Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Exploring Advanced Reasoning Abilities of Large Language Models in Slovak

Bachelor Thesis

2024
Adam Zahradník

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Exploring Advanced Reasoning Abilities of Large Language Models in Slovak

Bachelor Thesis

Study Programme:	Applied Computer Science
Field of Study:	Computer Science
Department:	Department of Applied Informatics
Supervisor:	Mgr. Marek Šuppa

Bratislava, 2024
Adam Zahradník

48302124

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Adam Zahradník
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: anglický
Sekundárny jazyk: slovenský

Názov: Exploring Advanced Reasoning Abilities of Large Language Models in Slovak
Skúmanie pokročilých schopností uvažovania veľkých jazykových modelov v slovenčine

Anotácia: V nedávnej minulosti veľké jazykové modely (LLM) preukázali pôsobivé schopnosti pri riešení rôznych kvantitatívnych úloh a úloh v oblasti vedomostí v oblastiach, ako je matematika, fyzika a informatika. Tieto úspechy sa zvyčajne hodnotia pomocou metodológií navrhnutých pre základné a stredné školy, často odvodených z ich štandardizovaných testov. Tento prístup však predstavuje problém: štandardizované testy sa často dostanú do tréningových datasetov LLM, čo vedie k skresleným hodnoteniam ich výkonu. Okrem toho hodnotenie rozumových schopností (či schopnosti uvažovania) LLM prebieha primárne v angličtine, čo vyvoláva obavy z použiteľnosti a zovšeobecniteľnosti výsledkov týchto hodnotení.

Cieľ: Ciele bakalárskej práce zahŕňajú (ale nie sú obmedzené na)
- analýza súčasného stavu v hodnotení schopnosti uvažovania
- vytváranie a/alebo zhromažďovanie logických hodnotiacich datasetov v slovenskom jazyku na základe úloh/cvičení z matematických, fyzikálnych alebo informačných olympiád, ako aj rôznych korešpondenčných seminárov z rovnakých oblastí
- vyhodnotenie najmodernejších LLM na pripravených datasetoch
- analýza výstupov (a chybových režimov) vytvorených modelmi s najlepšími výsledkami

Literatúra: Hendrycks, Dan, et al. "Measuring mathematical problem solving with the math dataset." arXiv preprint arXiv:2103.03874 (2021). (https://arxiv.org/pdf/2103.03874.pdf)
Cobbe, Karl, et al. "Training verifiers to solve math word problems." arXiv preprint arXiv:2110.14168 (2021). (https://arxiv.org/pdf/2110.14168.pdf)
Wei, Jason, et al. "Chain-of-thought prompting elicits reasoning in large language models." Advances in Neural Information Processing Systems 35 (2022): 24824-24837. (https://arxiv.org/abs/2201.11903)
Sawada, Tomohiro, et al. "Arb: Advanced reasoning benchmark for large language models." arXiv preprint arXiv:2307.13692 (2023). (https://arxiv.org/pdf/2307.13692.pdf)

Vedúci: Mgr. Marek Šuppa
Katedra: FMFI.KAI - Katedra aplikovanej informatiky

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Vedúci katedry: doc. RNDr. Tatiana Jajcayová, PhD.

Dátum zadania: 15.10.2023

Dátum schválenia: 16.10.2023

doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.....

študent

.....

vedúci práce

48302124

Comenius University Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname: Adam Zahradník
Study programme: Applied Computer Science (Single degree study, bachelor I. deg., full time form)
Field of Study: Computer Science
Type of Thesis: Bachelor´s thesis
Language of Thesis: English
Secondary language: Slovak

Title: Exploring Advanced Reasoning Abilities of Large Language Models in Slovak

Annotation: In the recent past, Large Language Models (LLMs) have shown impressive capabilities in tackling various quantitative reasoning and knowledge challenges in fields like mathematics, physics, and computer science. These achievements are typically assessed using benchmarks designed for primary and secondary school levels, often derived from standardized tests. However, this approach poses a problem: standardized tests frequently find their way into the training data of LLMs, leading to skewed performance evaluations. Additionally, the evaluation of LLMs' reasoning abilities primarily occurs in English, raising concerns about the applicability and generalizability of the results.

Aim: The goals of the bachelor's thesis include (but are not limited to)
- analysis of the current state-of-the-art in reasoning capability evaluation
- creation and/or collection of reasoning evaluation datasets in Slovak language based on tasks/exercises of Olympiads in Mathematics, Physics or Informations, as well as various correspondence seminars in the same areas
- evaluation of state-of-the-art LLMs on the prepared datasets
- analysis of the outputs (and error modes) produced by the best performing models

Literature:
Hendrycks, Dan, et al. "Measuring mathematical problem solving with the math dataset." arXiv preprint arXiv:2103.03874 (2021). (https://arxiv.org/pdf/2103.03874.pdf)
Cobbe, Karl, et al. "Training verifiers to solve math word problems." arXiv preprint arXiv:2110.14168 (2021). (https://arxiv.org/pdf/2110.14168.pdf)
Wei, Jason, et al. "Chain-of-thought prompting elicits reasoning in large language models." Advances in Neural Information Processing Systems 35 (2022): 24824-24837. (https://arxiv.org/abs/2201.11903)
Sawada, Tomohiro, et al. "Arb: Advanced reasoning benchmark for large language models." arXiv preprint arXiv:2307.13692 (2023). (https://arxiv.org/pdf/2307.13692.pdf)

Supervisor: Mgr. Marek Šuppa
Department: FMFI.KAI - Department of Applied Informatics
Head of department: doc. RNDr. Tatiana Jajcayová, PhD.

48302124

Comenius University Bratislava
Faculty of Mathematics, Physics and Informatics

Assigned: 15.10.2023

Approved: 16.10.2023 doc. RNDr. Damas Gruska, PhD.
Guarantor of Study Programme

Student

Supervisor

Acknowledgments: Tu môžete poďakovať školiteľovi, prípadne ďalším osobám, ktoré vám s prácou nejako pomohli, poradili, poskytli dáta a podobne.

Abstrakt

Slovenský abstrakt v rozsahu 100-500 slov, jeden odstavec. Abstrakt stručne sumarizuje výsledky práce. Mal by byť pochopiteľný pre bežného informatika. Nemal by teda využívať skratky, termíny alebo označenie zavedené v práci, okrem tých, ktoré sú všeobecne známe.

Kľúčové slová: jedno, druhé, tretie (prípadne štvrté, piate)

Abstract

Abstract in the English language (translation of the abstract in the Slovak language).

Keywords:

Contents

Chapter 1

Introduction

Latest developments in the field of artificial intelligence led to great improvements in the abilities of large language models to solve many different types of tasks. Prior work demonstrates ambitious results on tasks that challenge the models' ability to reason about maths, physics and informatics. Researchers introduced many datasets and methods to evaluate the large language models' abilities to solve these problems.

These datasets and methods are usually based off standardized elementary and high school tests in these fields. Standardized tests that are publicly available can end up included in the models' training data, which can bias the models' evaluation results. Currently, most available datasets and evaluation methods are in English, which raises the question whether observed large language models' abilities can be generalized to other languages as well.

As organizers of Slovak high school competitions ourselves, we want to find out how current large language models deal with the problems to gain insights into their capabilities. A fellow organizers already experimented with using large language models to solve our problems, but we want to try it out on a much larger dataset. This will allow us to better understand the models, their pitfalls and explore different techniques of working with them.

1.1 Large Language Models

Large language models are a recent advancement in artificial intelligence, particularly in the realm of generative models. They operate by first receiving an initial input text, which is also called a prompt. Then, the model uses its neural network to predict the next word or token. This token is then appended to the prompt, creating an extended text input. The process is repeated until a specified length of text or another stopping criterion is reached. The result is a coherent and contextually relevant piece of text, generated entirely by the model's learned patterns and associations within its training

data.

The prompt that was given to the large language model together with the generated text is called a context window. The model can recall information from its context window. This allows the user to provide additional context to the model. Additionally, the model can in some circumstances use its context window as a kind of scratch pad and thus better prepare its output. This is taken advantage of by some of the prompting techniques.

It was observed that large language models exhibit common-sense reasoning capabilities [1]. More importantly, these models can perform advanced reasoning needed to solve mathematical problems [2]. Even though large language models are often less capable than humans in solving such problems, it still allows them to be used in a wide range of applications.

In our research, we will utilize language models provided by OpenAI, specifically the GPT-3.5-Turbo [3] and GPT-4 [2]. These models will be used to benchmark our dataset, enabling us to assess and compare the performance of each model under various scenarios.

In addition to those commercial models, we will also run our benchmarks against the open weight Llama 3 model from AI@Meta [4]. It should be noted that Llama is not multi-lingual, but we still wanted to explore its abilities.

Open weight models such as Llama are publicly available for download and use on your own hardware. This makes them very attractive to researchers and companies because they don't require a 3rd party API, like OpenAI's models do. They are also much cheaper to run, because you pay for your own hardware.

1.2 Prompting Techniques

Prior research has shown that manipulating the way in which the model's prompt is constructed can have a significant impact on the quality of the resulting output. In this section, we introduce successful techniques that we use to compare large language model performance in the Slovak language.

1.2.1 Zero-, One- and Few-Shot Prompting

Even though large language models are trained on generic text corpus datasets, prior research has indicated that LLMs do not require fine-tuning the model on the desired task [5][6].

Brown et al. [6] shows that this fine-tuning step can be replaced by a technique called few-shot prompting. Few-shot prompting provides the model with few examples of the desired task, along with sample solutions right in the model's input. This

measurably improves the models' capability to solve the desired task, even though the model was not trained to solve that particular task in the first place. Few-shot prompting is typically done with 10 to 100 task examples, depending on the size of the model's context window.

There are two related techniques to few-shot prompting, introduced by Brown et al. [6]: one-shot and zero-shot prompting. One-shot prompting is done in the same way as few-shot, but the model is provided with only one example of the task. Zero-shot prompting is a similar, but generally less effective, technique in which the model is provided with a natural language description of the task instead of any examples.

1.2.2 Chain-of-Thought

Another promising prompting technique is chain-of-thought introduced by Wei et al. [7]. This technique mimics one's own thought process when solving tasks. The goal of chain-of-thought prompting is to make the model generate a series of intermediate steps that lead to the final answer of a problem. Wei et al. [7] shows that large language models are capable of generating such chain-of-thoughts when provided with such chain-of-thoughts in the examples used for few-shot prompting. The model is provided with example solutions that walk the reader through the different steps leading to the final answer.

1.2.3 Zero-shot Chain-of-Thought

Classical chain-of-thought prompting as introduced by Wei et al. [7] has the disadvantage of needing to provide examples of task solutions including chain-of-thoughts, which are usually not readily available.

To address such problems, Kojima et al. [8] introduced a simple, yet effective technique, called zero-shot chain-of-thought. The idea is that when the model is prompted to "think step by step", it can generate a chain-of-though without needing any examples beforehand. So, the model is prompted with the question and a simple prompt like "Let's think step by step" to force it to generate a chain-of-thought.

1.2.4 Generated Knowledge

Another similar approach to zero-shot chain-of-thought was demonstrated by Liu et al. [9]. The generated knowledge prompting technique leverages the fact that large language models can use their context windows for short-term memory. This is very similar to some teaching techniques employed when teaching humans new concepts.

The model is prompted to first describe all concepts relevant to solving the problem and then attempt to solve the problem. This way, the model will introduce a lot of

new information into its context window. It can later retrieve information from the context window to help itself to solve the problem.

1.2.5 Dual Prompt Generated Knowledge

The generated knowledge prompting method has a slight disadvantage - the model has a limited number of words or tokens it can produce. Because of this, it can spend a lot of its available space on preparing the relevant context and end up not having enough tokens left for the solution itself.

Dual prompt generated knowledge improves on this by prompting the model to only generate the relevant context. After it generates the context, the model is prompted again with the original question and the context it generated. This allows the model to generate longer answers.

1.2.6 Least-to-Most Prompting

Least-to-most prompting takes advantage of the model's context window combined with multiple prompts. The idea introduced by Zhou et al. [10] is that we break the problem into smaller sub-problems, which are then solved sequentially.

We start by prompting the model with the problem and ask it to list out the sub-problems that are required to solve the whole problem. We then take the first sub-problem it generates and ask it to solve it. This is usually done by prompting it with the original problem and a question to solve a given sub-problem. The solution of the sub-problem is then appended to the prompt along with another sub-problem. This process is repeated until the model solves all sub-problems. At that point, we should have the whole solution.

1.3 Existing Datasets

Large language models have already been evaluated on mathematical reasoning tasks by researchers using numerous datasets. Most of these datasets were created by scraping problems from the internet. We provide a comparison of few selected datasets related to our work to better understand the types and problems involving creating such a dataset. We provide an overview of selected datasets in Figure 1.1.

MultiArith released by Roy and Roth [11] contains multistep arithmetic problems without irrelevant quantities. That means that those problems require a combination of different arithmetic operations to get the answer and don't have any information that isn't needed to solve the problem. The dataset contains symbolic solutions.

Math23K by Wang, Liu, and Shi [12] consists of Chinese elementary school maths problems scraped from the internet. This dataset contains only problems with single linear unknown variable. This dataset contains only symbolic solutions.

AQuA introduced by Ling et al. [13] consists of multi-choice word problems covering a broad range of topics and difficulty levels. This dataset also contains descriptions of the rationale to reach the correct answer.

MATH is a dataset consisting of challenging competition mathematic problems with step-by-step natural language solutions introduced by Hendrycks et al. [14]. The problems were retrieved from United States' mathematics competitions. These problems are designed to be challenging for humans and often require more than just basic application of mathematic tools.

GSM8K released by Cobbe et al. [15] consists of multistep middle school word problems with natural language solutions. These problems take between 2 and 8 steps to solve. A bright student should be able to solve all of them.

ASDiv is a mathematical word problem dataset with a strong emphasis on great diversity. This dataset of arithmetic and algebraic problems was introduced by Miao, Liang, and Su [16].

SVAMP introduced by Patel, Bhattamishra, and Goyal [17] contains many variations of elementary school mathematical word problems.

MGSM is a multilingual dataset introduced by Shi et al. [18] containing 250 manually translated grade-school problems from the GSM8K.

Dataset	Size	Answer	Difficulty	Language
MultiArith	600	symbolic	elementary school	English
Math23K	23 161	symbolic	elementary school	Chinese
AQuA	100 949	multi-choice	diverse	English
MATH	12 500	natural language	competitions	English
GSM8K	8 500	natural language	elementary school	English
ASDiv	2 305	symbolic	elementary school	English
SVAMP	1 000	symbolic	elementary school	English
MGSM	250	natural language	elementary school	multilingual

Figure 1.1: Comparison of existing datasets

REFOK FINAL

1.4 Evaluating LLM Answers

There are few ways to evaluate answers generated by LLMs. The method used varies depending on the type of question. We will only focus on problems that have known solutions that can be used to verify the model's answers.

For questions with concrete numerical answers, the most straightforward approach is to compare the LLM's numerical answer to the correct one. However, LLMs usually generate natural language output. This usually involves extracting the number or equation from the model's output. Such an approach is very precise, as shown by Hendrycks et al. [14], Wang, Liu, and Shi [12] and Sawada et al. [19].

LLMs can also be evaluated on multiple-choice problems. This is done by providing the model with the options and prompting it to select one of the provided options [18]. With the right prompting, models can output answers that can be extracted in more than 97% of the time [19]. Alternatively, the model is not provided with the options, but its answer is extracted as a number and then compared to the available options as in Amini et al. [20].

The most problematic questions are those which have open answers that cannot be easily extracted from the model's output. Unfortunately, these are the types of questions that we are most interested in. One of the approaches is to try and convert these questions into ones that allow automatic answer extraction. This is done by extracting a number or equation from the reference solution and trying to match it with the model's output. Other methods involve changing the question into a multiple-choice one. Both of those approaches, while valid, do not fully evaluate the model's capabilities to correctly solve problems that require natural language solutions.

The most straightforward approach to evaluating such problems is using a human evaluator. Nonetheless, this approach is highly labor-intensive and ineffective. We can leverage the existing model's capabilities and use it instead of a human to evaluate the answers. Such an approach is called model-based evaluation [19].

In its simplest form, the model is provided by the reference solution and the output that should be evaluated. It is then asked to grade the provided output. Prior research indicates that such an approach is possible, but the evaluations are not reliable enough to be used alone [21] [22]. Some researchers went so far as to avoid providing the model with the reference solution. Those experiments provided promising results, but they still failed to be reliable enough [23].

An improved approach was introduced by asking the model to generate evaluation rubrics, and then using those rubrics to evaluate the solutions [19]. The model is provided with the reference solution and generates rubrics and allocates points to them. It was shown by Sawada et al. [19] that GPT-4 designs rubrics that cover most of the solution steps correctly, but sometimes fail to properly allocate points based on their importance. The model is quite reliable on assigning the correct number of points to solutions based on the generated rubrics. However, the model cannot score solutions that do not follow the generated rubrics, but are otherwise correct. Another issue with this approach is that the model attempts to assign points to attempted solutions that are outside the generated rubrics. A human evaluator would score these solutions with

zero points [19]. Knowing its limitations, we will base our approach on this method to evaluate models on our dataset.

1.5 Prior Research

The MGSM paper by Shi et al. [18] tries to evaluate models' reasoning abilities in multiple languages. They achieve this by manually translating 250 problems from GSM8K [15] into ten typologically diverse languages, which they then used to benchmark GPT-3 on their dataset. Their research reveals that results are very similar, with insignificant differences between the various languages. It has been demonstrated that using an intermediate English chain-of-thought provides results that are on par with or better than answers written in the question's original language.

Another multilingual research by Ahuja et al. [24] comprehensively evaluates the models on various multilingual datasets. Even though this research does not evaluate the advanced reasoning abilities, it demonstrates the overall capabilities of large language models to reason in languages other than English. Their results show no significant differences between the results achieved in the different languages.

1.6 Related Benchmarks

In this section, we examine existing benchmarks and findings related to advanced reasoning abilities. We aim to establish a baseline against which we can compare our own findings. This comparison will not only help in highlighting any unique contributions from our research, but also in identifying potential gaps and opportunities for further investigation.

Experiments conducted by OpenAI et al. [2] indicate that when utilizing a few-shot prompting, GPT-3.5-Turbo achieves a score of 57.1% on the GSM8K benchmark. In contrast, the more advanced GPT-4 demonstrates significantly improved performance, reaching a score of 92.0% on the same benchmark. Similar results for GPT-4 were also published by Bubeck et al. [25], who scored 87.1%. Results for Llama 3 show a score of 79.6% on the same benchmark [4]. The GSM8K benchmark shares similarities with our dataset in that it features answers composed in natural language. However, the benchmark's difficulty is at an elementary school level, while our dataset is targeted for high school students.

In their research, Sawada et al. [19] employed a model-based evaluation technique to determine the proficiency of GPT-4 in solving academic challenges, specifically within the realms of mathematics and physics. The dataset used is similar to ours, featuring problems typically encountered in mathematical competitions. The results demon-

strated that GPT-4 achieved a score of 50.5% in physics problems with symbolic answers and 33.7% in similar mathematics problems. Additionally, when faced with problems resembling formal proofs, GPT-4 managed to secure a score of 38%.

Experiments conducted by Bubeck et al. [25] using the GPT-4 have demonstrated proficiency in solving mathematical problems, scoring 42.5% on the MATH benchmark. GPT-3.5-Turbo, on the other hand, scored 0.00% in the same benchmark. When tested against Llama 3, the scores were 30.0% [4]. This benchmark is particularly significant as it consists of natural language solutions and competition problems that are similar to those found in our dataset.

Dataset	**GPT-3.5-Turbo**	**GPT-4**	**Llama 3**
GSM8K	57.1%	87.1%	79.6%
MATH	TODO: %	42.5%	30.0%

Figure 1.2: Models' results on existing datasets

Chapter 2

Our Goals

Our research aims to explore the extent to which advanced reasoning capabilities exhibited by large language models can be observed in the Slovak language. To achieve this goal, we intend to compare our findings against existing research conducted in the English language. By using the techniques established in prior research, we seek to investigate whether similar patterns of reasoning can be seen when large language models are working with Slovak problems.

We will also experiment with multiple prompting techniques to compare their effectiveness in the Slovak language. By doing this, our research aims to provide insights into the ability to generalize results of large language models across different languages.

To accomplish this, we will firstly introduce a new dataset consisting of various tasks aimed at advanced reasoning in maths, physics and informatics in the Slovak language from local high school competitions.

We then establish a framework for evaluating large language models using our dataset. We proceed to benchmark multiple large language models on our dataset to evaluate their performance in tasks requiring advanced reasoning abilities. Through experimentation, we test different prompting techniques, aiming to replicate findings observed in prior research conducted in English.

We aim to answer the following questions:

- What is the overall performance of large language models on our dataset?
- Which prompting techniques are effective in the Slovak language, and to what extent?
- How do our results compare to those published by English researchers?
- How well do models perform when given English problems compared to Slovak problems?

Chapter 3

The Dataset

Our problem dataset contains various problems and their solutions from local high-school competitions. As they are competition problems, they are designed to be challenging for high-school students. An average student should be able to solve about half of our problems. The dataset contains problems split into three categories: maths, physics and programming. These problems usually require the student to embrace innovative approaches, and to document them thoughtfully in their solution. Programming problems require description of applied algorithms. Students also submit their code, which is then automatically tested against the original solution.

3.1 Creating the dataset

Most of the problems and solutions were taken from the competitions' archives in the form of Markdown or LaTeX documents. Those were then cleaned up from the competitions' special markup tags to generate the cleaned Markdown (with TeX maths) format of problems. This way, we were able to collect a total of 1108 problems. With 361 of them being from a maths competition, 479 from a physics competition and 268 from a programming competition as shown in Figure 3.1. This corresponds to about 8 years worth of competition problems.

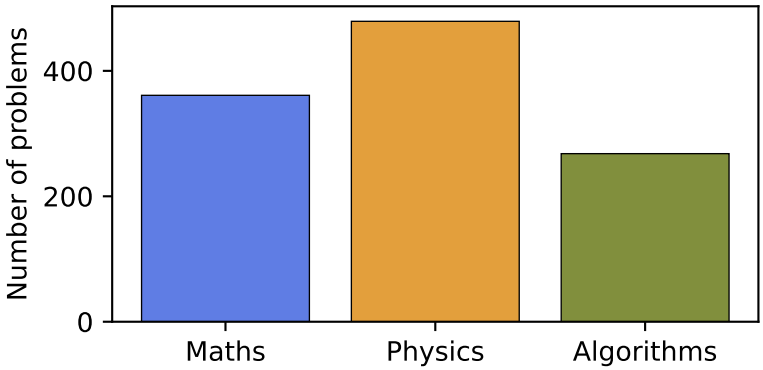

Figure 3.1: Composition of the dataset problems

REFOK FINAL

Additionally, we have tried to recover older problems that were accessible only in PDF format. We tried using the Nougat model by Blecher et al. [26] to convert those PDF files into raw text. Unfortunately, we encountered a few shortcomings when using the model on texts with diacritics. Nougat has on multiple occasions failed to read some letters correctly, confused diacritics with mathematical symbols and ignored diacritics altogether. Some of the problems observed are shown in Figure 3.2. However, this was not very surprising, since Nougat is still in its early stages. Because of those issues, we ultimately chose not to move forward with extracting problems from the PDF files. This may be further investigated in a future research when the maths OCR technology advances.

```
in:  počas vysokoškolského štúdia
out: pocas vysokolskolskolskolskolskolskols...

in:  ako mriežku n × m znakov
out: ako mri\(\hat{\text{e}}\)ku \(n\times m\) znakov
```

Figure 3.2: Examples of some Nougat failures

REFOK FINAL

3.2 Overview of the problems

The problems in our dataset are not only diverse in terms of the primary target area (maths, physics and algorithms), but in addition, there are also different types of such problems.

As all of our problems require not only an answer, but also a clear explanation of the calculations and reasoning used by the student. We will try to evaluate the quality of the explanation.

3.2.1 Maths Problems

The overwhelming majority of our maths problems are based on the student having to prove whether a given statement is true or not. Other problems require the student to quantify some equations or otherwise calculate a numerical result. There are some problems that want to enumerate all numbers, functions, etc. that satisfy certain conditions. Furthermore, there happens to be a tiny number of problems that require to carry out some geometric construction. The relative number of problems from each category is shown in Figure 3.3. An example maths problem is provided in Figure 3.4.

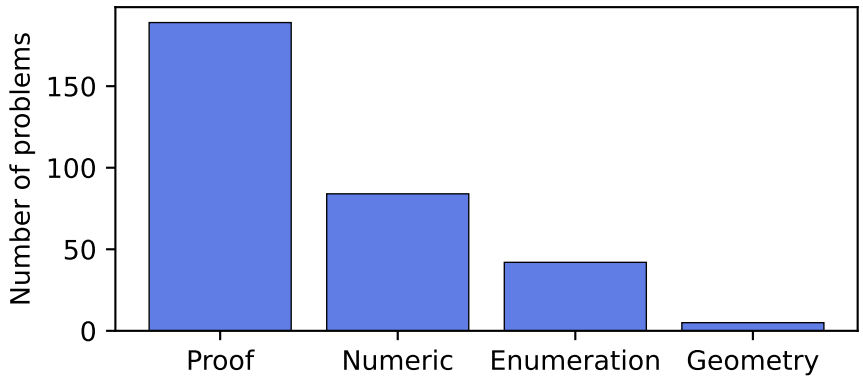

Figure 3.3: Types of maths problems in our dataset

REFOK FINAL

Let $f : \mathbf{R}^+ \to \mathbf{R}$ be a function such that the functions $f(x) - x^3$ and $f(x) - 3x$ are increasing. Determine whether the function $f(x) - x^2 - x$ must be monotone.

Figure 3.4: Example maths problem

REFOK FINAL

3.2.2 Physics Problems

We divide physics problems into two categories. The first category of problems is problems that only need theoretical knowledge to explain a relationship between physics variables or explain some physical phenomena. They usually involve figuring out some equations, explaining them, and using them to obtain an answer to the question. The second category requires the student to come up with an experiment setup, execute and document the experiment. The relative number of problems in each category is shown in Figure 3.5. An example problem is provided in Figure 3.6.

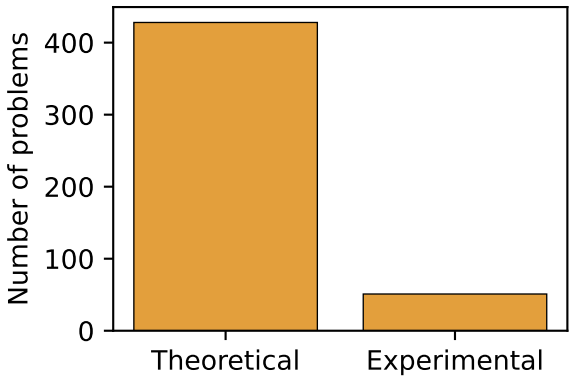

Figure 3.5: Types of physics problems

REFOK FINAL

I'm sitting in a bubble bath and bubbles are flowing up my back. They seem very cold, perhaps even colder than the surrounding air. Why is that?

Figure 3.6: Example physics problem

REFOK FINAL

3.2.3 Algorithmic Problems

All of our algorithmic problems focus on figuring an effective way to solve some problem. This usually means using different algorithms in unusual ways or coming up with entirely new algorithms to solve the problem. Algorithmic problems are also among the longest in our dataset. This is because they contain plenty of details about the input and output format, input size constraints, along with a fictional story to provide some practical background to the problem. An excerpt from one such problem is provided in Figure 3.7.

You have been given points on a plane. Find out how non-random they are, that is, the vertices of how many triangles they form.

Figure 3.7: Example of an algorithmic problem (excerpt)

REFOK FINAL

3.2.4 Difficulty

The problems in our dataset have varying degrees of difficulty. In the real competitions they were taken from, they tend to be sorted by estimated difficulty. The few first problems should be solvable by all high school students, whereas the last problems are usually solved only by students engaging in national or international competitions. The

relative difficulty data is used to assign every problem a difficulty score on a scale of 1 to 10, with 10 being the most difficult. The difficulty distribution is shown in Figure 3.8. This score will later be used to quantify the abilities of large language models in solving these problems.

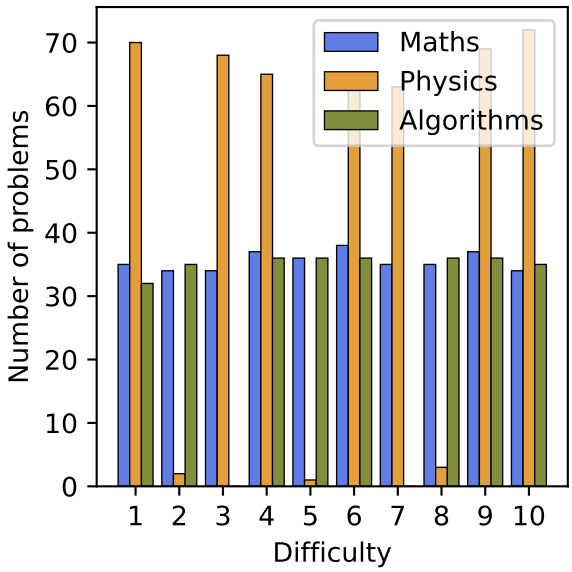

Figure 3.8: Problem difficulty distribution

REFOK FINAL

3.2.5 Length

An average problem in our dataset has a length of 195 words. However, most of our problems have less than 200 words, as shown on Figure 3.9. This is because mathematical and physical problems are typically brief, whereas algorithmic problems tend to be more extensive. This anomaly was discussed in section 3.2.3, and is mostly due to a longer problem story and details about handling input and output data.

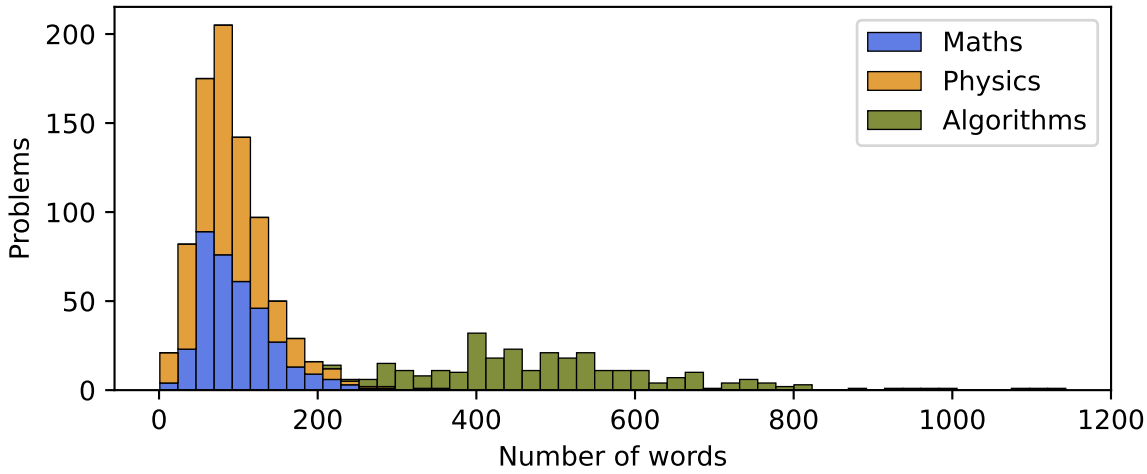

Figure 3.9: Number of words in the problem statement

REFOK FINAL

3.3 Overview of the solutions

As stated previously, our dataset contains solutions in natural language for every problem. That means that the solutions contain explanations, proofs and other details. An example of such a natural language solution is provided in Figure 3.10.

> When we put our finger below the surface, the water level rises a little. This will increase the hydrostatic pressure at the bottom of the right bowl. Since the pressure at the bottom of the left bowl has not increased, the scales will tip to the right.

Figure 3.10: Example solution

REFOK FINAL

Our reference solutions vary greatly in their word count. An average reference solution is 652 words long, with the longest reference solution consisting of 3682 words. In our maths problems, the average length is 487 words. For our physics problems, the typical length increases to 660 words. Meanwhile, our algorithmic problems tend to have more detailed solutions, with an average length of 850 words. The distribution of the word count is shown in Figure 3.11.

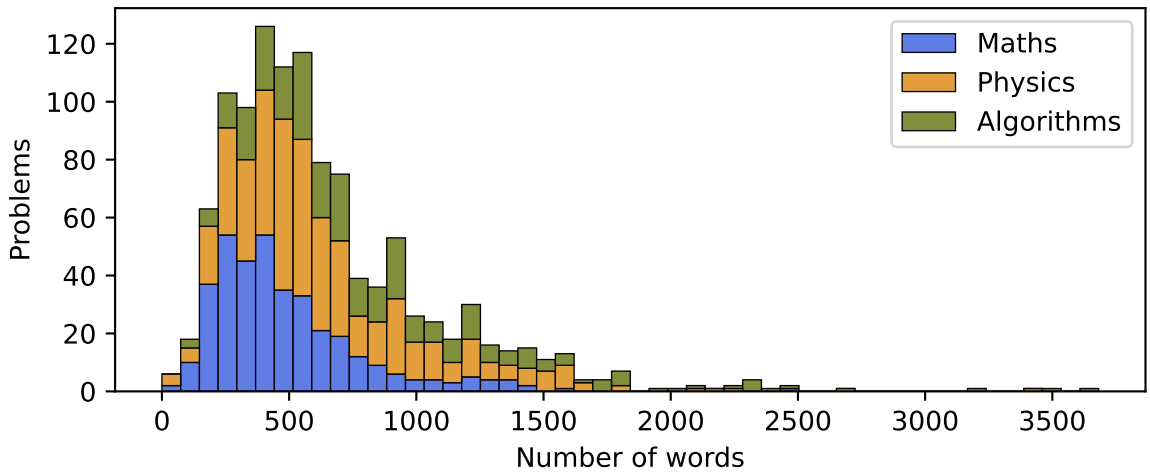

Figure 3.11: Number of words in the reference solution

REFOK FINAL

Chapter 4

The Benchmark

4.1 Prompting

4.1.1 Zero-Shot Baseline

At first, we evaluate the model's ability to zero-shot solve problems in our dataset to establish a baseline we will later use for comparison. The model is zero-shot prompted with the problem statement and its output is evaluated. We ask the model to end its output with the numerical answer of the problem if it exists. For problems that require measuring some experimental data, we ask the model to try and generate it. We do not want the model to generate code for programming problems, as we are mostly interested in its capabilities to produce a natural language solution.

4.1.2 One-Shot

We continue by one-shot prompting the model with a randomly selected example problem and solution from the dataset. The model is also provided by another problem which it tries to solve based on the one-shot example. Brown et al. [6] have shown that "one- and few-shot performance is often much higher than true zero-shot performance", which we try to reproduce in the Slovak language.

4.1.3 Few-Shot

The models are also evaluated with few-shot prompting. We randomly select a few (2-5) problems from the dataset. The models are then provided by the problem statements and reference solutions for the selected problems. Few-shot tests often achieve higher scores than when the models are one-shot [6].

4.1.4 Zero-Shot Chain-Of-Thought

We also evaluate the zero-shot chain-of-thought prompting technique introduced by Kojima et al. [8]. We zero-shot prompt the model with a problem and instruct it to "think step by step". Results observed by Kojima et al. [8] show promising improvements on most types of tasks when compared to classical zero-shot prompting and comparable results to normal chain-of-thought. We try to reproduce such results using our dataset in Slovak language.

4.1.5 Generated Knowledge

Additionally, we test generated knowledge prompting. The model is prompted to first write out all ideas and concepts required to solve the given problem, and to then produce the solution. Liu et al. [9] shows potential for improvements in the models' abilities when using generated knowledge prompting.

4.1.6 Dual-Prompt Generated Knowledge

We also test the models on a slight variation of the generated knowledge approach. At first, the model is provided with the problem statement and asked to write out all relevant ideas, concepts, equations, etc. When the model produces this relevant context, it is prompted to solve the provided problem. The main difference between this and the previously described generated knowledge prompting is that the model is prompted twice, and is always asked to only do one thing (write out all relevant information, solve the problem).

4.1.7 Least-To-Most Prompting

In addition, models are also least-to-most prompted. The model is provided with the problem statement and asked to produce sub-problems that need to be solved and lead to the final solution. When it comes up with the sub-problems, it is then sequentially prompted to solve each of the sub-problem one-by-one. Then, the model is prompted once again to write one complete solution that will be used for grading. This method is shown to reach improved results when compared to classic chain-of-thought [10].

4.2 Other Experiments

4.2.1 Output Language

When interacting with large language models, we have noticed that they sometimes decide to produce their output in English instead of the language that was used in the

prompt. We want to see whether the fact that the model decides to produce output in a different language on its own has any influence on the score it receives.

4.2.2 Problem Statement Language

We also want to test whether translating the problem statements itself into English has any impact on the scores. We are interested in this because the majority of data large language models are trained on is in English, so they might have better understanding in English. We also want to use these results to compare reasoning capabilities in English and Slovak language.

4.3 Answer Evaluation

Sice the vast majority of our problems do not contain a concrete numerical or symbolic solution, we must introduce a method to grade the natural language outputs produced by the models. One of the commonly used approaches is to use a language model to grade the answers.

We tried zero-shot prompting the model with the reference solution and the generated solution. This unfortunately led to unreliable and inconsistent results, similar to those observed by Kortemeyer [23].

Our approach to evaluating the answers is inspired by work by Sawada et al. [19]. We zero-shot GPT-4 with the reference solution and prompt it to generate a grading rubric. After playing with the prompts, we were able to achieve satisfactory results.

Then, those generated rubrics are used to produce a grading score on a scale of 0 to 10. This is achieved by zero-shot prompting the model with the rubric, the generated solution and asked to provide a comment for every point in the rubric and a final score. Our method is also visualized in Figure 4.1.

After some experiments, we have established the GPT-4 model as our grader and rubric generator. At first, we tried GPT-3.5-Turbo, but were dissatisfied with its capabilities. The model was often referencing to non-existent claims in the solution or the grading rubric itself. It sometimes made entirely new and incorrect claims about the concepts involved in the problem. The model also failed to keep attention to details, which was most noticed in maths expressions. For example, the model did not notice a difference between $\frac{x}{2}$ and $\frac{x+1}{2}$. We also found that the model failed to follow the mathematical reasoning of a solution properly, probably due to the aforementioned issues. We then experimented with the larger GPT-4 model, with which we did not experience any of those problems. We also noticed that the quality of produced comments was greatly improved.

We randomly picked a subset of the generated rubrics (60 rubrics in total) and

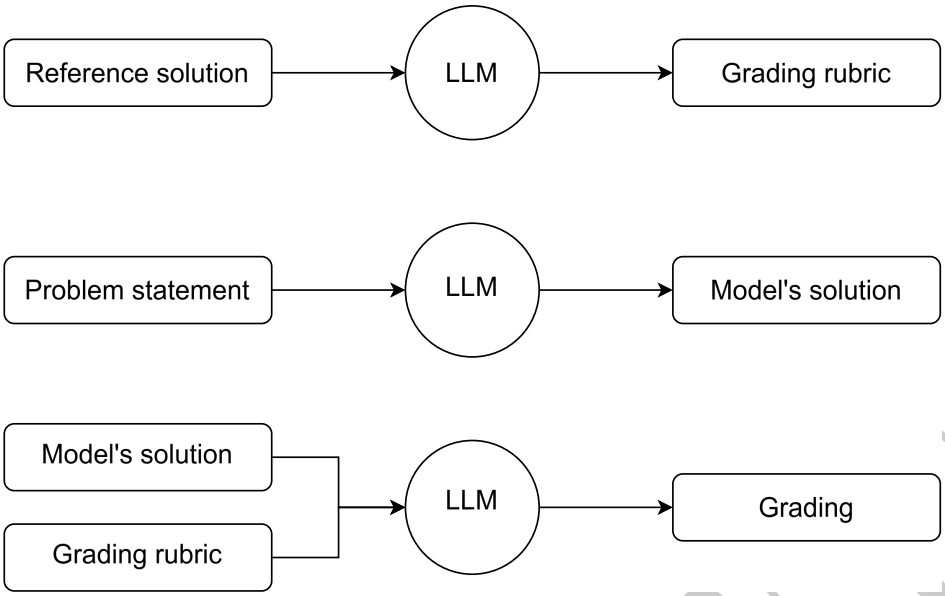

Figure 4.1: Our grading method

REFOK FINAL

questioned the competition organizers whether they agreed with them by using a 5-point Likert scale[1]. By doing this, we gained insight into the quality of the rubrics. On average, our organizers reached an agreement Likert score of 3.98.

We then asked the organizers to check the score provided by GPT-4 against the used rubric. This was done by grading a few randomly selected solutions for every examined rubric manually, while adhering to the rubric. The average difference between the grading provided by the model and a human evaluator, when both adhere to the generated rubric, was 1.05 points.

At last, we asked the organizers to grade a few solutions manually without using the generated rubric. Their scores differed to the ones provided by the model by 1.87 points on average.

We also compared the scores provided by the human evaluators when grading against the rubric and when grading based on their own knowledge. The score difference in that case was 0.96 points.

TODO: (whole section ↑) update numbers when physics data arrives

This system was also tested by providing it with the reference solutions to grade them. On average, GPT-4 graded the reference solution with 9.6 points out of 10, with most of the scores being 10/10, as shown on Figure 4.2. There was one instance of GPT-4 awarding 11 points, even though it was instructed to award up to 10.

[1]The Likert items used were: 1 Strongly disagree / 2 Disagree / 3 Neither agree nor disagree / 4 Agree / 5 Strongly agree

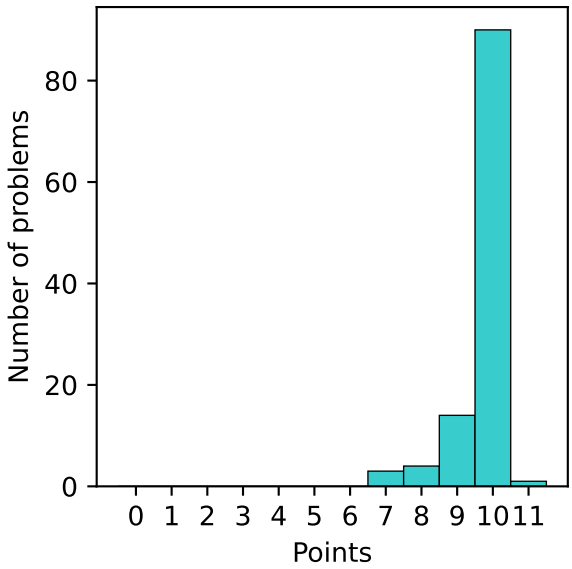

Figure 4.2: Scores awarded to the reference solutions.

REFOK FINAL

Chapter 5

Results

In this section, we present the results of the GPT-3.5-Turbo and GPT-4 models when scored on our dataset. In all cases presented, the models were zero-shot with the problem statement and their solutions were graded by GPT-4 as described in Section 4.3.

5.1 Zero-Shot Prompting (Baseline)

5.1.1 Maths Problems

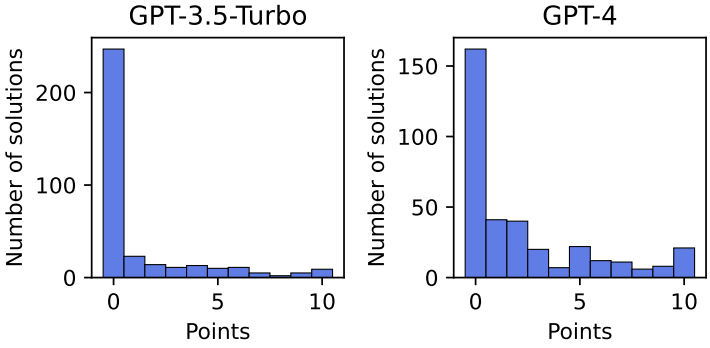

Figure 5.1: Models' performance on the maths portion of our dataset

REFOK TODO: llama data

The models performed poorly on the maths problems in our dataset. GPT-3.5-Turbo scored an average of 1.24 out of 10, with a median of 0. GPT-4 as the larger and more powerful model, scored a higher average of 2.26 out of 10, with a median of 1. As can be observed in Figure 5.1, the majority of scores were 0 out of 10.

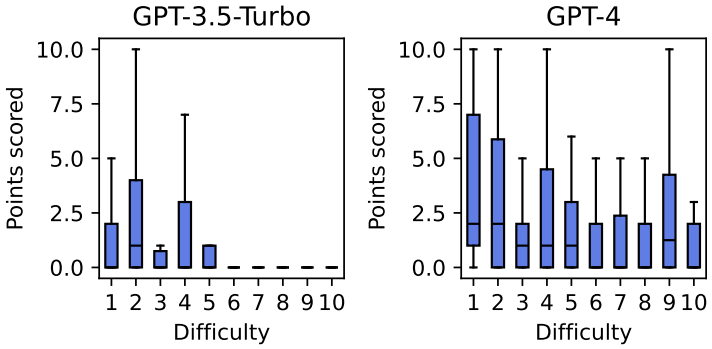

Figure 5.2: Models' performance in comparison to the difficulty when solving the maths portion of our dataset

REFOK TODO: llama data

When we look at the results in the terms of the difficulty of the problems, as shown in Figure 5.2, we notice that GPT-3.5-Turbo can only partially solve easier problems with difficulty between 1 and 5. It, however, does not seem to be able to solve harder problems at all.

GPT-4 is better overall, but it's better at solving simple problems. It is still severely lacking in the realm of our harder problems, but not as much as GPT-3.5-Turbo.

We also looked at the results difference between different types of our problems (proof, numeric, enumeration, geometry). There is no significant difference in scores received between the problem types.

5.1.2 Physics Problems

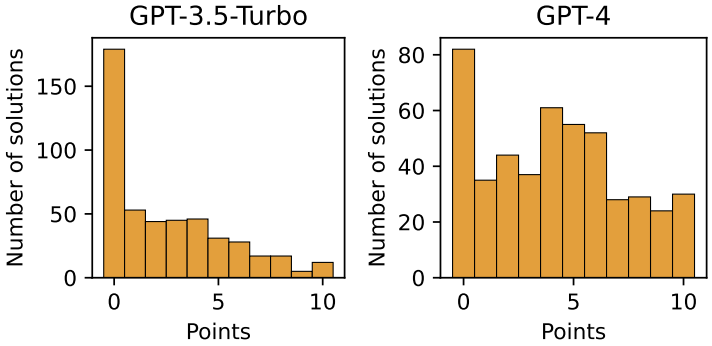

Figure 5.3: Models' performance on the physics portion of our dataset

REFOK TODO: llama data

The performance in our physics problems is a bit better. We have observed an average score of 2.46 out of 10 for GPT-3.5-Turbo with a median of 1.5 and an average of 4.15 and a median of 4 for GPT-4. We note that GPT-4 has a more even distribution of the scores, which can also be observed in Figure 5.3 and also has a median much closer

to the average. The largest number of scores in GPT-3.5-Turbo is still 0. This is still the case for GPT-4, but the score of zero is no longer a majority.

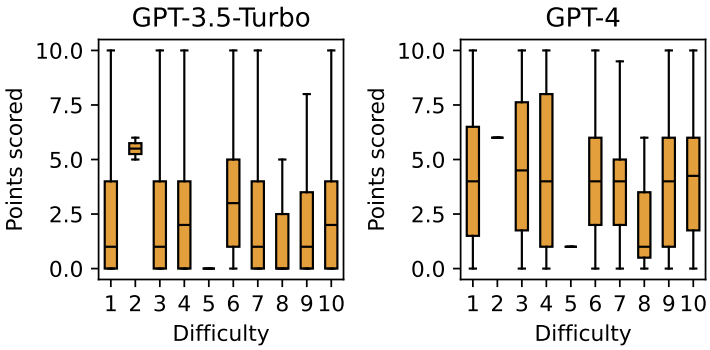

Figure 5.4: Models' performance in comparison to the difficulty when solving the physics portion of our dataset

REFOK TODO: llama data

When looking at the distribution of points based on the problem difficulty as shown in Figure 5.4. We notice an improvement across all difficulties. There is still a bias towards easier problems, but it is not that noticeable as it was in the case of our maths subset.

There is an anomaly in this chart at difficulty levels 2 and 5. The difficulty levels of our physics subset had to be scaled up to have a single scale across all parts of the dataset.

5.1.3 Algorithmic Problems

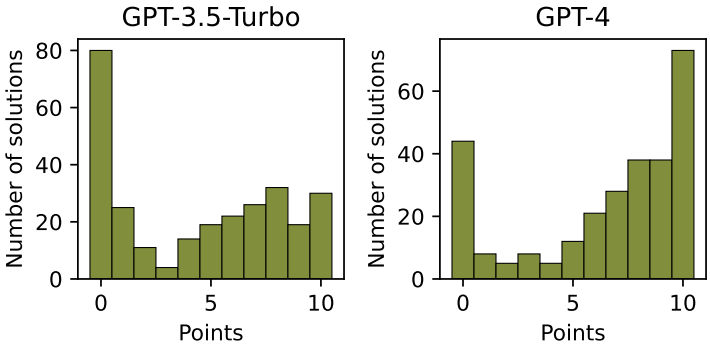

Figure 5.5: Models' performance on the algorithmic portion of our dataset

REFOK TODO: llama data

The performance in algorithmic problems is much better. On average, GPT-3.5-Turbo scored 4.35 with a median of 5. GPT-4, on the other hand, scored 6.38 on average with a median of 7.5 out of 10. An interesting trend can be spotted in Figure 5.5, as a large amount of GPT-4's solutions scored 10 points. This indicates that large language models have better capabilities to solve algorithmic tasks.

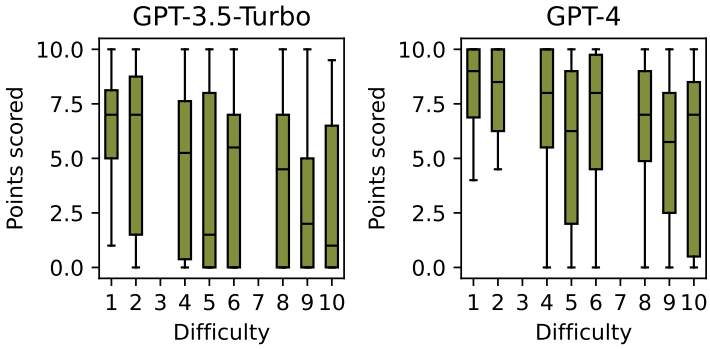

Figure 5.6: Models' performance in comparison to the difficulty when solving the algorithmic portion of our dataset

REFOK TODO: llama data

In Figure 5.6, we can once again see the relationship between scores and problem difficulty. The pattern observed in algorithmic problems is pretty different to the ones observed before. GPT-4 outperforms GPT-3.5-Turbo, but at the same time, provides pretty consistent results across the difficulty range. There is still some bias towards the easier problems, most of which can be solved by the model.

5.2 One-Shot Prompting

5.2.1 Maths Problems

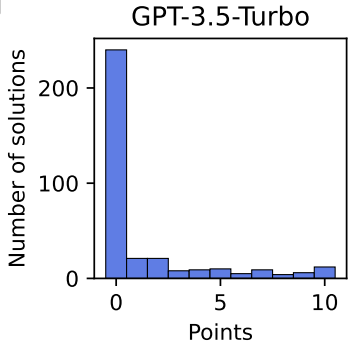

Figure 5.7: One-shot performance on the maths part of our dataset

REFOK TODO: gpt4 data, llama data

When one-shot prompting the models, GPT-3.5-Turbo scored on average 1.36 points on our maths problems. This result is 9.7% higher than our baseline measurement in zero-shot prompting. GPT-4 scored on average a 00% higher score of 0.00 points. TODO: llama results The scores received are shown on Figure 5.7.

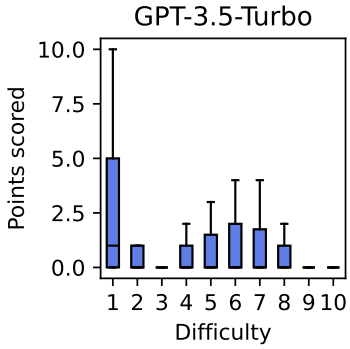

Figure 5.8: One-shot performance in comparison to the difficulty of the maths part of our dataset

REFOK TODO: gpt4 data, llama data

We observed that GPT-3.5-Turbo is seemingly more capable to solve simple problems (difficulty 1) when one-shot prompted in comparison to the zero-shot baseline. An interesting observation is the presence of a peak around difficulty 6. It should be noted that the randomly selected one-shot example was a problem with a difficulty of 8. TODO: gpt4, llama difficulty comment These differences can be seen in Figure 5.8.

5.2.2 Physics Problems

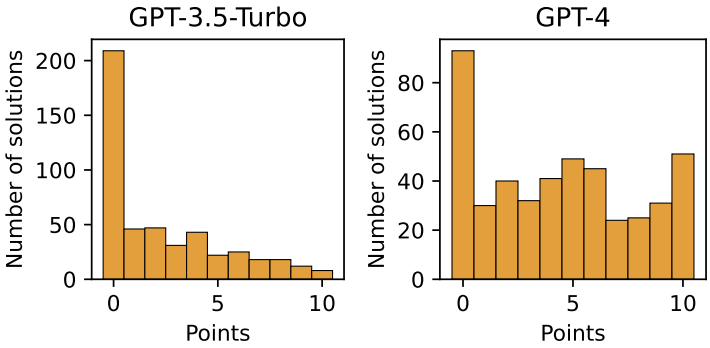

Figure 5.9: One-shot performance on the physics part of our dataset

REFOK TODO: llama data

When looking at the physics problems in our dataset, GPT-3.5-Turbo scored on average 2.3 points, which is 6.5% worse than the zero-shot baseline. Additionally, GPT-4 scored 4.38 points, or just about 5.54% better than the baseline. TODO: llama data The received scores are shown on Figure 5.9.

The median score was 1 for GPT-3.5-Turbo and 4 for GPT-4. TODO: llama data

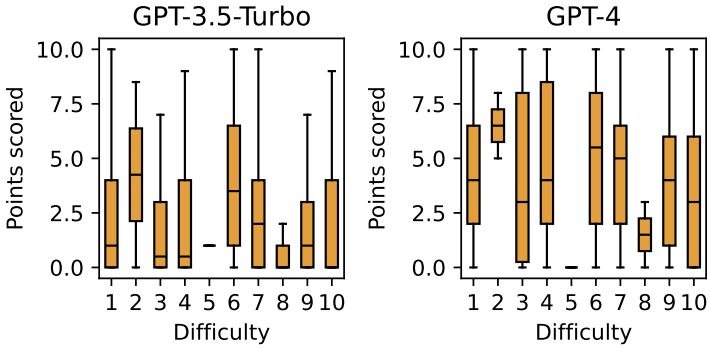

Figure 5.10: One-shot performance in comparison to the difficulty of the physics part of our dataset

REFOK TODO: llama data

For GPT-3.5-Turbo, the points scored in various difficulty levels follow the same pattern as in the zero-shot baseline. GPT-4 observes a small improvement in all difficulty levels, as shown in Figure 5.10. TODO: llama data

5.2.3 Algorithmic Problems

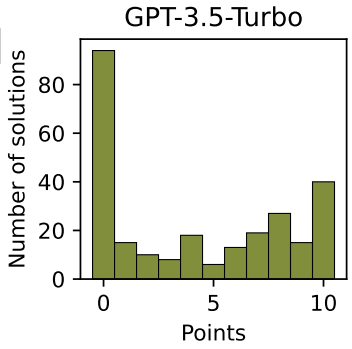

Figure 5.11: One-shot performance on the algorithmic part of our dataset

REFOK TODO: gpt4 data, llama data

Provided by our algorithmic problems, one-shot prompted GPT-3.5-Turbo scored 4.15 points, which is 4.6% worse than the zero-shot baseline. GPT-4, on the other hand, scored 0.00 on average, improving by 0.00% over the baseline. TODO: llama data The distribution of points obtained by the models is shown in Figure 5.11.

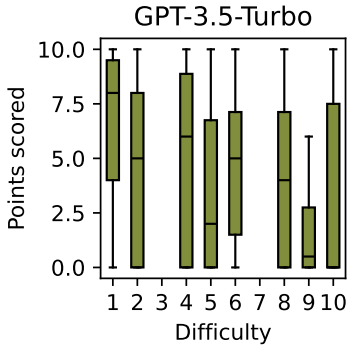

Figure 5.12: One-shot performance in comparison to the difficulty of the algorithmic part of our dataset

TODO: gpt4 data, llama data

The points scored based on difficulty shows a similar pattern to the one observed in the zero-shot prompting results. TODO: gpt4 data, llama data TODO: reference p-os-diff figure

5.3 Few-Shot Prompting

When experimenting with one-shot prompting, we experienced problems due to the long input prompts. The problems included, but were not limited to, hitting API limits and timeouts during response generation. We attempted to circumvent these issues, but unfortunately, no plausible fix was found. Due to these issues, we decided that we would not proceed with a few-shot prompting experiments, as they seemed unfeasible.

5.4 Zero-Shot Chain-Of-Thought

5.4.1 Maths Problems

TODO: waiting for experiment data

5.4.2 Physics Problems

TODO: waiting for experiment data

5.4.3 Algorithmic Problems

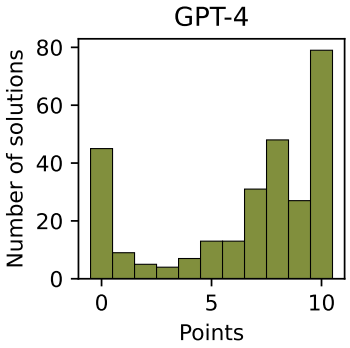

Figure 5.13: Zero-shot chain-of-though performance on the algorithmic part of our dataset

REFOK TODO: gpt35 data, llama data

When zero-shot chain-of-thought prompting GPT-3.5-Turbo, it achieves on average a score of 0.00 (with a median of 0), which is a 0.00% improvement over the baseline. Additionally, GPT-4 scores on average 6.44 points (with a median of 7.5), which is just 0.94% better than when zero-shot prompting. TODO: llama data The models' score distribution is shown in Figure 5.13.

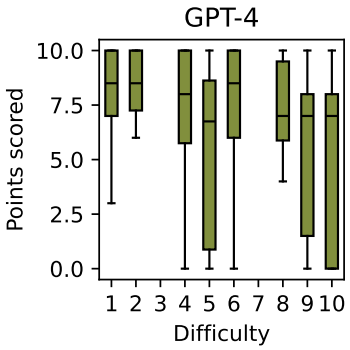

Figure 5.14: Zero-shot chain-of-though performance in comparison to the difficulty of the algorithmic part of our dataset

TODO: gpt35 data, llama data

In GPT-3.5-Turbo, we can ... . In GPT-4's results, we see a slight improvement in all difficulty levels over the baseline. TODO: gpt35 data, llama data TODO: ref p-zscot-diff figure

5.5 Generated Knowledge

5.5.1 Maths Problems

TODO: waiting for experiment data

5.5.2 Physics Problems

TODO: waiting for experiment data

5.5.3 Algorithmic Problems

TODO: waiting for experiment data

5.6 Dual-Prompt Generated Knowledge

5.6.1 Maths Problems

TODO: waiting for experiment data

5.6.2 Physics Problems

TODO: waiting for experiment data

5.6.3 Algorithmic Problems

TODO: waiting for experiment data

5.7 Least-To-Most Prompting

5.7.1 Maths Problems

TODO: waiting for experiment data

5.7.2 Physics Problems

TODO: waiting for experiment data

5.7.3 Algorithmic Problems

TODO: waiting for experiment data

5.8 Output Language

Figure 5.15 shows the difference between scores obtained when the model generated its solution in English and Slovak. In the zero-shot experiments, the models decided to output English solutions even though the problems were in Slovak in about 71.3%

of cases, with GPT-3.5-Turbo preferring English output more (89.2% of cases) while GPT-4 preferred Slovak (only 39% of solutions were in English). The models achieved an average score of 3.48 when outputting Slovak and 3.28 when outputting English.

When we look at the results per model, we see that GPT-3.5-Turbo scores on average 1.47 in Slovak and 2.69 in English. GPT-4 also excels in English solutions by scoring 4.27 on average, while scoring an average of 3.95 in Slovak.

TODO: llama

We also checked that the difficulty of problems was distributed almost evenly across both languages.

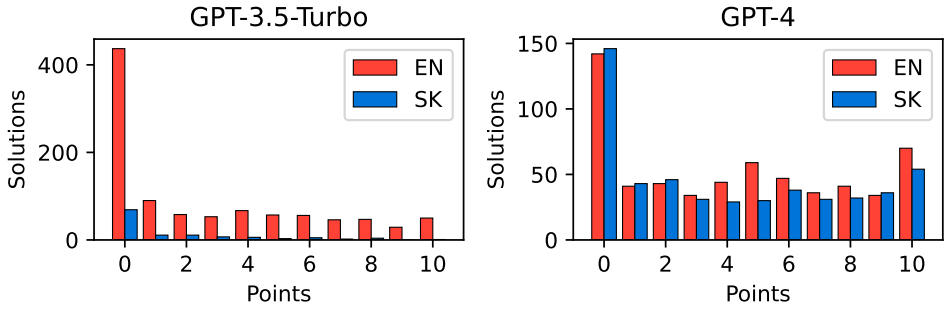

Figure 5.15: Models' performance when outputting text in Slovak and English

REFOK TODO: llama

5.9 Problem Statement Language

TODO: re-run on larger subset?

We have also experimented with translating the problem statements into English. A small subset of our maths problems ($n = 36$) was translated into English and prompted to the models.

In GPT-3.5-Turbo, there almost was no measurable difference between the scores received from English and Slovak statements. On Slovak statements, GPT-3.5-Turbo scored on average 0.60, while it scored 0.58 on English statements.

When tested with GPT-4, the average score for Slovak statement was 3.30 and 1.32 for English.

TODO: some chart?

Chapter 6

Discussion

We introduce the largest dataset of problems and solutions from Slovak high school competitions.

Large language models show promising results when solving high school competition problems in Slovak. In all parts of our dataset, GPT-4 outscored GPT-3.5-Turbo. TODO: llama The most prominent of them being our algorithmic dataset, where GPT-4 scored on average 6.38 out of 10. We explain the big gap between results in algorithmic and other problems by the fact that such problems are readily available on the internet, and thus more prevalent in the training dataset of the models, while maths and physics problems are much more scarce.

TODO: comparisons between different prompting techniques Figure 6.1

We observe much worse results than English researchers. We speculate that this is because most of the English datasets were created by scraping the web / standardized tests. The same data was most probably also used to train the models, whereas our problems may be less exposed. In addition, newer large language models are trained on training subsets of many of the mentioned problem datasets.

We also notice that translating our problems to English does not seem to have an effect on the scores. This allows us to think that our problems themselves are hard for the models, and our results are not worsened by the fact that the problems are in Slovak.

Our results show great potential for future research and experiments using large language models. We will continue exploring the abilities of such models and experiment with new ways their results can be improved.

Subset	Approach	GPT-3.5-Turbo	GPT-4	Llama 3
Maths	Zero-Shot	1.24	2.26	TODO:
	One-Shot	1.36	TODO:	TODO:
	Zero-Shot CoT	TODO:	TODO:	TODO:
	Gen. Knowledge	TODO:	TODO:	TODO:
	Dual-Prompt GK	TODO:	TODO:	TODO:
	Least-To-Most	TODO:	TODO:	TODO:
Physics	Zero-Shot	2.46	4.15	TODO:
	One-Shot	2.30	4.38	TODO:
	Zero-Shot CoT	TODO:	TODO:	TODO:
	Gen. Knowledge	TODO:	TODO:	TODO:
	Dual-Prompt GK	TODO:	TODO:	TODO:
	Least-To-Most	TODO:	TODO:	TODO:
Algorithms	Zero-Shot	4.35	6.38	TODO:
	One-Shot	4.15	TODO:	TODO:
	Zero-Shot CoT	TODO:	6.44	TODO:
	Gen. Knowledge	TODO:	TODO:	TODO:
	Dual-Prompt GK	TODO:	TODO:	TODO:
	Least-To-Most	TODO:	TODO:	TODO:

Figure 6.1: Models' results in our benchmarks

REFOK

Bibliography

[1] Chengwei Qin et al. *Is ChatGPT a General-Purpose Natural Language Processing Task Solver?* 2023. arXiv: 2302.06476 [cs.CL].

[2] OpenAI et al. *GPT-4 Technical Report*. 2024. arXiv: 2303.08774 [cs.CL].

[3] OpenAI. *Models*. https://platform.openai.com/docs/models/gpt-3-5-turbo. [Accessed 09-05-2024].

[4] AI@Meta. *Llama 3 Model Card*. https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md. [Accessed 25-05-2024]. 2024.

[5] Alec Radford et al. "Language Models are Unsupervised Multitask Learners". In: 2019. URL: https://api.semanticscholar.org/CorpusID:160025533.

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

[7] Jason Wei et al. *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. 2023. arXiv: 2201.11903 [cs.CL].

[8] Takeshi Kojima et al. *Large Language Models are Zero-Shot Reasoners*. 2023. arXiv: 2205.11916 [cs.CL].

[9] Jiacheng Liu et al. *Generated Knowledge Prompting for Commonsense Reasoning*. 2022. arXiv: 2110.08387 [cs.CL].

[10] Denny Zhou et al. *Least-to-Most Prompting Enables Complex Reasoning in Large Language Models*. 2023. arXiv: 2205.10625 [cs.AI].

[11] Subhro Roy and Dan Roth. *Solving General Arithmetic Word Problems*. 2016. arXiv: 1608.01413 [cs.CL].

[12] Yan Wang, Xiaojiang Liu, and Shuming Shi. "Deep Neural Solver for Math Word Problems". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Ed. by Martha Palmer, Rebecca Hwa, and Sebastian Riedel. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 845–854. DOI: 10.18653/v1/D17-1088. URL: https://aclanthology.org/D17-1088.

[13] Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. "Program Induction by Rationale Generation: Learning to Solve and Explain Algebraic Word Problems". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Regina Barzilay and Min-Yen Kan. Vancouver, Canada: Association for Computational Linguistics, July 2017, pp. 158–167. DOI: 10.18653/v1/P17-1015. URL: https://aclanthology.org/P17-1015.

[14] Dan Hendrycks et al. *Measuring Mathematical Problem Solving With the MATH Dataset*. 2021. arXiv: 2103.03874 [cs.LG].

[15] Karl Cobbe et al. *Training Verifiers to Solve Math Word Problems*. 2021. arXiv: 2110.14168 [cs.LG].

[16] Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. *A Diverse Corpus for Evaluating and Developing English Math Word Problem Solvers*. 2021. arXiv: 2106.15772 [cs.AI].

[17] Arkil Patel, Satwik Bhattamishra, and Navin Goyal. *Are NLP Models really able to Solve Simple Math Word Problems?* 2021. arXiv: 2103.07191 [cs.CL].

[18] Freda Shi et al. *Language Models are Multilingual Chain-of-Thought Reasoners*. 2022. arXiv: 2210.03057 [cs.CL].

[19] Tomohiro Sawada et al. *ARB: Advanced Reasoning Benchmark for Large Language Models*. 2023. arXiv: 2307.13692 [cs.CL].

[20] Aida Amini et al. "MathQA: Towards Interpretable Math Word Problem Solving with Operation-Based Formalisms". In: *CoRR* abs/1905.13319 (2019). arXiv: 1905.13319. URL: http://arxiv.org/abs/1905.13319.

[21] Gerd Kortemeyer. "Toward AI grading of student problem solutions in introductory physics: A feasibility study". In: *Physical Review Physics Education Research* 19.2 (Nov. 2023). ISSN: 2469-9896. DOI: 10.1103/physrevphyseducres.19.020163. URL: http://dx.doi.org/10.1103/PhysRevPhysEducRes.19.020163.

[22] Johannes Schneider, Bernd Schenk, Christina Niklaus, and Michaelis Vlachos. *Towards LLM-based Autograding for Short Textual Answers*. 2023. arXiv: 2309.11508 [cs.CL].

[23] Gerd Kortemeyer. *Performance of the Pre-Trained Large Language Model GPT-4 on Automated Short Answer Grading*. 2023. arXiv: 2309.09338 [cs.CL].

[24] Kabir Ahuja et al. *MEGA: Multilingual Evaluation of Generative AI*. 2023. arXiv: 2303.12528 [cs.CL].

[25] Sébastien Bubeck et al. *Sparks of Artificial General Intelligence: Early experiments with GPT-4*. 2023. arXiv: 2303.12712 [cs.CL].

[26] Lukas Blecher, Guillem Cucurull, Thomas Scialom, and Robert Stojnic. *Nougat: Neural Optical Understanding for Academic Documents*. 2023. arXiv: 2308.13418 [cs.LG].