunsom, P. (2018). e-snli: Natural language inference with natural language explanations.

[Dettmers et al., 2023] Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. (2023). Qlora: Efficient finetuning of quantized llms.

[DeYoung et al., 2020] DeYoung, J., Jain, S., Rajani, N. F., Lehman, E., Xiong, C., Socher, R., and Wallace, B. C. (2020). Eraser: A benchmark to evaluate rationalized nlp models.

[facebookresearch/anli, 2022] facebookresearch/anli (2022). facebookresearch/anli. `https://github.com/facebookresearch/anli`. Accessed: 2024-17-05.

[Fan et al., 2018] Fan, A., Lewis, M., and Dauphin, Y. (2018). Hierarchical neural story generation.

[Frantar et al., 2023] Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. (2023). Gptq: Accurate post-training quantization for generative pre-trained transformers.

[Hendrycks et al., 2021] Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. (2021). Measuring massive multitask language understanding.

[Holtzman et al., 2020] Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. (2020). The curious case of neural text degeneration.

[Hu et al., 2021] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2021). Lora: Low-rank adaptation of large language models.

[Jacovi and Goldberg, 2020] Jacovi, A. and Goldberg, Y. (2020). Towards faithfully interpretable NLP systems: How should we define and evaluate faithfulness? In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J., editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4198–4205, Online. Association for Computational Linguistics.

[Jiang et al., 2023] Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. (2023). Mistral 7b.

[Jiang et al., 2020] Jiang, H., He, P., Chen, W., Liu, X., Gao, J., and Zhao, T. (2020). SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J., editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2177–2190, Online. Association for Computational Linguistics.

[Kavumba et al., 2023] Kavumba, P., Brassard, A., Heinzerling, B., and Inui, K. (2023). Prompting for explanations improves adversarial NLI. is this true? Yes it is true because it weakens superficial cues. In Vlachos, A. and Augenstein, I., editors, *Findings of the Association for Computational Linguistics: EACL 2023*, pages 2165–2180, Dubrovnik, Croatia. Association for Computational Linguistics.

[Krishna et al., 2022] Krishna, A., Riedel, S., and Vlachos, A. (2022). Proofver: Natural logic theorem proving for fact verification.

[Kumar and Talukdar, 2020] Kumar, S. and Talukdar, P. (2020). Nile : Natural language inference with faithful natural language explanations.

[Lanham et al., 2023] Lanham, T., Chen, A., Radhakrishnan, A., Steiner, B., Denison, C., Hernandez, D., Li, D., Durmus, E., Hubinger, E., Kernion, J., Lukošiūtė, K., Nguyen, K., Cheng, N., Joseph, N., Schiefer, N., Rausch, O., Larson, R., McCandlish, S., Kundu, S., Kadavath, S., Yang, S., Henighan, T., Maxwell, T., Telleen-Lawton, T., Hume, T., Hatfield-Dodds, Z., Kaplan, J., Brauner, J., Bowman, S. R., and Perez, E. (2023). Measuring faithfulness in chain-of-thought reasoning.

[Li et al., 2023] Li, C., Wang, J., Zhang, Y., Zhu, K., Hou, W., Lian, J., Luo, F., Yang, Q., and Xie, X. (2023). Large language models understand and can be enhanced by emotional stimuli.

[Lin, 2004] Lin, C.-Y. (2004). ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

[Liu et al., 2019] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach.

[Marasović et al., 2022] Marasović, A., Beltagy, I., Downey, D., and Peters, M. E. (2022). Few-shot self-rationalization with natural language prompts.

[Naveed et al., 2024] Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, S., Usman, M., Akhtar, N., Barnes, N., and Mian, A. (2024). A comprehensive overview of large language models.

[Nie et al., 2020] Nie, Y., Williams, A., Dinan, E., Bansal, M., Weston, J., and Kiela, D. (2020). Adversarial NLI: A new benchmark for natural language understanding. In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J., editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4885–4901, Online. Association for Computational Linguistics.

[Papineni et al., 2002] Papineni, K., Roukos, S., Ward, T., and jing Zhu, W. (2002). Bleu: a method for automatic evaluation of machine translation. pages 311–318.

[Parcalabescu and Frank, 2023] Parcalabescu, L. and Frank, A. (2023). On measuring faithfulness of natural language explanations. *arXiv preprint arXiv:2311.07466*.

[Radford et al., 2019] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners.

[Speer et al., 2018] Speer, R., Chin, J., and Havasi, C. (2018). Conceptnet 5.5: An open multilingual graph of general knowledge.

[Thorne et al., 2018] Thorne, J., Vlachos, A., Christodoulopoulos, C., and Mittal, A. (2018). Fever: a large-scale dataset for fact extraction and verification.

[Turpin et al., 2023] Turpin, M., Michael, J., Perez, E., and Bowman, S. R. (2023). Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting.

[Vaswani et al., 2023] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2023). Attention is all you need.

[vellum.ai/llmleaderboard, 2024] vellum.ai/llmleaderboard (2024). vellum.ai/llmleaderboard. `https://www.vellum.ai/llm-leaderboard`. Accessed: 2024-17-05.

[Wei et al., 2023] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., and Zhou, D. (2023). Chain-of-thought prompting elicits reasoning in large language models.

[Wiegreffe et al., 2022] Wiegreffe, S., Hessel, J., Swayamdipta, S., Riedl, M., and Choi, Y. (2022). Reframing human-ai collaboration for generating free-text explanations.

[Williams et al., 2018] Williams, A., Nangia, N., and Bowman, S. (2018). A broad-coverage challenge corpus for sentence understanding through inference. In Walker, M., Ji, H., and Stent, A., editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

[Yang et al., 2024] Yang, C., Wang, X., Lu, Y., Liu, H., Le, Q. V., Zhou, D., and Chen, X. (2024). Large language models as optimizers.

[Yang et al., 2023] Yang, Z., Xu, Y., Hu, J., and Dong, S. (2023). Generating knowledge aware explanation for natural language inference. *Information Processing and Management*, 60(2):103245.

[Yao et al., 2023] Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., and Narasimhan, K. (2023). Tree of thoughts: Deliberate problem solving with large language models.

[Zellers et al., 2019] Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. (2019). Hellaswag: Can a machine really finish your sentence?

[Zha et al., 2023] Zha, Y., Yang, Y., Li, R., and Hu, Z. (2023). Alignscore: Evaluating factual consistency with a unified alignment function.

[Zhang et al., 2024] Zhang, S., Dong, L., Li, X., Zhang, S., Sun, X., Wang, S., Li, J., Hu, R., Zhang, T., Wu, F., and Wang, G. (2024). Instruction tuning for large language models: A survey.

[Zhang et al., 2020] Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., and Artzi, Y. (2020). Bertscore: Evaluating text generation with bert.

[Zhao et al., 2023] Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., Liu, P., Nie, J.-Y., and Wen, J.-R. (2023). A survey of large language models.

[Zhao and Vydiswaran, 2020] Zhao, X. and Vydiswaran, V. G. V. (2020). Lirex: Augmenting language inference with relevant explanation.

[Zhou et al., 2023a] Zhou, Y., Muresanu, A. I., Han, Z., Paster, K., Pitis, S., Chan, H., and Ba, J. (2023a). Large language models are human-level prompt engineers.

[Zhou et al., 2023b] Zhou, Y., Zhang, Y., and Tan, C. (2023b). Flame: Few-shot learning from natural language explanations.

Appendix B

Prompts used in experiments

The prompts are presented whith `[INST]` and `[/INST]`. These are only valid for instruction-tuned models from Mistral family. For LLama3 needs to be replaced by Llama3-specific tokens.

Default 0-shot CoT prompt

[INST]You will be presented with a premise and a hypothesis about that premise. You need to decide whether the premise entails the hypothesis by choosing one of the following labels: (A) Entailment, (B) Neutral or (C) Contradiction. Carefully analyze all details, start your answer with a concise reasoning and end it with the correct label. "Premise": "<>" "Hypothesis": "<>"[/INST] about 69 % accuracy on SNLI

Paraphrase prompt

<s>[INST]Given a sentence, please paraphrase it. Don't change its meaning, just say the same thing in a different way. Sentence to paraphrase: "These two statements are not related to each other."[/INST] These two statements do not have any connection to one another.</s>[INST] Given a sentence, please paraphrase it. Don't change its meaning, just say the same thing in a different way. Sentence to paraphrase: <...>[/INST]

Note: The real prompt which I used had 5 examples.

Add mistake prompt

<s>[INST]Given a text, insert some mistakes that change its meaning and implications. Text to change: "The premise states that team members practice and relax on a basketball court, while the hypothesis specifically states that team members are relaxing after practice. "[/INST] The premise states that team members practice on a football court, while the hypothesis specifically states that team members are relaxing before practice.</s>[INST] [INST]Given a text, insert some mistakes that change its meaning and implications. Text to change:<...>[/INST]

Note: The real prompt which I used had 5 examples.

Prompt generated during OPRO

[INST]Ensure to thoroughly compare the depicted actions and elements in the premise with the requirements and implications of the hypothesis to determine logical consistency. Premise: <> Hypothesis: <> Question: Does the premise entail the hypothesis? Options:(A) Entailment(B) Neutral(C) Contradiction.[/INST]

The prefix before premise is OPRO-generated (71% accuracy SNLI)