City, University of London
MSc in Data Science
Project Dissertation
2024

Llamas in the Classroom: Investigating how the Information and Structure of Simple Maths Word Problems Affect the Accuracy of Smaller Llama Language Models

Fasih Munir

Supervised By: Tillman Weyde

Submitted At: 2nd October 2024

By submitting this work, I declare that this work is entirely my own except those parts duly identified and referenced in my submission. It complies with any specified word limits and the requirements and regulations detailed in the assessment instructions and any other relevant programme and module documentation. In submitting this work I acknowledge that I have read and understood the regulations and code regarding academic misconduct, including that relating to plagiarism, as specified in the Programme Handbook. I also acknowledge that this work will be subject to a variety of checks for academic misconduct.

Signed: *Fasih Munir*

ABSTRACT

This study investigates the performance of quantized versions of Llama 2 (7B) and Llama 3 (8B) models on simple addition and subtraction word problems. This comparison is significant as we aim to understand what about the structure of a word problem makes it challenging.

Four features are explored: information relevance, which include direct and misleading information; problem length, which is word count; number size, ranging from single to multi-digit numbers; and number of steps, which is the number of steps needed to solve the problem. Our evaluation addressed two outputs: the final answer and the extraction of relevant numbers for mathematical operation. . These outputs are generated through four types of prompts: basic prompts for direct answers, basic prompts with examples, extraction prompts for identifying relevant numbers and extraction prompts with examples. We run a series of experiments and save parsed results for evaluation. Results are compared to better understand the effect of increasing complex word problems.

The dataset (Hosseini et al., 2014), provided a foundation for investigated features. Overall, this research contributes to the understanding of how language models extract mathematical information from text, providing insights towards their difficulties which can be used to further research in improving language models.

Keywords: Small Models, Simple Word Problems, Word Problem Complexity, Prompts

Table of Contents

1. Introduction and Objectives

1.1. Problem Background and Beneficiaries

Large Language Models (LLMs) have demonstrated remarkable capabilities in a wide range of tasks, from summarising vast amounts of text (Basyal and Sanghvi, 2023) to generating complex lines of code (Jiang et al., 2024). However, despite these impressive advancements, LLMs still exhibit notable shortcomings in understanding context and relational reasoning, particularly when tasked with parsing cause-and-effect relationships or hierarchical structures (Sasaki, Watanabe, and Komanaka, 2024; Li et al., 2024; Williams and Huckle, 2024). These limitations become especially apparent in solving mathematical word problems, where models must comprehend and process the information presented in natural language to perform arithmetic operations like addition or subtraction.

Solving maths word problems is a test of an LLM's contextual understanding and reasoning ability, as these problems encapsulate a variety of challenges. They involve natural language processing complexities, such as understanding actions, maintaining context across multiple steps, and applying real-world knowledge (Srivatsa and Kochmar, 2024). Furthermore, solving word problems accurately requires models to not only identify and focus on relevant information but also understand and apply the correct operations based on the problem structure. This makes word problems more complex than simpler language generation tasks.

To address these shortcomings, researchers have developed curated benchmark datasets, such as the GSM8K dataset (Cobbe et al., 2021), specifically designed to test and improve LLMs' arithmetic reasoning abilities. Additionally, several approaches have been employed to improve LLMs' performance in maths word problems. For instance, researchers have integrated external equation solvers to enhance output coherence (He-Yueya et al., 2023), scaled model training to improve arithmetic reasoning (Chowdhery et al., 2022) and used efficient prompting techniques to elicit better responses from models (Wei et al., 2022).

While much of the research focuses on improving LLM performance directly, it is equally important to understand the structural elements of word problems that cause models to fail. Identifying these structural issues can allow researchers to better target improvements. Prior work has explored individual factors such as information relevance (Shi et al., 2023) and question length (Xu et al., 2024), yet these studies

overwhelmingly focus on large LLMs rather than their smaller counterparts, such as the widely-popular 7B parameter models found on Hugging Face (Dey et al., 2023).
As LLMs become increasingly integrated into everyday life and devices, including smartphones that incorporate machine learning features (Liu et al., 2024), the focus on smaller models becomes more important. Small LLMs are more likely to be deployed in resource-constrained environments, such as mobile devices, due to their lower computational requirements and understanding if they will be good enough from an efficiency versus cost point of view is important. Yet, research on smaller model reasoning capabilities, particularly in simple arithmetic tasks, remains scarce. Large models can solve simple word problems but are they able to solve them because they truly understand how to solve the problem or because they have enough parameters to recall patterns?

This paper addresses this gap by focusing on simple maths problems involving only addition and subtraction, allowing for a controlled examination of small LLM reasoning abilities without the added complexity of advanced operations like multiplication or division. Understanding how these models process the structure of such problems can highlight underlying reasoning limitations, which may not be visible when models are tasked with more complicated operations or when tested on models that are sufficiently large.

We suspect our research will benefit several key groups:

- Machine Learning Researchers: By exploring how small LLMs perform in basic arithmetic reasoning tasks, this research will provide valuable insights into areas where reasoning capabilities in model architectures can be refined. Researchers can use these findings to develop new strategies for improving the reasoning performance of small models.

- Educators: Small LLMs will be useful in resource-limited settings, such as classrooms where computational power may be restricted. This research could lead to improvements in model reasoning ability, making them important tools for tutoring and education. Even in elementary level teaching, maths teachers can harness these tools to provide more children with an understanding of the steps involved in solving word problems.

- Companies and Startups: Companies interested in deploying LLMs for real-world applications but constrained by the high computational cost of large models will benefit from insights into improving small models. If smaller LLMs can be optimised for basic reasoning tasks like arithmetic, they can be used in a wider range of products, from inventory management to accounting.

Ultimately, the research aims to answer the question:

How Does the Structure of Simple Addition and Subtraction Word Problems Affect the Accuracy and Reasoning Ability of Smaller Large Language Models?

1.2. Objectives

To answer the research question, a few objectives need to be completed. We will conduct all experiments on Llama 2 7B and Llama 3 8B.

- **Assess the Word Problem Structure to Understand What Affects Accuracy**: We will focus on identifying specific structural elements of word problems that cause models to fail or succeed. Understanding these elements allows for more targeted LLM improvements as opposed to simply increasing training data.
  Criterion: Experiments set up and conducted, results reported and analysed.
- **Compare How the Word Problem Structure Affects Performance Between Asking for a Direct Answer Versus Asking for the Steps**: We will explore whether smaller LLMs are better at producing direct answers or extracting the steps required to solve a problem. Comparing these two approaches will help uncover if there is an understanding of what steps are needed to solve the answer as opposed to being able to predict a pattern for a direct answer.
  Criterion: Experiments set up and conducted, results reported and analysed.
- **Understand How Performance Can Improve on Solving Simple Addition and Subtraction Word Problems**: We will examine the impact of different prompting structures and additionally the impact of generational advancement within an LLM family. These points can provide insights on how to improve smaller LLMs.

Criterion: Experiments set up and conducted, results reported and analysed.

- **Observe Whether Smaller Models Are Good Enough in Their Reasoning Ability**: We will assess the model performance on our designed experiments to understand how well the models do. We will additionally try and compare scores to larger models tested on different benchmarks to get an idea of whether these models are "good enough".
  Criterion: Experiments set up and conducted, results reported and analysed.

1.3. Methods for Completing Objectives

Assess the Word Problem Structure to Understand What Affects Accuracy

- **Dataset Selection**: A dataset containing simple addition and subtraction word problems will be chosen. To avoid training leaks, we will select a dataset that is unknown or less commonly used for training large models.
- **Feature Engineering**: We will engineer features similar to those used in other research studies on LLMs and word problems, such as question length, number of steps, information irrelevance, and the size of the numbers involved. This will help link comparisons.

Compare How the Word Problem Structure Affects Performance Between Asking for a Direct Answer and Asking for the Steps

- **Prompting Structure**: We will create two distinct prompts. One that asks the LLMs to provide direct answers, and another that asks them to extract the steps required to solve the problems (the numbers and their respective operations). LLMs will be prompted within this structure to output their answer in a specific format.
- **Output Parsing**: A parsing mechanism will be designed to extract and assess the parsing of the solution for the specified format in the prompt.
- **Step Verification**: To ensure that the parsed steps are correct, we will develop a validation process to determine if the numbers and respective operations are indeed the ones required to solve the problem.

Understand How Performance Can Improve on Solving Simple Addition and Subtraction Word Problems

- **Generational Comparison**: We will assess how performance varies between Llama 2 and Llama 3. This will help us understand if improvements in generational models result in better performance on maths word problems.
- **Zero-Shot vs. Few-Shot Prompting**: We will experiment with both zero-shot and few-shot (using 3 examples) prompting techniques to see how much examples improve model performance.

Observe Whether Smaller Models Are Good Enough in Their Reasoning Ability

- **Performance Assessment**: Outputs will be compared against the larger model counterparts to determine whether small model performance is similar. While the dataset used will not be similar to the counterparts, by using similar features we hope to at least begin a conversation around performance.

1.4. Changes From Research Proposal

Initially, our research aimed to improve the output of language models by incorporating knowledge graphs. LLMs have several known weaknesses, and we began exploring how knowledge graphs could address one of them which included managing context and hierarchies. We wanted to assess this on Dyck words. Dyck words involve large, repeated strings of highly specific knowledge, requiring models to manage context and understand relationships. Early tests showed that even the most advanced large models struggled to consistently manage context. For example, from our initial proposal testing, when prompting GPT-4 to define the rules of Dyck words it provided the output "Invalid Dyck Word: *XXYYXY* or (())()". This is an incorrect output as the word it generated is in fact a valid Dyck word as it consists of parentheses that are correctly balanced.

However, during our literature review, we encountered a more directly relevant area of weakness in LLMs: mathematical information extraction and reasoning. This caught our attention as we realised a common thread between both Dyck words and maths word problems. Both require LLMs to reason through complex sequences. As a result, we decided to shift our focus from forcing a solution through

knowledge graphs to first understanding what is fundamentally problematic for LLMs in this reasoning space. By identifying these challenges, we could then later propose solutions, including knowledge graphs or other methods, to improve LLM outputs.

This pivot led us to concentrate specifically on maths word problems, where we chose to focus on small LLMs and simple arithmetic problems (addition and subtraction). Our literature review revealed that while there is substantial research on large models, much of it isolates individual features like information irrelevance and focuses on the solution directly. Much of the literature we read did not look at the same kind of research but through the lens of the steps required to solve the problem. Additionally, most research focuses on large models, leaving a gap in knowledge regarding how smaller, resource-efficient models manage these tasks. This is particularly important for smaller, computationally limited devices.

By working with small LLMs and simple maths problems, we aim to highlight the underlying reasoning abilities of these models in a controlled setting, without introducing unnecessary complexity from more advanced mathematics. Comparing two small models from the same family also allows us to investigate whether generational improvements and increased training data actually enhance models or if other methods should be explored. Even understanding how small models fail can provide valuable insights for optimization.

1.5. Work Plan

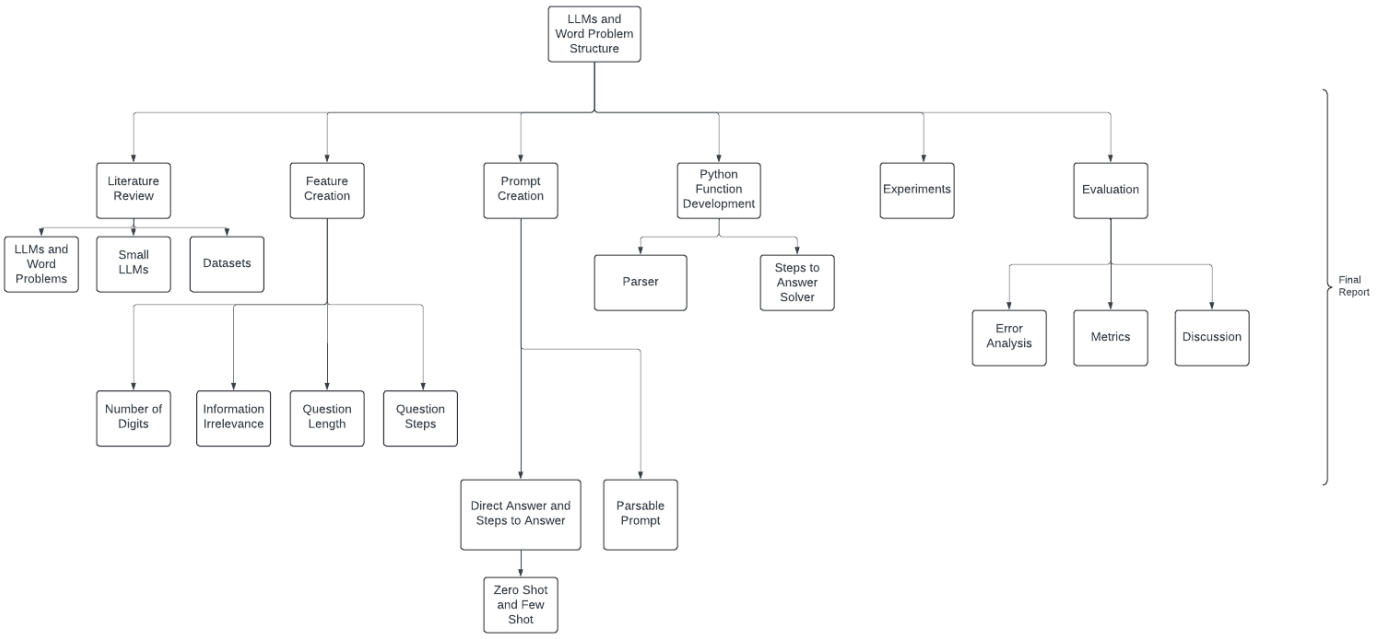


Figure 01 - Updated Work Plan

Figure 01 above details the updated work plan for this research as detailed in section 1.4. Literature Review and Feature Creation steps cover Objective 1. The Prompt Creation and Python Functions cover Objective 2. Objectives 3 and 4 are covered by Experiments and Evaluation.

1.6. Report Structure

Chapter 1 provides an introduction to the problem, highlighting our four objectives and the methods to solve them.

Chapter 2 provides context on the current state of small LLMs, why LLMs struggle with maths word problems, the ways in which researchers have tried to improve these outputs and a review of the Llama 2 and 3 models.

Chapter 3 describes in detail the methods employed to complete the objectives. These include sourcing and engineering a dataset, creating a set of prompts, understanding the parameters of the models we want to use, creating a way to ensure reasoning steps are correct, creating a way to parse the right answer from the output generated and finally detailing the experimentation pipeline as a whole.

Chapter 4 explains the results of the experiments and focuses on presenting them clearly. It highlights key observations around the reasoning abilities of the used models.

Chapter 5 discusses these results in comparison to the objectives and in the context of the literature reviewed.

Chapter 6 evaluates the project as a whole and recommends areas for further work.

2. Context

2.1. The Current State of LLMs

2.1.1. Large Language Models Today

Large Language Models (LLMs) have rapidly evolved in the past few years, largely due to the introduction of the transformer architecture by Vaswani et al. (2017). A key feature of this architecture is the attention mechanism, which allows models to handle long contexts more efficiently and effectively compared to earlier architectures like recurrent or convolutional neural networks. This innovation has enabled LLMs to perform more complex tasks and manage sequences more effectively.

These models come in a wide variety of sizes, ranging from millions to trillions of parameters, and are available in both their raw forms or fine-tuned for specific tasks. LLMs are being deployed for applications such as translation, conversational agents, and code generation (Naveed et al., 2023). Moreover, many of these models are becoming increasingly integrated into our daily lives, with increasing efforts to integrate them on small devices like mobile phones (Liu et al., 2024). This is reflective of their expanding role in business and our lives.

Due to their generalizability across tasks, continuous improvements in LLMs have become a significant area of research. These improvements have been achieved through several methods, including fine-tuning for specific use cases (He et al., 2022), increasing the size of training datasets to enhance applicability (Xue et al., 2021) and quantization, which reduces the precision of the numbers used to store model weights and parameters, thereby making models more efficient on resource constrained hardware (Tao et al., 2022). Additionally, prompt engineering has emerged as an effective method to improve model outputs without the need for extensive training or fine-tuning (Si et al., 2023).

Prompting and quantization are gaining prominence due to the high cost of training larger models. With the increasing integration of language models into everyday devices, such as smartphones, the need to improve smaller models becomes essential.

2.1.2. Smaller Large Language Models

Smaller large language models (LLMs) are generally considered to be models with fewer than 10 billion parameters and when fine-tuned or used with the right prompting strategies they can exhibit improved performance (Fu et al., 2023). These models are attractive due to their efficiency and resource-friendly nature, making them suitable for low resource deployment.

This idea is shown well through the small LLM, Paramanu-Ganita, a 208 million parameter model fine-tuned for mathematical reasoning. Despite its much smaller size, Paramanu-Ganita outperforms larger small models such as Llama 2-7B and Falcon 7B in mathematical tasks (Niyogi and Bhattacharya, 2024). This highlights the potential of smaller models when they are tuned specifically for particular tasks.

Another way to enhance the performance of small models is through prompt engineering. Zhang, Yuan, and Avestimehr (2024) show that using chain-of-thought prompting (Wei et al., 2022), which guides the model through a step-by-step reasoning process, can improve the results of small models like Mistral 7B. By including clear, direct examples in the prompt, small LLMs are able to better understand the task objectives and generate more accurate outputs.

Popular small models include Llama 2-7B (Touvron et al., 2023), Falcon 7B (Almazrouei et al., 2023), and Mistral 7B (Jiang et al., 2023), each of which has seen wide use in a variety of tasks. However, the performance of these models varies significantly depending on the task at hand. For instance, Mistral 7B achieves 83% accuracy on the GSM8K dataset, which is designed to evaluate mathematical reasoning (Li et al., 2024). In contrast, Falcon 7B and Llama 2-7B perform less consistently in common-sense reasoning tasks, scoring in the 70s on the HellaSwag dataset, while both Llama and Mistral models tend to underperform in tasks related to human-evaluated coding ability, with scores in the 30s or less (Minaee et al., 2024).

Although smaller models lack the brute force of larger models, they offer significant advantages in terms of efficiency and deployment flexibility.

2.1.3. Quantized Models

Quantization is a method used to reduce the size of large language models by lowering the memory they require to hold their weights. This is achieved by reducing the precision of the numbers that represent the model's weights, converting them from high-precision formats like FP32 (32-bit floating-point) to lower-precision formats such as FP16 or FP4. This makes models faster and more memory-efficient at the cost of some accuracy (Gholami et al., 2021).

A simple way to understand this is by looking at the memory savings achieved through quantization. For example, if a model using FP32 precision takes 100GB of memory to store its weights, reducing it to FP16 would cut the memory requirement in half, down to 50GB. Further reducing it to FP8 would bring it down to 25GB. Each reduction in precision means that the weights take half the space, just like image compression reduces pixel size to make image files smaller but at the cost of image quality.

There are several methods of quantization, such as dynamic quantization, where only activations are quantized during inference (Baras et al., 2023). However, an effective technique is QLoRA (Quantized Low-Rank Adaptation), which has gained popularity for its ability to reduce models down to FP4 precision while still maintaining performance comparable to FP16 (Dettmers et al., 2023). QLoRA achieves this efficiently by only updating a subset of the most important weight matrices, rather than updating entire model weights.

Jin et al. (2024) highlight that LLMs can be quantized down to FP4 without losing significant performance. While it is technically possible to go even lower in precision, it is generally not recommended, as it tends to result in substantial drops in performance.

The work on 4-bit quantization has given us confidence to use this process to further test the limits of small models.

2.1.4. LLMs Solving Mathematics and Word Problems

While large language models (LLMs) have demonstrated strong performance across a wide range of tasks, they often still struggle with relatively simple maths word problems. Take for example the following question.

"*Fred went to 36 basketball games this year but missed 35. He went to 11 games last year. How many basketball games did Fred go to in total?*" (Hosseini et al., 2014)

This is a relatively straightforward question which is simple for humans, but can confuse LLMs. The confusion could be because of the complex and intertwined processes needed for solving this type of

problem. It would involve language comprehension, focusing on relevant numbers, understanding the required mathematical operations and then executing those operations.

Researchers have explored several methods to improve the arithmetic abilities of LLMs, especially for solving maths word problems. These approaches include, increasing model size or training data, fine tuning models for specific tasks, separating calculation from natural language understanding and improving outputs through prompting.

One of the most common approaches has been to increase the parameter count and the scale of training data. For instance, models have evolved from GPT-3 with 175 billion parameters (Brown et al., 2020), to PaLM with 540 billion parameters (Chowdhery et al., 2022), and most recently to GPT-4 (OpenAI, 2023), which is speculated to have over a trillion parameters (Howarth, 2024). Each new iteration has shown performance improvements across a variety of benchmarks. However, even the developers of GPT-4 note that despite its vast size, it still suffers from limitations such as hallucinations and limited context windows (OpenAI, 2023).

However, Chowdhery et al (2022) have observed that training scale improvements have likely not yet plateaued. It then begs the question that, are these models actually improving in arithmetic reasoning or are they just getting better at predicting what the answer should be? One can argue that this may be in fact how humans learn, given that the more practice we do on similar questions, the more likely we are to get new questions correct (Haith and Krakauer, 2018) and so we might not want to worry about this distinction.

But, if we do choose to avoid this question, then it does become more important to understand the influence of training data, the nuances in the language used, the kinds of questions available and the ways in which the questions might create failure modes which can then be used for future improvements.

Prompting has become an important area for improving model performance without additional training or fine tuning. For instance, as mentioned earlier, chain of thought prompting, introduced by Wei et al. (2022), encourages models to generate step-by-step reasoning before producing an answer. In their experiments with PaLM 540B, this technique significantly improved performance, even outperforming fine-tuned versions of smaller models like GPT-3. However, the effectiveness of prompting scales with model size, meaning that larger models benefit more from such techniques.

Fine-tuning is another powerful method to enhance model performance, particularly for smaller models. Instead of retraining a model from scratch, fine-tuning exposes the model to specific data, enabling it to

leverage existing generalisation abilities while learning the nuances of a particular task. For example, Yang et al. (2023) showed that a fine-tuned GLM-10B model could outperform GPT-4 on maths problems demonstrating that smaller models, when fine-tuned, can surpass larger models.

Combining fine-tuning with chain of thought prompting has also proven effective. Nye et al. (2021) introduced the concept of a scratchpad, where the model is trained to produce intermediate steps for a maths problem rather than simply outputting the final answer. Within the fine tuning dataset, these intermediate steps are shown as examples resulting in a model that is able to accurately output the steps. This method significantly improved accuracy, increasing performance on polynomial evaluation tasks from 31.8% to 50.7% using 8B parameter models.

Some researchers have worked with the limitations of LLMs by integrating external tools to support areas where models fall short. For example, He-Yueya et al. (2023) used few shot prompting to extract maths equations rather than final answers, which were then solved using external tools like SymPy, a Python library for symbolic computation. This method ensures high quality step-by-step computations, using external systems to complement reasoning. This approach has partly inspired ours, described further in the Methods chapter of this research.

Similarly, Zhang et al. (2023) developed an intermediate model that predicts the operations required to solve a maths word problem, which the LLM uses to generate the final answer. Their aim was to improve interpretability of the outputs but this additionally puts an increasing focus not on the LLM to answer correctly but on the intermediate model to correctly predict the required steps which is one additional point of possible failure.

While a lot of work has been on improving the outputs, relatively little research has focused on why maths word problems are difficult for LLMs. Research by Shi et al. (2023) and Anantheswaran et al. (2024) has shown that the presence of irrelevant information in a maths word problem can cause models to perform poorly. Irrelevant information, such as unnecessary details like Fred missing 35 games (mentioned at the start of this section) can lead to errors. It should also be noted that their work was done on models much larger than 7B parameter models.

Similarly, Xu et al (2024), found that the GPT-3 family of models were negatively affected by increasing the narrative of the question and then worked towards creating a more intuitive prompt that can help solve the longer narrative questions better. This prompt involved using words specifically designed to

understand the final goal of the problem and while we did not use the exact prompt structure we did employ a similar thinking in our final prompts, asking models to focus strictly on the final ask of the question.

These papers inspired us to engineer irrelevance and length features within our chosen dataset as well.

Most of the work done on understanding what makes a word problem difficult has been done using the larger sized models. Many of the papers are also strictly observing the final answer as opposed to the steps. And while researchers are looking to improve the sequence of steps to solve word problems, the papers have not looked at why the structure of a word problem would cause the sequence of steps to be wrong. Given that smaller LLMs are becoming more popular and integrating into our everyday devices, it becomes important to understand how smaller, possibly even quantized models, process word problems. Using larger models, given their larger parameter size, can possibly mask simple problems just by nature of having a deeper network that can recall more patterns.

2.2. The Llama Models

The Llama family of models is a series of open-source LLMs developed by Meta. These models are freely available to use. They have become popular due to their competitive performance and open source nature.

Llama 2, introduced in 2023 by Touvron et al., is available in four sizes: 7B, 13B, 34B, and 70B parameters. Llama 2 models are trained on 2 trillion tokens of publicly available data. Meta placed emphasis on the quality of data rather than volume and avoided relying on third-party data annotations. While this decision led to a smaller training dataset, it significantly enhanced the quality of the data. As a result, the 7B version of Llama 2 outperforms Falcon 7B on most benchmarks, including mathematical and common sense tasks. The 70B version of Llama 2 was shown to surpass all other open-source models in terms of performance.

Llama 3, introduced in 2024 by Dubey et al., is available in 8B, 70B, and 405B parameter sizes. The Llama 3 models were trained on a much larger dataset of 15 trillion tokens, doubling down on the emphasis on data cleaning and rigorous annotation. This increase in data volume and quality, coupled with a longer training period (longer than compute optimal for the smaller models), has enabled Llama 3 models to outperform earlier versions and other competitive models such as Mistral 7B.

Notably, the Llama 3 8B model consistently outperforms Mistral 7B across all tasks. The largest model, Llama 3 405B, performs comparably to GPT-4, showing substantial improvement.

However, Dubey et al (2024) note that in specific tasks, particularly mathematical tasks, performance varies between adversarial and non-adversarial benchmarks. The performance of Llama 3, on adversarial tasks, even post-training, remains lower than on non-adversarial tasks, which suggests that the performance could be sensitive to the training data, particularly when it comes to irrelevant information.

Huang et al. (2024) have further conducted experiments on quantizing Llama 3 down to 4 bits using LoRA and QLoRA, followed by fine-tuning on the Alpaca dataset. Previous Llama models, when fine-tuned post quantization are able to recover some of the performance lost due to quantization, Llama 3 on the other hand was unable to benefit from this. For instance, in the STEM category, performance dropped from 55.3 to 49.3 after quantization.

The authors hypothesise that the Llama 3 full precision is already highly optimised due to its clean and vast dataset and the quantization introduces errors that fine tuning cannot help recover. Dubey et al (2024) also note that Llama 405B is sensitive to certain types of quantization and only reference quantization to FP8 in their paper. Despite these challenges, Huang et al (2024) note that quantized 4 bit Llama 3 8B models still outperform previous models with the same quantization.

3. Methods

3.1. Data Gathering and Manipulation

We initially examined two distinct datasets for potential use from the many that are available (Ahn et al, 2024). The first dataset was GSM8K, introduced by Cobbe et al. (2021). This dataset has been utilised in numerous studies to train LLMs, particularly for word problems in arithmetic. It has also been referenced in many other studies such as by Shi et al. (2023), which used a variation of GSM8K to explore more complex mathematical reasoning. The second dataset was sourced from an earlier study by Hosseini et al. (2014), which compiled word problems from various online sources.

3.1.1. Rationale for Dataset Selection

While GSM8K contains data on a variety of arithmetic operations such as addition, subtraction, multiplication, and division, the focus of this study was to explore the challenges smaller LLMs face in solving simpler problems, specifically addition and subtraction. The GSM8K dataset and its variation in Shi et al. (2023) contain problems designed for training LLMs, often involving longer and complex word problems. For instance, questions may include multiple arithmetic operations and large quantities of information such as in Figure 02 below.

Figure 02 - Taken from Cobbe et al. (2021)

Problem: Mrs. Lim milks her cows twice a day. Yesterday morning, she got 68 gallons of milk and in the evening, she got 82 gallons. This morning, she got 18 gallons fewer than she had yesterday morning. After selling some gallons of milk in the afternoon, Mrs. Lim has only 24 gallons left. How much was her revenue for the milk if each gallon costs $3.50?
Mrs. Lim got 68 gallons - 18 gallons = <<68-18=50>>50 gallons this morning.
So she was able to get a total of 68 gallons + 82 gallons + 50 gallons = <<68+82+50=200>>200 gallons.
She was able to sell 200 gallons - 24 gallons = <<200-24=176>>176 gallons.
Thus, her total revenue for the milk is $3.50/gallon x 176 gallons = $<<3.50*176=616>>616.
Final Answer: 616

Given that our research aimed to examine simpler arithmetic problems, we decided to limit our focus to addition and subtraction. These operations are conceptually simpler compared to multiplication and division, and we hypothesised that understanding the basic challenges for smaller LLMs would provide a clearer picture of model performance on fundamental tasks such as keeping inventory count. However, we chose not to extract a subset of GSM8K's addition and subtraction problems because it is widely available online and has been extensively used in training various models, increasing the risk of data leakage. As

the Llama family of models is open-source and trained on publicly available data, there was a possibility that these models had already seen some portion of GSM8K during their pre-training.

To mitigate the risk of data leakage, we opted to use the dataset from Hosseini et al. (2014). This dataset consists of 395 word problems aimed at students in grades 3 to 5 and exclusively contains addition and subtraction problems in English language. It was originally used to test a novel development of natural language based problem solving, unrelated to large language models. The dataset was sourced from two distinct online websites that randomly generate arithmetic questions. Since this dataset is not widely available online, it is less likely to have been included in the Llama models' training data, thus minimising the risk of data leaks.

3.1.2. Data Format and Preprocessing

The Hosseini et al. (2014) dataset was provided in a text file format, containing only the word problems themselves. Although the original study included answer and equation files, these outputs were generated by the solution described in their paper and were therefore not directly applicable to our research. We manually processed the dataset by parsing the text file in Python, using the "?" symbol as a delimiter to separate each question. We then applied basic preprocessing techniques to clean the data, such as removing extra white spaces and correcting punctuation using string substitutions and the strip() function.

Once the questions were extracted and cleaned, we manually created the corresponding equations and answers to ensure the dataset was complete and accurate. This was done using a Google Sheet, where we generated answers using the “+” and “-” operators in one column and used the FORMULATEXT function in another column to create the equation associated with each problem.

3.1.3. Feature Engineering

To better understand the factors contributing to word problem complexity and how they influence LLM performance, we engineered several features from the dataset. These features were designed to capture various aspects of the word problems that could potentially affect model accuracy:

- Question Length: This feature represents the total number of characters in the word problem and is divided into three categories: less than 100, between 100 and 199, and greater than 200. The length of the question might affect the model's ability to parse and retain relevant information. Other work done by Srivatsa and Kochmar (2024) and Xu et al (2024).

- Number of Digits: This feature categorises the largest number present in the word problem by its number of digits. For instance, a question mentioning 37 seashells and 5 seashells, would be categorised as a 2-digit problem. This feature helps determine whether larger numbers, which generally require more human cognitive load, impact the model's performance. This is inspired by question length as each number is an extra token hence a longer context.
- Information Irrelevance: This feature categorises the word problem based on how much irrelevant information is included. This feature was designed to assess whether LLMs struggle more when faced with extraneous details in the problem. Other work done by Shi et al (2023) and Anantheswaran et al (2024).

  Please note that the categorization was created by us and can be subjective. To mitigate subjectivity, we identified a strict set of rules that we aimed to adhere to while marking each question. Our findings for this feature should be interpreted within this context.

  Problems were labelled as follows and examples are shown in Figure 03:

  - 0: Contains direct information. Every number and entity is involved.
  - 1: Contains extra content but no extra figures or steps.
  - 2: Includes irrelevant numbers that are not part of the solution.

Information Irrelevance	Question	Explanation
0	Alyssa has 37 blue balloons, Sandy has 28 blue balloons and Sally has 39 blue balloons. How many blue balloons do they have in all?	Purely talking about blue balloons and their count. Every number and entity is involved.
1	Sara's high school played 12 basketball games this year. The team won most of their games. They were defeated during 4 games. How many games did they win?	Mentioning that most games were won is extra content, however there are no extra numbers or steps
2	Sara picked 35 pears and 27 apples from the orchard. She gave 28 pears to Dan. How many pears does Sara have?	Picking 27 apples is not relevant

Figure 03 - Irrelevance Examples

- Question Steps: This feature counts the number of arithmetic steps required to solve the problem. For example, if two numbers are simply added, the problem requires one step. However, if

multiple operations are required (e.g., adding two numbers and then subtracting another), the number of steps increases. This feature helps evaluate whether multi-step problems pose greater challenges for the models. Inspired by Wei et al (2022) who introduced chain of thought prompting to enable more complex reasoning that involves multiple steps.

3.1.4. Final Dataset

After processing and feature engineering the dataset from Hosseini et al. (2014), we obtained a clean and transformed dataset of 395 word problems with 5 features that we would analyse to determine the sources of complexity for LLMs. There were small variations in question templates but not a lot of difference. These questions were not designed to test LLMs but to test human students. This dataset provides a controlled environment, and our contribution in adapting the dataset for this research, to study the effects of problem structure on the Llama models' ability to solve simple addition and subtraction problems. Refer to Appendix B for the full data set.

3.2. Models and Hardware Review

In this study, we worked with limited hardware that had access to GPUs capable of running large language models (LLMs). Specifically, we chose to use Google Colab, a widely available and free platform that provides access to T4 GPUs in its free tier, allowing us to run small models around the 7 billion parameter (7B) mark. The choice of using small models was influenced by their lower memory requirements, as their parameters and respective weights are smaller, making them more efficient to load and run on limited hardware. While our initial testing was done on free T4 GPUs, our experimentation pipeline used bigger A100 GPUs provided by paid versions of Google Colab just to speed up our experimentation and avoid any possible bottlenecks such as timeouts or out of memory issues.

We leveraged the Hugging Face Transformers library (Wolf et al., 2020) to test a variety of small models, including Falcon 7B, GPT-2, and Llama 2 7B, amongst others. This further enabled us to adjust key parameters such as max tokens, temperature, top-p, and repetition penalty to control the performance of the chosen models. Additionally, during our testing we were able to access the logits and probabilities of the generated tokens, which can be used for metric calculations like perplexity.

In our testing, despite being able to run full 7B models on Google Colab free tier, our focus on running experiments under constrained hardware conditions led us to opt for quantization to reduce model size further. Quantization is a technique that compresses the model, reducing memory usage and computational power while increasing inference speed (Gholami et al., 2021). We specifically used 4-bit quantization, which was inspired by the QLoRA (Quantized Low-Rank Adaptation) method introduced by Dettmers et al. (2023). QLoRA allows for efficient fine-tuning by quantizing the model to 4-bit precision while maintaining performance comparable to traditional 16-bit methods. The key innovation in QLoRA is the use of LoRA (Low-Rank Adaptation) (Hu et al., 2021), which introduces small, low-rank matrices that are fine-tuned while leaving the original model weights untouched, drastically improving efficiency. This works as we are not updating the entire original matrix of weights.

While there are many quantization techniques available (Gholami et al., 2021), the Hugging Face Transformers library offers seamless integration with QLoRA, making it an optimal choice for our resource-constrained environment (Hugging Face, 2023)

Although we tested multiple small models, including Falcon 7B and GPT-2, we ultimately decided to focus on Llama 2 7B (chat-hf) and Llama 3 8B (instruct). As noted in the Context chapter, the Llama family of models is developed by Meta, a leading technology company known for its significant investments in language models through computational resources, energy, and research. By comparing two small models from the same family, we aimed to assess whether the advancements made in Llama 3 8B truly improved performance in practical, resource-constrained applications. The decision to compare models from the same family provided additional benefits, such as controlling for factors like general architecture, training approach, and external influences like research priorities and budgets. This approach allowed us to better isolate why performance improvements might occur between Llama 2 and Llama 3, rather than being influenced by differences between architectures or optimizations.

For both Llama models, we maintained consistent settings across key parameters to ensure a fair comparison. We used the following parameters: max_tokens = 50/150, temperature = 0.6, top_p = 0.8, and repetition_penalty = 1.0. These settings were based on established definitions and guidelines (Wang et al., 2024) and our extensive testing to ensure effective output parsing. The temperature parameter was set to 0.6 to maintain moderate control over randomness, given that we were solving word problems and wanted the models to provide the most likely correct answers rather than overly creative outputs. Additionally, top-p was set to 0.8, which limits token selection to the top 80% of the most likely tokens by cumulative probability, effectively reducing irrelevant words. The repetition penalty was kept at 1.0,

meaning no penalty was applied for repeated tokens. This was critical for our word problem use case, where numbers may need to be repeated in the solution, and we did not want the model to avoid generating them because of an arbitrary penalty. The max_tokens parameter was set to either 50 or 150, depending on the complexity of the question. For questions with a larger number of digits (e.g., 17 or 18 digits), the max_tokens was increased to 150 to accommodate the generation of longer answers. This parameter limits how many new tokens the model generates beyond the input tokens, ensuring the responses are concise and avoid unnecessary verbosity, which could otherwise consume additional computational resources and make output parsing more difficult. During our testing we had noticed a tendency for the Llama models to begin creating new examples or filling in the token space with irrelevant text which was increasing compute time, despite prompts to stop. Llama 3 especially was guilty of this.

For the outputs, we only retained the generated text and chose not to store the logits and probabilities, as calculating metrics from these added around 2 to 3 minutes of compute time per query (similar to max tokens). By not storing these additional data points, we were able to save significant time across our experiments.

3.3. Ensuring Correct Reasoning

We wanted to explore why smaller LLMs struggle with correctly solving simple mathematical word problems beyond just answering the question. Given this, we designed experiments to extract two outputs from the LLMs: the direct answer to the problem, and a separate output consisting of just the numbers and their associated operations (e.g., addition or subtraction), which can then be used for calculations outside of the LLM.

This approach stemmed from the idea that it may be easier for the LLM to extract relevant numbers from a word problem than it is to predict the final answer. If extracting numbers proves to be easier for the model, we needed a way to ensure those numbers were accurate and that the correct operations were applied. In essence, we needed a reliable mechanism for calculation, separating the prediction task from the computation task. Pallagani et al. (2024) has shown various ways of improving the output of LLMs, one of which includes external augmentation methods. Rather than relying solely on the model's internal logic, augmenting the LLM by integrating it into a Python environment would allow us to control certain aspects of the process, such as how calculations were performed. Qian et al (2022) also explored callable programmes that can handle computation outside of the LLM architecture. He-Yueya et al (2023) also

used a similar approach by using few shot prompting to extract the equation and then running the equation through an external solver to get the correct answer. This method worked well with our pipeline because it gave us more control over how the LLM outputs were enhanced, as opposed to using pre-built connections or relying entirely on prediction capabilities.

This approach is similar to work done by Guellec et al. (2024) in the domain of radiology. In their research, open-source LLMs were successfully used to extract necessary information from free-text radiology reports, allowing human decision-making without needing further fine-tuning of the model. This flexibility was essential because existing models were pre-trained for specific tasks, while the LLMs in their study needed to handle more varied reports. Similarly, simple word problems involving a few steps of addition and subtraction are elementary tasks, the underlying concepts of which are applied all the time in everyday life from cash counting to inventory management. This should ideally be well within the capabilities of LLMs, which are trained on vast datasets containing natural language and code. However if it is challenging, then we want to understand what would possibly need to be improved or tuned. Analysing the complexity of word problems with respect to calculation steps will give a deeper insight as to where and why LLMs struggle.

We hypothesised that LLMs should generalise well to such problems by extracting the necessary numbers and operations, leaving the actual calculation to an external process. This also helps to bypass any issues the LLM might encounter with complex numbers or situations that it hasn't seen during training, where predicting the answer directly could prove challenging. By separating the number extraction from the calculation, we remove some of the potential pitfalls associated with LLM predictions of complex or rare operations.

To augment the extracted numbers, we developed a simple counting function. This function initially began as a room counter, which was used in early tests focused on counting the number of people entering or leaving a room (see Appendix C1). In this set of tests, when asked for the answer directly, we observed that very large numbers or complicated problems proved difficult for the LLMs. Figure 04 below shows a sample of what was tested. Simple numbers are maximum 2 digits and complex problems have multiple subjects to account for.

Category	Example	Count of Solved	Ability To Solve
Simple Problem - Simple Numbers	32 people enter a room. 5 leave, 15 enter and 27 leave. How many people are in the room?	4/5	Easily solved
Simple Problem - Complex Numbers	322456 people enter a room. 52356 leave, 15151 enter, 6233 enter and 27634 leave. How many people are in the room?	0/5	Usually always wrong
Complex Problem - Simple Numbers	32 men and 14 women enter a room. 6 women leave, 17 men enter and 3 children enter. How many people are in the room?	1/5	Difficult

Figure 04 - Tested by manual prompts on GPT2, Falcon 7B and Llama 2 7B. Results may vary if prompted again or differently.

When creating the basic Counter class, initially the methods were named “Enter” and “Leave” as a response to our initial set of problems but eventually the names were generalised. The final class code can be viewed in Appendix C2 as part of final experimentation code. The class starts with a count initialised to zero and has three main methods:

- Add Method: This method takes a number and adds it to the current count.
- Subtract Method: This method subtracts a number from the current count, but if the result is less than zero, it returns the current count without making the subtraction. This prevents negative counts from occurring, though in hindsight, we recognize that this restriction could be adjusted.
- Current Count Method: This method returns the final count after all additions and subtractions have been performed.

In our experiments, after the LLM extracted the numbers and their respective operations from a word problem, we passed these values to our counting function. By using the Counter class to handle the calculation, we ensured that the final number was computed correctly, based on the numbers provided, as opposed to relying on the LLM to directly predict the answer. This setup allowed us to verify the accuracy of the extraction process and focus on refining our extraction attempts.

While we ensure accurate calculation, the LLM can still return inaccurate steps and it is the purpose of research to understand why.

3.4. LLM Output Parsing

3.4.1. Prompt Structure

As we are getting both the direct answers and the extracted steps, tests on prompt structure were required. While the prompts for direct answers were relatively straightforward, more effort was needed to develop the right prompt structure for extracting the numbers and operations. This experimentation was necessary to ensure the LLM would output the specific format we required. When talking about prompt engineering, there are various strategies for prompting LLMs, ranging from zero-shot to chain-of-thought prompting (Vatsal and Dubey, 2024).

To manage the scope of our experiments, we concentrated on four prompt types:

- A simple prompt asking for the answer directly.
- A simple prompt with three examples.
- An extracting prompt asking for the steps directly (i.e., extracting numbers and operations).
- An extracting prompt with three examples.

By keeping the number of prompts to these four categories, we were able to maintain control over the number of experiments conducted. Each of these prompt types was designed with the goal of eliciting structured and useful outputs from the LLM. Although variations of each prompt were tested, the goal was to arrive at a prompt structure that reliably produced parsable and structured output for further processing.

In the context of LLM prompting, our prompt designs align with established techniques. These prompts can be categorised as either zero-shot or few-shot prompts, depending on whether or not examples were included. All the prompts were instruction-based and explicitly asked the model to avoid irrelevant information, which is a common best practice in LLM prompting (Shi et al., 2023; Sanh et al., 2022). We also decided against using chain of thought prompting as since we are using small models, chain of thought prompts will likely not exhibit improvements in the way that they do when used with larger models with greater than 100 billion parameters (Wei et al, 2022). Additionally, adding another element to the experimentation plan increases the number of prompts dramatically, and while this would make the analysis richer, it would be at the cost of computation and time, which was limited. The techniques we employ have been well-documented in the literature and have been shown to improve performance on specific tasks by guiding the model toward the desired output structure.

The final prompts we used followed a simple and consistent structure:

- Set the scene: Provide context for the task at hand.
- Give the task: Clearly explain what the model is expected to do.
- Provide a set of instructions: Include clear guidelines on how the output should be formatted.
- Provide examples (for few-shot prompts): Show examples of similar problems with solutions, guiding the model on how to respond.
- Give a final instruction: Include an instruction on how the model should clearly highlight the final answer or extracted steps.

This structure can be seen in the below extraction prompt with examples. To see all the final prompt structures and variations of prompts tested, refer to Appendix D.

You are skilled at extracting numbers for addition and subtraction.

The user will ask you a question, for which you will extract the numbers that are being added or subtracted.

The numbers of things that are increasing should be categorized as 'add', and those that are decreasing should be categorized as 'subtract.'
Focus only on the steps to add or subtract.
Only consider relevant information.
Do not create additional examples.
Keep your response concise.
Focus on returning your result only for what is asked.

Return your result as a list of tuples in the format [(number ,'add' or 'subtract'), (number, 'add' or 'subtract')].
Expand the list for as many extracted numbers.

Here are some examples:

Question: "Saleem has 6 yellow and 15 green marbles. Jawad took 5 of Saleem's yellow marbles. How many yellow marbles does Saleem now have?"

Answer: [(6, 'add'), (5, 'subtract')]

Question: "There are 3 dogs and 17 cats currently in the park. People will bring more animals today. When the people come there will be 45 dogs in the park. How many dogs did people bring today?"
Answer: [(45, 'add'), (3, 'subtract')]

Question: "Josie had 120 coins. She won 80 more in a game, and then lost 45. How many coins were left?"
Answer: [(120, 'add'), (80, 'add'), (45, 'subtract')]

Highlight your returned result by specifying, "My final answer is:" and then show your result. Stop after this point.

As seen in the example above, in all our prompts, we included the phrase:
“Highlight your answer by specifying, 'My final answer is:' and then show your answer. Stop after this point.”
This instruction helped ensure that the model would output its answer in a predictable format (the number of errors discussed in section 3.4.2 were a small percentage of the overall) however it did not particularly improve the conciseness of answers. The aim was to avoid the generation of unnecessary explanatory text, as we were focused purely on extracting the final answer or the list of numbers and operations.

In the case of the extracting prompt, the structure needed to ensure compatibility with our Counter class for calculating the final result. This required the output to be in the form of a list of tuples, where each tuple contained a number and the associated operation (e.g. [(45, 'add'), (3, 'subtract')]). To achieve this, we included specific instructions for the model to format its output in the desired structure. The words “add” and “subtract” would be used in a simple regex match to direct a function to either use the add or subtract method to update the counter.

The phrasing of these extracting prompts was key to obtaining a reliable and parsable output. Through multiple iterations of prompt phrasing, we were able to design prompts that consistently produced the required list format.

Finally, at the end of the prompt structure, the user question was appended (e.g. The user has asked:...) to complete the prompt that was used as the input for the LLM.

3.4.2. Parsing Pipeline

Our parsing involved multiple steps, including writing custom functions and designing regex patterns to accurately match and extract the relevant information from the model responses.

For the simple prompts, our task was straightforward: extract the final answer, which was expected to be a number. The LLM was prompted to return the answer in a specific format: "My final answer is: [number]". To parse this, we simply used a regex pattern that matched the digits after the phrase "My final answer is:". Since the LLM was explicitly instructed to return only a number, this method worked well, as it avoided unnecessary parsing complexity and focused solely on extracting digits.

The extraction prompts required a more intricate approach. Initially, we attempted to extract all content within square brackets following the "My final answer is:" string. These square brackets contained a list of tuples, with each tuple comprising a number and an associated operation (e.g., [(12, "add"), (5, "subtract")]).

To check the validity of the extracted string, we initially used Python's ast.literal_eval function, which safely evaluates strings containing Python literals. While this approach helped ensure that the extracted string was syntactically valid, it missed various edge cases. For example, the LLM might return valid-looking output, but with slight formatting issues, incorrect ordering of elements or unexpected extra characters that the literal_eval function could not handle.

Recognizing the limitations of ast.literal_eval, we replaced it with a custom function called clean_tuples. The purpose of this function was to manually build the list of tuples by carefully extracting each number (whether an integer or float) and its associated operation (either "add" or "subtract").

The clean_tuples function included, matching all numbers (integers and floats), handling multiple quotes and removing extra white spaces and unnecessary parentheses that could distort the structure.

Additionally, to ensure that our parsing was reliable, we implemented a loop within each parsing function, which would attempt to parse the output up to three times. If all three attempts failed, a default value (0) was used to signify a failure in parsing. This fallback mechanism helped us avoid situations where an unparsed output would lead to issues in the next stages of the pipeline. We only stored the final parsed result, not the individual parsing attempts. The technique to have an evaluation check and then a default

process is similar to Cobbe et al (2021) during their testing of the GSM8K dataset when trying to inject calculator simulations (see Cobbe et al, 2021, appendix C).

Despite the robustness of our functions, several edge cases were missed by the automated parsing process, leading to errors in the final output. During our error analysis, we identified two common issues:

- For simple prompts: In some cases, the LLM would return the entire equation (e.g., "5 + 7 = 12") instead of just the answer. This required us to modify the parsing logic to handle equations, but due to time constraints, these cases were manually corrected in the results dataset.
- For extraction prompts: Occasionally, the LLM would reverse the order of the tuple, returning the operation first (e.g., ("add", 12) instead of (12, "add")). While this could have been handled with further debugging, we manually corrected these errors as well.

In total, there were 230 errors that required manual correction. Although this manual intervention was not ideal, it allowed us to continue with our analysis without significant delays. Can be viewed in Appendix E.

For more details on the code and functions used in this pipeline, please refer to Appendix C2.

3.5. Experimentation Pipeline

Before finalising our experimentation pipeline, we explored integrating additional tools and techniques, including LangChain and conducting decision-making experiments.

LangChain is a framework designed to facilitate the integration of LLMs with various external tools, such as databases and Python environments, making it easier to create more dynamic applications (Topsakal and Akinci, 2023). We initially experimented with LangChain to see if we could create a chain where the LLM output could dynamically interact with the Counter class. The goal was to use the output of one step as the input for the next step in the chain, allowing the model to update and return an answer based on the results of our custom counting function.

Additionally, we explored building a decision-making component within the pipeline. The purpose of this was to determine whether the LLM could make a choice about whether to use an external counting function. Specifically, we wanted to see if the model could recognize when it needed help to ensure the correct mathematical operations were applied based on the extracted numbers and operations. If given the

option, would the LLM decide to use the counting function to compute the answer? We successfully created the code (see Appendix C1) for this testing scenario but ultimately decided to leave this feature out of the final experiments. This was because it added complexity to the experiments and inflated the scope beyond what was manageable, given the focus of our study.

The final pipeline was simplified to focus on the core objectives of the study. The structure is as follows:

- Model Loading and Quantization:
    - We used Llama 2 7B and Llama 3 8B models. Both models were quantized to reduce memory usage and computational requirements.
- Dataset Preparation:
    - Our cleaned dataset, consisting of 395 word problems, was loaded into the pipeline.
- Prompt Generation:
    - Using the prompt structures we designed, we created final prompts for the LLMs. For each of the four prompt categories, we generated one prompt per word problem. This resulted in 1,580 prompts in total.
- LLM Interaction:
    - Each of the 1,580 prompts was passed to both Llama 2 7B and Llama 3 8B, resulting in 3,160 total LLM interactions.
    - In order to account for variability and ensure robust results, each prompt was run five times (attempts) by each LLM, as suggested by similar studies (Srivatsa and Kochmar, 2024). This resulted in a total of 15,800 experiments (3,160 prompts x 5 runs).
    - After each attempt the colab memory was reset so previous prompts did not influence next prompts
- Output Parsing and Storage:
    - Each output from the LLMs was then parsed using our custom parsing functions. The parsed data, including the extracted numbers, operations, and final answers, were stored for further evaluation.
    - To gather results, we split our code into 4 notebooks. It was the same code except with different filters on which LLM to use and which set of prompts to use. We did this to run the experiments in parallel to save on time.
    - Data was stored as a csv from each notebook which was later uploaded to Google Sheets for exploration and manual correction

While we ran 15,800 experiments, we had to re-run the set of experiments pertaining to digits that were either 17 or 18 digits long as our initial LLM parameters were not set high enough to generate enough tokens to view the full answer. See Appendix E for all results.

4. Results

Upon completion of all the experiments and the reruns, we conducted a detailed analysis of our results. We, once again, identified an issue with questions involving numbers that were 17 or 18 digits long. When opening the results as CSV files, we noticed precision errors in these specific cases, which affected the accuracy of both the final answers and the extracted steps. This issue only occurred with 17 and 18 digit decimals, while other decimals were unaffected. Upon further investigation, we discovered that this seemed to be caused by the limitations of floating point arithmetic as defined by the IEEE 754 specification, which is used by spreadsheet programs to store floating point numbers (Microsoft, 2024; Hyde, 2021). When large, repeating decimals, such as the ones in our dataset (all 17 and 18 digit questions were large, repeating decimals), cross the floating point threshold, they introduce small errors in displaying and calculating the results. Take for example the following word problem:

A construction company ordered 0.16666666666666666 ton of concrete, 0.16666666666666666 ton of bricks and 0.5 ton of stone. How many tons of material did the company order in all?

The answer by the LLM was “0.833333333333333”, and the human solved answer was “0.833333333333332”. The LLM extracted the steps: [(0.16666666666666666, 'add'), (0.16666666666666666, 'add'), (0.5, 'add')]. You will notice that the question displays 17 decimal places but the answers display only 15. The displayed answers have been extracted by the LLM output and viewed in a spreadsheet. This difference, though seemingly small, will be marked wrong by our evaluation method as we are matching exact strings. Despite our efforts to ensure accuracy, these issues, which occurred during the process of handling the data in Google Sheets, were unforeseen. We could have rounded the questions and answers after obtaining the results but given that the LLMs were given long strings of decimals and were not prompted to round to a certain decimal place, we expected unrounded, complete strings. Doing so post the experiments would detract from a robust design. We therefore decided to remove these questions from the final analysis. In total, we removed 32 questions from the original 395, accounting for 1,280 prompts out of the 15,800 total prompts, or about 8% of the dataset. Our following analysis will be on 14,520 prompts.

4.1. Llama Performance Overall

When evaluating performance, we consider three key metrics.

First, Accuracy, which is the number of correct answers divided by the total number of questions. For this, we assess whether the strings in the answers are accurately matched. This applies to both simple and extracting prompts—if the steps extracted in the extracting prompts are correct, the calculated final answer from those steps will also be accurate.

Second, we examine the Mean Relative Error (MRE), which measures the average difference between the actual and predicted values, expressed as a percentage of the actual values. This metric helps us understand the relative size of errors rather than just their absolute differences.

Lastly, we assess statistical significance at the 5% and 1% levels to compare accuracy, ensuring that any differences observed between models are meaningful and not due to random chance.

Figure 05 below shows the per attempt accuracy of both Llama models across all prompts.

		Llama 2-7B			Llama 3-8B		
Attempt	Total Questions	Correct Answers	Accuracy of Answers	Mean Relative	Correct Answers	Accuracy of Answers	Mean Relative Error
1	1,452	863	59%	25%	1,060	73%	19%
2	1,452	868	60%	24%	1,080	74%	18%
3	1,452	873	60%	25%	1,059	73%	20%
4	1,452	879	61%	25%	1,080	74%	18%
5	1,452	869	60%	24%	1,063	73%	20%
All Attempts	7,260	4,352	60%	25%	5,342	74%	19%

Figure 05 - Accuracy and MRE of both Llamas across prompts and 5 attempts

From the table, we can see that both models consistently reach similar accuracies within themselves across attempts. Llama 2 has an accuracy of approximately 60% while Llama 3 has an accuracy of approximately 74% which is a much higher accuracy. We further calculated the associated p-value through a two proportion z-test and obtained a value of 4.02408e-68 which indicates statistical significance at both the 5% and 1% levels. This tells us that the observed difference between the two models is extremely unlikely to be due to chance and is reflective of an actual improvement in performance.

In addition to accuracy improvements, the MRE for both models is stable and Llama 3 also shows much lower MRE across all attempts by approximately 6%. On average, the errors made by the Llama 3 model

when compared to the actual answers are smaller than those by Llama 2 which reinforces Llama 3's improvements in accuracy and less severe errors when incorrect.

This observation is seen within the variation of unique answers as well. Across all attempts and prompts, the variation in unique answers was lower for Llama 3 as compared to Llama 2. For the questions in our dataset, post removal of questions that were related to 17 or 18 digit inputs, there were 205 unique answers. Llama 2 generated 467 unique answers or 2.28 different answers per unique answer while Llama 3 generated 336 unique answers or 1.64 different answers per unique answer. This shows that the new generation is generating relatively less variance as noticed by the increase in accuracy.

Given the consistency per attempt of the LLMs, we can now take a look at the accuracy and MRE by prompt category and features across all the attempts without additional worry about variation in attempts.

Beginning with the scores across features, Figure 06 below shows a summary of the results.

			Llama 2-7B			Llama 3-8B		
Feature	Value	Total Questions / Prompts	Correct Answers	Accuracy of Answers	Mean Relative Error	Correct Answers	Accuracy of Answers	Mean Relative Error
Question Length	A. x<100	660	585	89%	7%	615	93%	6%
	B. 100<=x<200	5,960	3,495	59%	25%	4,332	73%	20%
	C. x>=200	640	272	43%	41%	395	62%	24%
Information Irrelevance	0	1,740	1,461	84%	9%	1,530	88%	10%
	1	2,720	1,641	60%	25%	2,003	74%	17%
	2	2,800	1,250	45%	34%	1,809	65%	27%
Question Steps	1	6,180	3,961	64%	24%	4,639	75%	20%
	2	1,080	391	36%	25%	703	65%	17%
Question Digits	1	1,760	1,167	66%	19%	1,339	76%	19%
	2	3,140	1,874	60%	30%	2,321	74%	21%
	3	880	604	69%	16%	704	80%	13%
	4	1,120	606	54%	28%	777	69%	20%
	5	340	91	27%	14%	181	53%	16%
	6	20	10	50%	0%	20	100%	0%

Figure 06 - Accuracy and MRE by features across 5 attempts

From Figure 06 we can see that, in relation to the question length, the results of Srivatsa and Kochmar (2024) and Xu et al. (2024) are present in our experiments as well. As the question length increases the accuracy looks to decrease and the MRE increases. The drop off is quite large as we move to questions that are 100 characters with Llama 2 losing 30% accuracy and Llama 3 losing 20% accuracy.

With information irrelevance, once again we can see confirmation of results from existing literature by Shi et al. (2023) and Anantheswaran et al. (2024). Both models are losing accuracy and increasing the magnitude of errors as the irrelevant information in the question increases. Questions marked as 2 in the irrelevant feature category typically introduce one extra number that is irrelevant. Even with that simple introduction we see accuracy performance drop by 39% for Llama 2 and 23% for Llama 3.

The question step feature is interesting as, for simplicity, our data only had 1 or 2 steps. The introduction of just a second step has made Llama 2 perform considerably worse by losing 28% in accuracy while Llama 3 lost 10%.

For question digits, scores for the first 3 digits are relatively stable for both models but the 4th digit onwards we begin to see worse scores with some variance.

What is surprising about these results is that we have used a simple dataset and taken away much of the potential complexity. The questions do not have particularly large numbers, long steps or large amounts of irrelevant information as compared to the datasets used by other researchers. Even in this simple setting, the limitations of our chosen, recently developed smaller LLMs are put on stark display. We will dive deeper in the next sections.

Taking a look at the performance by prompt category, we can see results in Figure 07 below.

			Llama 2-7B		Llama 3-8B		
Prompt Category	**Total Questions**	**Correct Answers**	**Accuracy of Answers**	**Mean Relative Error**	**Correct Answers**	**Accuracy of Answers**	**Mean Relative Error**
Extract-Relevant	1,815	841	46%	44%	936	52%	39%
Extract-Relevant-FewShot	1,815	1,073	59%	29%	1,139	63%	31%
All Extract	3,630	1,914	53%	37%	2,075	57%	35%
Simple-Relevant	1,815	1,227	68%	13%	1,621	89%	5%
Simple-Relevant-FewShot	1,815	1,211	67%	12%	1,646	91%	3%
All Simple	3,630	2,438	67%	12%	3,267	90%	4%
All Prompts	7,260	4,352	60%	25%	5,342	74%	19%

Figure 07 - Accuracy and MRE by prompt category across 5 attempts

From the table, taking a look at the simple prompts first, which involved asking for the answer directly, Llama 2 achieves an accuracy of approximately 67% while Llama 3 achieves an impressive 90%. The

difference between direct prompting and few shot prompting is negligible in this case. Asking for the answer directly also results in much lower MREs with Llama 2 have an approximately 12% error and Llama 3 having approximately 4%. Both results show a large difference between the models with Llama 3 out performing Llama 2.

Results from the extract prompts show dips in accuracies for both models. Directly asking for the steps to be added or subtracted gives the worst accuracies where Llama 2 achieves 46% and Llama 3 achieves 52%. While there is a difference of 6% between these models, it is not as large when compared to the simple prompts. The few shot prompts improves the extracting results for both models with Llama 2 achieving 59% and Llama 3 achieving 63%, but once again the difference of 4% is not as pronounced as the simple prompts. In fact, providing examples within the prompts almost brings Llama 2 performance on par with Llama 3. This can be observed within the MRE scores as well where although the examples decreased the relative error, Llama 2 benefited more from these examples than Llama 3 (approximately 15% reduction for Llama 2 as compared to 8% for Llama 3) to the point where its relative error was lower than Llama 3. This suggests that in terms of thinking about the steps to solve a problem, there are only a few percentage points that separate the models. In terms of LLM advancement, this is still a good improvement, but when compared against the simple prompts, the relative improvement is not as high.

Figure 08 below looks at some comparisons for the significance of the differences between the accuracies.

Accuracy Significance Table			
Comparison	**P Value**	**5% Level**	**1% Level**
llama 2 extract relevant vs llama 3 extract relevant	1.60914E-03	Yes	Yes
llama 2 extract relevant fewshot vs llama 3 extract relevant fewshot	2.47515E-02	Yes	**No**
llama 2 simple relevant vs llama 3 simple relevant	5.70674E-57	Yes	Yes
llama 2 simple relevant fewshot vs llama 3 simple relevant fewshot	1.30593E-69	Yes	Yes
llama 2 extract relevant vs llama 2 extract relevant fewshot	1.23093E-14	Yes	Yes
llama 3 extract relevant vs llama 3 extract relevant fewshot	9.83677E-12	Yes	Yes
llama 2 simple relevant vs llama 2 simple relevant fewshot	5.71746E-01	**No**	**No**
llama 3 simple relevant vs llama 3 simple relevant fewshot	1.66623E-01	**No**	**No**

Figure 08 - Comparison of significant differences between accuracies by prompts across 5 attempts

For within model differences for simple prompts, the differences in accuracy by including examples are not significant at either the 5% or 1% levels. The accuracy differences themselves were neglible and any

difference can be due to chance as the variation of answers across attempts, as we saw before, was relatively stable. The accuracy differences between models for the simple prompts were significant at both levels and confirms the large improvements.

For within model differences for extract prompts, the presence of examples was meaningful for both models at both levels. However, when comparing across models, we notice that at the 1% level for few shot extract prompts compared between models, the difference is not significant (although it is at 5%).

In terms of generating an output from the input prompt and storing the result, Llama 3 was slower across both simple and extract prompts as compared to Llama 2. For simple prompts, Llama 2 took 1.24s/prompt while Llama 3 took 1.86s/prompt. For extract prompts, Llama 2 took 2.38s/prompt while Llama 3 took 4.30s/prompt. Please see Appendix F for complete times.

Llama 3 generally performs better, if a bit slower, but in relation to extracting prompts, where the LLMs were prompted to output the steps to solve the word problem, the difference between Llama 2 and Llama 3 is small, and the models seem more similar than expected.

4.2. Effect of Word Problem Structure on Performance

4.2.1. Number of Digits

		Llama 2-7B				Llama 3-8B	
Number of Digits	Total Questions	Correct Answers	Accuracy of Answers	Mean Relative Error	Correct Answers	Accuracy of Answers	Mean Relative Error
1	1,760	1,167	66%	19%	1,339	76%	19%
2	3,140	1,874	60%	30%	2,321	74%	21%
3	880	604	69%	16%	704	80%	13%
4	1,120	606	54%	28%	777	69%	20%
5	340	91	27%	14%	181	53%	16%
6	20	10	50%	0%	20	100%	0%

Figure 09 - Accuracy and MRE by digits across 5 attempts

Figure 09 above shows how the models perform based on the number of digits that are involved in the question. You will notice that for 6 digits, Llama 2 has a 50% accuracy but 0% MRE. This is not actually 0% but an extremely small figure, 0.0013% to be exact. Overall the results do not seem to follow a

specific trend, for example constant decline in accuracy as number of digits increases however we do see a notable drop in accuracy for both models when the number of digits is 5. From a peak of approximately 69%, Llama 2 drops to 27% while Llama 3 drops from 80% to 53%. Llama 3 performs better in all cases along with lower relative errors although for single digit questions, both models have a near equal MRE suggesting that both models make errors of similar magnitudes for these types of questions. Additionally, for 5 digit questions, while Llama 2 has a much worse accuracy, the MRE is about 2 percentage points better suggesting that for these digits when the model is incorrect, it is generally incorrect by a smaller margin than Llama 3.

For 6 digit questions, given the small sample size, we can not consider the scores fully representative despite the flawless performance of Llama 3. The previous trend suggests that the score should decrease and if the sample size were larger we might see the trend continue.

			Llama 2-7B			Llama 3-8B		
Prompt Category	Number of Digits	Total Questions	Correct Answers	Accuracy of Answers	Mean Relative Error	Correct Answers	Accuracy of Answers	Mean Relative Error
Extract-Relevant	1	440	192	44%	39%	259	59%	34%
	2	785	368	47%	57%	405	52%	41%
	3	220	126	57%	28%	125	57%	31%
	4	280	114	41%	37%	109	39%	48%
	5	85	36	42%	18%	33	39%	38%
Extract-Relevant-FewShot	1	440	254	58%	25%	263	60%	34%
	2	785	466	59%	34%	489	62%	34%
	3	220	145	66%	28%	159	72%	20%
	4	280	168	60%	29%	178	64%	29%
	5	85	35	41%	10%	45	53%	25%
Simple-Relevant	1	440	357	81%	6%	405	92%	5%
	2	785	518	66%	17%	709	90%	6%
	3	220	171	78%	4%	210	95%	1%
	4	280	171	61%	21%	241	86%	3%
	5	85	10	12%	12%	51	60%	0%
Simple-Relevant-FewShot	1	440	364	83%	7%	412	94%	4%
	2	785	522	66%	12%	718	91%	3%
	3	220	162	74%	5%	210	95%	1%
	4	280	153	55%	26%	249	89%	1%
	5	85	10	12%	13%	52	61%	1%

Figure 10 - Accuracy and MRE by prompts and digits across 5 attempts

Figure 10 above shows a deeper breakdown of the number of digits by prompt category. In the table, 6 digit questions are not shown since the sample size is too small to consider. For simple problems, Llama 3 consistently outperforms Llama 2 in both accuracy and MRE. We additionally see declining accuracy by the 5th digit.

The case of extracting prompt is unique as without examples, for the first two digits, Llama 3 was outperforming Llama 2 by about 15 and 5 percentage points. However, from the third digit onwards, Llama 2 was performing on par or better, admittedly by a small margin. The roles are reversed once examples are introduced as Llama 3 begins to show much improved accuracy while Llama 2 struggles to take advantage of the additional information to maintain the better performance before examples. Additionally, the Llama 2 MRE for the 5 digit questions, without examples, was dramatically lower than Llama 3 (18% to 38%). This trend continued when examples were introduced indicating that Llama 2 is not that wrong when it outputs an incorrect answer.

While Llama 3 performs much better on simple prompts, when it comes to understanding and extracting steps, both models have a similar performance with Llama 3 performing marginally better meaning that the number of digits are handled similarly for both models.

4.2.2. Question Length

		Llama 2-7B			Llama 3-8B		
Question Length	Total Questions	Correct Answers	Accuracy of Answers	Mean Relative Error	Correct Answers	Accuracy of Answers	Mean Relative Error
A. x<100	660	585	89%	7%	615	93%	6%
B. 100<=x<200	5,960	3,495	59%	25%	4,332	73%	20%
C. x>=200	640	272	43%	41%	395	62%	24%

Figure 11 - Accuracy and MRE by question length across 5 attempts

In the figure above, we see the accuracy and MRE of both models with respect to the question length of the simple maths word problem. We can see that for both models as the length of the question increases, the accuracy decreases and the MRE increases. For questions that are less than 100 characters, both models perform well with Llama 2 reaching 89% in accuracy while Llama 3 reaches an impressive 93% in accuracy. In the case of Llama 2, when the question length reaches 200 or more characters, the MRE dramatically increases as well, indicating that this model may not handle longer prompts (or input tokens) well.

			Llama 2-7B			Llama 3-8B		
Prompt Category	Question Length	Total Questions	Correct Answers	Accuracy of Answers	Mean Relative Error	Correct Answers	Accuracy of Answers	Mean Relative Error
Extract-Relevant	A. x<100	165	121	73%	21%	140	85%	11%
	B. 100<=x<200	1,490	674	45%	44%	742	50%	40%
	C. x>=200	160	46	29%	66%	54	34%	53%
Extract-Relevant-FewShot	A. x<100	165	152	92%	5%	145	88%	11%
	B. 100<=x<200	1,490	845	57%	30%	910	61%	32%
	C. x>=200	160	76	48%	41%	84	53%	39%
Simple-Relevant	A. x<100	165	159	96%	2%	165	100%	0%
	B. 100<=x<200	1,490	994	67%	12%	1,329	89%	5%
	C. x>=200	160	74	46%	29%	127	79%	3%
Simple-Relevant-FewShot	A. x<100	165	153	93%	1%	165	100%	0%
	B. 100<=x<200	1,490	982	66%	12%	1,351	91%	3%
	C. x>=200	160	76	48%	27%	130	81%	2%

Figure 12 - Accuracy and MRE by prompts and question length across 5 attempts

In the figure above, we see the further breakdown of Question Length by the prompt category. Across all prompt types, as the length of the question increases, the accuracy of both models decrease. Both models perform extremely well with simple prompts (asking for the question directly) and short questions with Llama 2 achieving 96% accuracy and Llama 3 getting a perfect score. With simple, few shot prompts, Llama 3 is able to increase the accuracy for all question lengths. Llama 2 on the other hand is not able to take advantage of the examples in the same manner.

Once again, the case of the extracting prompts is a curious one. While Llama 3 continues to outperform by a small margin, when examples are introduced, Llama 2 is able to take incredible advantage of this in questions less than 100 characters to achieve an accuracy 92%. This is unlike the digits case where Llama 2 was not able to take advantage of the added examples. The MRE scores between the simple and extract prompts also suggest that both models found the extract prompts more difficult to answer and made errors in outputting the correct steps by a large margin.

Llama 3 outperforms on simple tasks but both models handle extract cases similarly, with Llama 2 showing potential to improve based on examples.

4.2.3. Information Irrelevance

		Llama 2-7B			Llama 3-8B		
Information Irrelevance	Total Questions	Correct Answers	Accuracy of Answers	Mean Relative Error	Correct Answers	Accuracy of Answers	Mean Relative Error
0	1,740	1,461	84%	9%	1,530	88%	10%
1	2,720	1,641	60%	25%	2,003	74%	17%
2	2,800	1,250	45%	34%	1,809	65%	27%

Figure 13 - Accuracy and MRE by information irrelevance across 5 attempts

The figure above describes how Llama 2 and Llama 3 handle irrelevant information in a question. As stated before, a question with information irrelevance of 0 means that every number and entity is involved for the answer and there is no extra information. A category of 2 implies that there are additional values that are not needed to obtain the answer.

For questions where all information is relevant (0), both models handle the word problems well with Llama 3 performing about 4% better than Llama 2, with both models scoring in the 80s.

With some irrelevant information (1) we begin to see performance drops with notably larger drops in Llama 2.

Questions with the most irrelevant or confounding information makes Llama 2 perform very poorly, losing nearly 40 percentage points of accuracy as compared to Llama 3 that is still able to achieve an accuracy of 65% which is about 20 percentage points higher than Llama 2.

For both models, the MRE tends to increase as well as irrelevant information increases.

			Llama 2-7B			Llama 3-8B		
Prompt Category	Information Irrelevance	Total Questions	Correct Answers	Accuracy of Answers	Mean Relative Error	Correct Answers	Accuracy of Answers	Mean Relative Error
Extract-Relevant	0	435	296	68%	21%	357	82%	13%
	1	680	346	51%	39%	330	49%	39%
	2	700	199	28%	63%	249	36%	55%
Extract-Relevant-FewShot	0	435	353	81%	10%	310	71%	26%
	1	680	406	60%	31%	435	64%	29%
	2	700	314	45%	39%	394	56%	36%
Simple-Relevant	0	435	409	94%	1%	430	99%	0%
	1	680	452	66%	15%	619	91%	1%
	2	700	366	52%	18%	572	82%	11%
Simple-Relevant-FewShot	0	435	403	93%	3%	433	100%	0%
	1	680	437	64%	15%	619	91%	0%
	2	700	371	53%	15%	594	85%	7%

Figure 14 - Accuracy and MRE by prompts and information irrelevancy across 5 attempts

A similar pattern is repeated in the figure above where for the simple prompts Llama 3 performs exceptionally well and outperforms Llama 2 by a large margin. No accuracy score of the Llama 3 model is less than 80% demonstrating that at being asked for the question directly, irrelevant information is less of a problem, although both models see decreases in accuracy. Llama 2 is able to answer questions correctly when all the information is relevant, achieving an accuracy of about 94% (with and without examples) but as the irrelevant information increases, performance decreases significantly.

In the case of extract prompts, with all relevant information, the LLMs need to identify which numbers are to be added and which are to be subtracted. Without examples in the prompt, Llama 2 has an accuracy of 68% while Llama 3 performs well with 82% accuracy. However, when examples are introduced into the prompt, for questions with all relevant information, Llama 2 is able to take advantage and improve accuracy to 81% whereas Llama 3 unexpectedly performs worse with a 71% accuracy. For the rest of the categories, Llama 3, as usual, begins to perform better although overall, as irrelevance increases, the accuracy of both models decreases.

Llama 3 outperforms on the simple prompts and while we notice a pattern again where Llama 2 and Llama 3 can have comparable performance in extracting steps with examples introduced, Llama 3 tends to handle the irrelevant information better.

4.2.4. Question Steps

		Llama 2-7B			Llama 3-8B		
Question Steps	Total Questions	Correct Answers	Accuracy of Answers	Mean Relative Error	Correct Answers	Accuracy of Answers	Mean Relative Error
1	6,180	3,961	64%	24%	4,639	75%	20%
2	1,080	391	36%	25%	703	65%	17%

Figure 15 - Accuracy and MRE by question steps across 5 attempts

In the figure above we see the accuracy of both models by the number of steps required to solve the question. Even in these simple maths word problems, we see a sizable decrease in accuracy for the Llama 2 model by nearly 28 percentage points, from 64% to 36%. Llama 3 sees a decrease as well of 10 percentage points, from 75% to 65%, which is not as pronounced as Llama 2. In cases for both question steps, Llama 3 performs better than Llama 2.

			Llama 2-7B			Llama 3-8B		
Prompt Category	Question Steps	Total Questions	Correct Answers	Accuracy of Answers	Mean Relative Error	Correct Answers	Accuracy of Answers	Mean Relative Error
Extract-Relevant	1	1,545	775	50%	46%	818	53%	40%
	2	270	66	24%	35%	118	44%	33%
Extract-Relevant-FewShot	1	1,545	974	63%	29%	986	64%	32%
	2	270	99	37%	29%	153	57%	24%
Simple-Relevant	1	1,545	1,119	72%	12%	1,407	91%	5%
	2	270	108	40%	18%	214	79%	4%
Simple-Relevant-FewShot	1	1,545	1,093	71%	11%	1,428	92%	2%
	2	270	118	44%	17%	218	81%	5%

Figure 16 - Accuracy and MRE by prompts and question steps across 5 attempts

In the figure above we can see Llama 3 significantly outperforming Llama 2 in the simple prompts. Neither model is able to take advantage of the examples available in the few shot prompts. The MRE for Llama 3 in the simple cases are significantly lower as well suggesting very few errors.

In the case of extract prompts however, we can once again see that the difference between Llama 2 and Llama 3 while present, is not as large. Llama 2 is clearly worse when the number of steps increases but when there is only one step, there is less than a 5 percentage point difference that separates the 2 models. In fact, Llama 2 is able to take advantage of the examples available in the few shot prompts to such an

extent that it increased its score by 13 percentage points and reduced the difference between the models to only 1 percentage point, 63% to 64%. It is even able to reduce the relative size of errors when it makes mistakes to below 30%.

Llama 3 once again, performs better than Llama 2 for simple prompts. In the case of extract prompts, when there is only one step, the performance is comparable but as step size increases, Llama 3 begins to perform better.

4.3. Error Analysis

Here we investigate specific cases where both Llama 2-7B and Llama 3-8B struggled to produce correct answers. Each question was tested 20 times per LLM, across 4 different prompt types (simple and extract, with and without examples) for 5 attempts each. We focus on why certain questions might have confused the models, as well as variations in performance between the two LLMs.

4.3.1. Spaceship Travel Problem

Question: "A spaceship travelled 0.5 light-year from Earth to Planet X and 0.1 light-year from Planet X to Planet Y. Then it travelled 0.1 light-year from Planet Y back to Earth. How many light-years did the spaceship travel in all?"

Model Performance: Both LLMs consistently answered this question incorrectly at every attempt. For simple prompts, the answer provided by both models was 0.6, and for extracted steps, the answer included a misinterpretation of the journey back to Earth, outputting [(0.5, 'add'), (0.1, 'add'), (0.1, 'subtract')]. The correct answer should be 0.7.

Analysis: This question likely confused both models due to the back-and-forth travel between planets, causing an error in the addition of distances. The models seem to have mistakenly treated the journey back as a subtraction instead of continuing to add distances. This could be due to the phrasing of the question where both models appear to have interpreted the reverse journey incorrectly.

4.3.2. Mary's Money Problem

Question: "After paying 6 dollars for the pie, Mary has 52 dollars, her friend has 43 dollars. How much money did she have before buying the pie?"

Model Performance: Both models were wrong at every attempt. Llama 3 consistently returned 98 as the direct answer in all simple prompts. However, extracted steps varied significantly across prompts and models, with Llama 2 and Llama 3 both producing illogical step sequences (e.g., Llama 2's steps: [(6.0, 'add'), (52.0, 'add'), (43.0, 'subtract')], Llama 3's steps: [(52.0, 'add'), (6.0, 'subtract'), (43.0, 'subtract')]).

Analysis: The confusion likely arose from the irrelevant information about Mary's friend's money, which distracted the models from focusing on the key steps of the problem despite being prompted to ignore irrelevant information. Llama 3's consistency in returning 98 suggests it may have misinterpreted the structure of the problem while Llama 2 showed more variability but still struggled with filtering out information.

4.3.3. Oranges and Apples Problem

Question: "Mary picked 14 oranges and Jason picked 41 oranges. Keith picked 38 apples. How many oranges were picked in all?"

Model Performance: Llama 2 answered 17 in four attempts and in the remaining attempts answered 93. Extracted steps frequently included an error such as [(14.0, 'add'), (41.0, 'add'), (38.0, 'add')], which included the apples. Llama 3 only got this wrong once, correctly identifying oranges in most cases.

Analysis: Llama 2 struggled to separate the two categories of fruit. This indicates a failure to properly handle irrelevant information. Llama 3, on the other hand, handled the irrelevant data better, demonstrating stronger information filtering.

4.3.4. Pokemon Cards Problem

Question: "Joan had 695 Pokemon cards and 6 were torn. Sara bought 133 of Joan's Pokemon cards. How many Pokemon cards does Joan have now?"

Model Performance: Llama 2 initially struggled in the extraction prompts without examples, producing steps like [(695.0, 'add'), (6.0, 'subtract'), (133.0, 'add')]. All 5 attempts here were wrong. However, with the few-shot examples, Llama 2 began to get the correct answer with steps like [(695.0, 'add'), (133.0, 'subtract')]. All attempts here and in the simple prompts were correct. Llama 3 consistently got the question wrong across all prompts.

Analysis: This question involves reverse reasoning and managing different operations (adding and subtracting). Llama 2 benefited significantly from the few-shot examples, suggesting that the additional context helped the model learn the relationship more effectively. Llama 3, however, struggled to identify the correct relationship between the numbers, particularly failing to recognize the importance of subtracting Sara's cards.

4.3.5. Baseball Cards Problem

Question: "Dan had 97 baseball cards and 8 were torn. Sam bought 15 of Dan's baseball cards. How many baseball cards does Dan have now?"

Model Performance: Surprisingly, Llama 2 outperformed Llama 3 significantly, getting 9 out of 10 extraction prompts correct and 1 out of 10 simple prompts correct. Llama 3, on the other hand, did not answer this question correctly in any prompt.

Analysis: The word "torn" seems to have caused confusion, especially for Llama 3, which consistently failed to pick up on the required subtraction from torn cards. Llama 2, though initially struggling with simple prompts, improved with examples. This is similar to the Pokemon Cards problem.

4.3.6. Patterns with the Word "Torn"

Across the 6 unique questions referencing the word "torn" (120 prompts), Llama 3 performed very poorly, getting only 5 out of 120 correct (all in simple prompts without examples), whereas Llama 2 managed to get 53 out of 120 correct.

The word "torn" appears to consistently confuse Llama 3, particularly in extraction prompts. The model may have learned that the word “torn” should be subtracted as opposed to ignored, if the context requires it. Llama 2 demonstrated a better ability to manage this type of question, possibly benefiting more from the examples and showing greater flexibility in understanding the context of the word.

4.3.7. This Year Last Year Questions

During error analysis, another curious pattern emerged. Consider the following questions (discussed further in section 5.1.2)

Q1: Sara's high school played 12 basketball games this year. The team won most of their games. They were defeated during 4 games. How many games did they win?

Q2: Joan went to 4 football games this year. She went to 9 games last year. How many football games did Joan go to in all?

Q1 is a double digit question, with a single step and an information irrelevance of 1.

Q2 is a single digit question, with a single step and an information irrelevance of 1.

For Q1, both Llama 2 and Llama for simple prompts (with the exception of no answer in output for one simple prompt on Llama 2) got the question correct every time. In the case of extract prompts, Llama 3 got the question correct every time and Llama 2, got this question wrong on extract only prompts but with extract few shot prompts got this question correct.

For Q2, both Llama 2 and Llama 3 were able to answer the question correctly every time using simple prompts. However, in the case of extract prompts, both LLMs got the steps wrong every time. Both outputted, on different attempts, the steps of [(4.0, 'add'), (9.0, 'subtract')] and [(9.0, 'add'), (4.0, 'subtract')].

4.3.8. Questions That Were Always Wrong

The following table shows a list of questions that both LLMs got wrong at every attempt at every prompt.

Fixed_Question	Llama 2 Correct	Llama 3 Correct
A spaceship traveled 0.5 light-year from Earth to Planet X and 0.1 light-year from Planet X to Planet Y. Then it traveled 0.1 light-year from Planet Y back to Earth. How many light-years did the spaceship travel in all?	0	0
After paying 6 dollars for the pie, Mary has 52 dollars, her friend has 43 dollars. How much money did she have before buying the pie?	0	0
Alyssa went to 11 soccer games this year, but missed 12. She went to 13 games last year and plans to go to 15 games next year. How many soccer games will Alyssa go to in all?	0	0
Dan's cat had kittens and 5 had spots. He gave 7 to Tim and 4 to Jason. He now has 5 kittens. How many kittens did he have to start with?	0	0
Fred went to 36 basketball games this year, but missed 35. He went to 11 games last year. How many basketball games did Fred go to in total?	0	0
Melanie had 10 quarters and 17 pennies in her bank. Her dad gave her 27 pennies and her mother gave her 19 pennies. How many pennies does Melanie have now?	0	0
Mike went to 15 basketball games this year, but missed 41. He went to 39 games last year. How many basketball games did Mike go to in total?	0	0
There are 48 erasers in the drawer and 30 erasers on the desk. Alyssa placed 39 erasers and 45 rulers on the desk. How many erasers are now there in total?	0	0
Tom bought a skateboard for $ 9.46 and spent $ 9.56 on marbles. Tom also spent $ 14.50 on shorts. In total, how much did Tom spend on toys?	0	0
Tom went to 4 hockey games this year, but missed 7. He went to 9 games last year. How many hockey games did Tom go to in all?	0	0

Figure 17 - Questions where both Llama models were wrong at every attempt

The first and 9th question in the table are marked as level 1 of information irrelevance in the data while the rest are marked at level 2. The models seem to struggle with these likely because of the non standard or less than typical language used in the question. For example when talking about Dan's kittens, it requires reverse engineering to get to the original answer.

In the 9th question, the question asks about toys and does not indicate which of the objects in the sentence, skateboard, marbles or shorts are toys, leaving this up to interpretation for the models.

The words “missed” seem similar to the word “torn” in the previous discussion as a cause of confusion but in this case llama 2 was not able to pick up on the similarity and get even a few attempts correct.

5. Discussion

5.1. Research Objectives

5.1.1. Assess the Word Problem Structure to Understand What Affects Accuracy

To assess how the structure of word problems affects model accuracy, we started by reviewing relevant literature. Studies by Srivatsa and Kochmar (2024) and Xu et al. (2024) highlight how question length can create challenges for language models, while Shi et al. (2023) and Anantheswaran et al. (2024) point to the issue of information relevance where irrelevant details in a question can distract models and lead to errors. Additionally, inspired by the work of Wei et al. (2022) on chain-of-thought prompting, we wanted to examine how the number of steps required to solve a problem impacts performance. We also considered the number of digits in the questions as a potential factor. The more digits present in a question, the more tokens the model needs to generate accurately, which increases the complexity of the task. Large numbers are less likely to be found in publicly available training data, making it more difficult for models to handle them.

While previous research primarily tested these features on large models with high parameter counts, we chose to explore these features in small models as higher parameters could mask simpler sources of failure. Larger models like those tested on datasets such as GSM8K (Cobbe et al., 2021) often involve complex algebraic operations, which could obscure the specific challenges posed by the structure of simpler word problems. To isolate the effects of problem structure, we restricted ourselves to basic addition and subtraction.

While we cannot definitively claim that the models were better trained on addition and subtraction than on multiplication or division, our decision to focus on these operations minimises the risk of complications arising from large numbers or repeating numbers that can be the outputs of multiplication or division operations. Such numbers are less likely to be encountered in training data and could trigger repetition penalties in the models, negatively affecting their performance. By simplifying the tasks, we keep the focus on the structure of word problems itself rather than the complexity of the arithmetic.

To test these structural features, we selected a dataset by Hosseini et al. (2014), which consisted of simple word problems that matched our criteria. With this dataset as our base, using python we created the previously examined features like question length, information relevance, number of digits, and question steps.

Within this simple dataset and controlled environment we are able to observe our chosen LLMs struggle with minor increases in complexity.

Comparing performance through the lens of the number of digits, with our simple approach we are able to highlight the inflection point of error. Figure 09 in section 4.2.1 shows, to begin with, that Llama 3 outperforms Llama 2 at every digit increase which suggests that it is much more capable at handling complexity of digits. What we also notice is that there are variations in accuracy for questions between 1 and 3 digits across both models. Llama 2 stays between 60% and 70% while Llama 3 stays between 70% and 80%. From the 4th digit onwards, both models begin to show large declines in accuracy. This suggests that both models might be optimised for simpler tasks and begin to struggle when mathematical operations begin with numbers in the thousands. We have not found specific papers that have similar results with this kind of feature but we can possibly attribute this drop in performance to lesser examples of larger numbers within this context in the training data. We should note that we have a much smaller sample of 6 digit questions and so we will not be able to generalise this specific result. Additionally we were unable to compare the 17 and 18 digit questions that involved large, repeating decimals due to not accounting for floating point errors when working in a spreadsheet tool. Overall, we can see an effect of the number of digits on the model performance that begins after the 3 digit mark.

Our results for the question length feature are reflective of the results from Srivatsa and Kochmar (2024) and Xu et al. (2024) albeit at a smaller scale. Figure 11 in section 4.2.2 shows that while there are notable differences between models and Llama 3 outperforms Llama 2 on increasing question lengths, both models have performance declines as question length increases. Llama 2 has steeper declines as compared to Llama 3. Although for short questions, which are less than 100 characters, both models perform well achieving an accuracy of 89% for Llama 2 and 93% for Llama 3. Once again indicating an optimisation for simpler tasks. It seems that longer narratives are challenging for these smaller, quantized models and while, given our current research, we can not comment on the accuracy without quantization, it is not likely that the performance will improve by a large margin. We might still see similar trends with just higher accuracies. For questions longer than 200 characters, Llama 2 shows a very high MRE suggesting significant difficulty in parsing a longer question. It might be interesting to see a more granular breakdown of lengths to pinpoint exactly where the accuracies begin to diminish. But, we do see an effect of the length of the problem on model performance.

Shi et al. (2023) and Anantheswaran et al. (2024) noted the problem of irrelevant information within maths word problems being challenging for LLMs and we can see their results in our simpler setting as

well. While their work was done on significantly more complex datasets, our information relevance was limited. No question was longer than 350 characters and the amount of irrelevant information within our questions was smaller and controlled. At most there was one extra number and/or one extra subject that followed the same lexical similarity as the rest of the question. In Figure 13 in section 4.2.3 We once again notice a similar trend in that while Llama 3 is generally the better performer, questions with no irrelevant information, or in other words simple and direct questions, had both models performing well, achieving accuracies of 84% for Llama 2 and 88% for Llama 3. As simple, irrelevant information was introduced into the questions (level 0 compared to level 2), both models had a sharp drop in performance. Here we are able to show that it does not particularly matter how complex the question is.

Take for example the following two questions.

Q1: Tom has 9 yellow balloons Sara has 8 yellow balloons. How many yellow balloons do they have in total?

Q2: Sara has 31 red and 15 green balloons. Sandy has 24 red balloons. How many red balloons do they have in total?

They are quite similar questions except that in question 2, we introduce one more number. The number of operations are still the same, the number of digits do not vary within the question and the question length hardly increases. These are the kinds of questions that our LLMs were tested on as opposed questions such as the ones found in Figure 02 in section 3.1.1. Through our simple structure we are able to highlight that despite all the significant resources spent training, there is still a fundamental misunderstanding of comprehension.
We can argue that our information irrelevance feature is subjective, but to mitigate this we clearly defined rules for each category and marked them as such. We would urge to explore deeper categorization of word problems in this manner for a much more robust exploration of what kinds of irrelevant features are more challenging than others.

The most surprising result for us was that just by introducing one additional step in the arithmetic as seen in Figure 15 in section 4.2.4, going from adding two numbers to needing to add three numbers, both LLMs saw a large dip in performance. These are not questions that are designed with 4 or 5 steps with a mix of arithmetic (Figure 02 - section 3.1.1) and yet we see the smaller LLMs struggle. It then makes us question how much worse performance would get if more complex scenarios were introduced. Further

testing on non-quantized versions can be useful to understand whether the quantization is causing this drop.

All features used in our experiments have had an adverse effect on model performance.

5.1.2. Compare How the Word Problem Structure Affects Performance Between Asking for a Direct Answer Versus Asking for the Steps

To achieve this objective, we implemented a prompting structure designed to get both direct answers and the steps required to reach those answers. We utilised zero shot and few shot prompts (with 3 examples), along with instructions to avoid irrelevant information and focus solely on answering the question. Additionally, a Python parsing function was created to extract answers from the model outputs. In cases where steps were extracted, a counting function was introduced to calculate the correct answer based on the extracted steps, focusing on step accuracy rather than direct answer prediction.

While some parsing errors occurred, we manually corrected these to maintain the momentum rather than expanding the parser to account for every edge case.

As shown in Figure 07 in section 4.1, comparing all simple prompts to extract prompts across five attempts reveals that both Llama 2 and Llama 3 performed better when asked for direct answers rather than extracting steps. Llama 3 significantly outperformed Llama 2 in this aspect, achieving an accuracy of 90% compared to 67% for Llama 2. Additionally, Llama 3 had a lower MRE at 4%, compared to 12% for Llama 2, indicating that Llama 3 answers were closer to the correct values. However, when the models were asked to extract steps, both performed poorly, with Llama 2 achieving 53% accuracy and Llama 3 slightly better at 57%. While we test detailed significant results in the next section, Llama 3 exhibited a much larger performance drop when switching from simple to extract prompts, which is surprising given the significant improvement observed in simple prompts.

In Figure 10 in section 4.2.1, the analysis focused on the number of digits in a question and its impact on the model performance for both simple and extract prompts. As the number of digits increased, both models struggled more with extracting steps, with Llama 2 occasionally performing marginally better in extraction tasks without examples. The addition of examples improved performance, but results remained within a 10% difference within the models, except in cases of 5-digit questions which might be due to the sample size being smaller.

The findings suggest that the number of digits in a problem does not affect the model understanding of which numbers to add or subtract, as both models generally recognized the numbers but struggled with correctly identifying the required operations. This could also be due to the generally lexical similarity of the questions we have used. We should also note here that no model, during the extraction prompts, made up any numbers. All the numbers returned were the numbers in the questions which might explain why performance was relatively stable. The models may recognise numbers better but are worse at recognising what to do with them which leads to the lower accuracies.

Figure 12 in section 4.2.2 highlights the effect of question length on model performance. Llama 3 showed better performance on direct answers but only marginally better when asked to extract steps as compared to Llama 2. Accuracy for extracting steps dropped by a large margin when the lengths of the questions crossed 100 characters. It seems to be that the longer the question is, the more challenging it becomes for the models to correctly answer extraction prompts. This might be because it becomes harder to manage the context and track steps correctly, as extracting steps requires maintaining a sequence of operations, which becomes more difficult as the narrative length increases as there is more context to manage.

Figure 14 in section 4.2.3 examines the impact of information irrelevance on model performance. Similar to the effect of question length, Llama 3 performed significantly better on direct prompts, but its accuracy dropped sharply from 91% to 60% when irrelevant information was added, especially in extraction tasks. Llama 2 also saw a performance drop from 70% to 55%. The similar average MRE of 33% for both models in extraction tasks suggests that while Llama 3 handles direct answers much better, its ability to manage complex reasoning steps has not particularly improved over Llama 2.

Our error analysis in section 4.3.7 gives us an example to dive deeper into irrelevant information. Both questions presented in the section are mostly similar. They are about games and there is only one step to take. Q1 has one more digit to account for and the irrelevant information is similar, possibly we can argue that Q1 has slightly more irrelevant information given the longer length. However, one key difference between the questions is a temporal element. Q1 is only asking about games this year whereas Q2 is asking about games this year and last year. Looking at the extracted steps of the question it seems like both models, in Q2 interpreted something happening this year and something happening last year as things that have to be added or subtracted without consideration of the context in which the words have been used. This confusion is clear in that at times the models wanted to add the games this year and at times subtract the games this year, demonstrating a lack of contextual awareness. The question simply

asked, how many games were gone to in all, but the LLMs failed to pick up on this queue. Even in the following question:

> *Jason went to 11 football games this month. He went to 17 games last month and plans to go to 16 games next month. How many games will he attend in all?*

Both LLMs, got every extract prompt wrong, and consistently returned [(11.0, 'add'), (17.0, 'subtract'), (16.0, 'add')]. We can see an interpretation of last month as needing to be subtracted.

In Figure 16 in section 4.2.4, the comparison between simple and extract prompts for different question steps (1 step and 2 step) shows a significant decline in model performance when moving from direct answer prompts to step extraction. For 1 step questions, Llama 3 dropped accuracy from 90% to 58%, while 2 step questions saw an even larger performance decline.

Overall, we have seen that switching from simple prompts to extract prompts makes both models perform worse. We notice the same trends in extracting, as in the simple case, that as the digit, irrelevance, length and steps increase, extracting the right operations becomes harder. Work such as chain of thought prompting (Wei et al, 2022) has focused on improving the output of the answer, not by understanding the reasoning steps but by generating reasoning steps. The reasoning steps themselves can be problematic which would still lead to the incorrect answer. What we have done differently here is that we have prompted our models to specifically extract only the numbers from the question which places the focus more on contextual understanding of numbers needing to be added or subtracted as opposed to generating a long, explained output. Max new tokens were restricted, temperature was controlled and there was no repetition penalty to ensure focus on simply understanding the operations rather than creative generation which can plague longer generated outputs. In this manner we are able to examine the thoughts of the chain as opposed to the generation of a chain. We are able to see that our engineered features, except for possibly the number of digits which would require further testing, negatively affect the understanding of addition and subtraction. Working on improving the contextual understanding can in turn improve the generation of the chains which have already been shown to improve performance.

5.1.3. Understand How Performance Can Improve on Solving Simple Addition and Subtraction Word Problems

During the course of our analysis, we observed that Llama 3 generally performs better than Llama 2, but understanding which numbers need to be added or subtracted remains exceedingly difficult, even on the simplest problems. When comparing zero shot versus few shot prompting, we saw clear improvements in the extract cases, where both models were able to take advantage of the three examples shown in the prompt. However, in the simple cases, where the examples were just sample questions and their answers, neither model could leverage the examples meaningfully.

In Figure 08 in section 4.1, we present a significance table that shows P-values and their significance at the 5% and 1% levels. Notably, introducing few shot examples for simple prompts did not yield significant improvements, suggesting that our examples were not robust enough to enhance performance. However, the accuracy between Llama 2 and Llama 3 on simple prompts was significant at both levels, indicating that the improvements in Llama 3 reflect meaningful advancements in the model architecture and data. Within-model differences between zero shot and few shot extract prompts were significant for both models, meaning that examples definitely helped improve their accuracy. However, the difference between models at the 1% level for few shot extract prompts was not significant, though it was at the 5% level, implying some uncertainty as to the degree of improvement. This suggests that when it comes to understanding the steps in a simple maths word problem, Llama 2 and Llama 3 may be more similar than expected.

For few shot extract prompts, Llama 2 benefited more from the examples on questions involving question length and number of digits, sometimes even outperforming Llama 3. Llama 3, despite not taking as much advantage of the examples, consistently outperformed Llama 2, likely due to its larger parameter count and more robust training data. However, as problem complexity increased, Llama 2 was not able to consistently take advantage of examples, suggesting that either its architecture is less equipped to handle complex reasoning, or a different prompting structure (perhaps with more examples) could lead to improved performance. These findings highlight that while Llama 3 shows significant gains in performance, its intermediate reasoning abilities still struggle and might require further fine tuning.

The authors of the Llama 3 paper also noted that the model performs worse on tasks involving irrelevant information compared to relevant information, a pattern we have confirmed in our experiments. For

simple, direct questions, Llama 3 vastly outperforms Llama 2, consistently achieving greater than 80% accuracy (give or take a few points) across various features. However, when irrelevant information is introduced, while Llama 3 still performs better than Llama 2, there remains much room for improvement.

The differences between the two models largely come down to Llama 3 having an additional 1 billion parameters and a much larger, cleaner dataset. The improvements on simple prompts and the general gains on extract prompts suggest that the expanded size and better data of Llama 3 have resulted in meaningful performance improvements. However, when analysing extract prompts in detail, it becomes evident that quantization likely plays a role in dampening the benefits of Llama 3 improvements.

Huang et al. (2024) noted that quantizing Llama 3 to 4 bit was not as effective as with Llama 2, although Llama 3 still performed better overall. Their claim that Llama 3 precision is already highly optimised due to its large and clean dataset is reasonable as Dubey et al (2024) have suggested that Llama 3 has a more complex pre-training which can result in highly optimised weights. If these weights are quantized, reducing their precision can result in significant performance drops, especially since Llama 3 might already be operating near optimal performance. Attempting to fine tune on a small subset of weights post quantization would then likely not be sufficient to recover the lost performance. Dubey et al (2024) also highlighted that while Llama 3 performance does not degrade significantly at int8 precision, it remains sensitive to certain types of quantization.

Moreover, smaller models in the Llama 3 family were trained for longer than what is typically optimal. Sardana et al (2023) argue that training beyond compute-optimal thresholds (following scaling laws such as the Chinchilla scaling laws, Hoffman et al., 2022) might lead to a point where adding more data no longer produces meaningful improvements. For Llama 3, this may have resulted in overfitting, where the model learned to optimise on the pre-training data to the point that it struggles when quantized. In other words, the larger dataset and prolonged training have improved Llama 3 performance but quantization has negatively affected this optimised state possibly revealing an overfit model. Further experimentation without quantization would help clarify these findings and for now it is probably better to use an unquantized version of Llama 3.

Lastly, our analysis highlights a recurring issue with contextual misunderstanding in the models. For example, in section 4.3.7, we explored questions involving temporal elements like “this year” and “last year.” Both models consistently failed to understand the contextual relationship between these timeframes, likely due to a lack of robust contextual learning in the quantized models. While quantization

might play a role here, it is also possible that the byte pair encoding (BPE) tokenizer, used by both models, struggles to effectively capture sentence level relationships in word problems. Sentence tokenization, rather than BPE, might better capture the context and improve performance on word problems.

5.1.4. Observe Whether Smaller Models Are Good Enough in Their Reasoning Ability

While we conducted extensive experimentation with smaller models and employed similar features as seen in the literature, it remains difficult to compare the performance of these models to their larger counterparts. A key reason for this difficulty is our decision not to use more standard datasets like GSM8K. This was to avoid potential data leaks when testing with Llama 3, given that it is a newer model trained on even more public data, data that could include the standard sets. Additionally, we aimed to simplify the testing environment by focusing exclusively on addition and subtraction. Although we could have used a subset of only addition and subtraction of more standard datasets, most research papers train and report results on the entire dataset, which would have made direct comparisons difficult.

To address this limitation, we took an alternative approach. We selected a set of questions from section 4.3.8 which were questions that both Llama 2 and Llama 3 got wrong across every prompt and every attempt. We then tested these same questions on legacy GPT 4. The questions were simply presented to GPT 4, asking it to solve them and outline the steps of what should be added or subtracted, with no additional instructions provided.

Out of the 10 questions, legacy GPT 4 got 2 wrong. The following is an analysis of these errors:

Q: There are 48 erasers in the drawer and 30 erasers on the desk. Alyssa placed 39 erasers and 45 rulers on the desk. How many erasers are now there in total?

GPT 4 incorrectly added only the 30 erasers and 39 erasers on the desk, ignoring the 48 erasers in the drawer. When asked to focus specifically on what the question was asking, GPT 4 repeated the same answer, justifying that the question was focusing on the desk, which is an incorrect interpretation since the question asks for the total count of erasers.

Q: Tom bought a skateboard for $9.46 and spent $9.56 on marbles. Tom also spent $14.50 on shorts. In total, how much did Tom spend on toys?

GPT 4 added $9.46 for the skateboard and $9.56 for the marbles, ignoring the shorts. While shorts are not technically toys, the interpretation was reasonable based on the way the question was phrased, so we consider this a correct answer.

We can see that even in a simplified setup, smaller models like Llama 2 and Llama 3 struggle greatly with basic problems. Even GPT 4, a much larger model with likely over a trillion parameters, made some incorrect assumptions. This raises concerns about the practical use cases for smaller Llama models.

One potential application for smaller models could be as personal, automated tutors. However, if these models are unable to answer simple questions correctly every time, it is unlikely that parents would trust these models to tutor young children, especially those below grade 5, which is the target audience for the dataset used in this research.

In another use case, smaller Llama models could be deployed on edge devices as virtual budget tracking assistants. However, inaccuracies in tracking personal budgets could lead to significant real world consequences, which makes complete accuracy vital in such applications.

Similarly, these models could be used as health assistants, perhaps calculating medication doses based on previously taken amounts. However, in such scenarios, it would be unwise to trust a model that is not fully accurate.

Given these findings, it is clear that much more improvement is needed in these smaller models, whether by using quantized versions of larger models or developing new architectures capable of capturing the contextual complexity that current models seem to miss. More research is required to explore these possibilities and improve the performance of smaller LLMs in reasoning tasks.

5.2. Research Question

How Does the Structure of Simple Addition and Subtraction Word Problems Affect the Accuracy and Reasoning Ability of Smaller Large Language Models?

In this research we sought out to answer, what about the structure of a simple maths word problem is problematic for LLMs to solve. We wanted to focus on smaller LLMs given that a larger model might mask the problems just by nature of having a large parameter count and training dataset. While formal significance testing was not conducted on every data point, we hope that the patterns observed are enough to spark deeper conversations around context understanding. In our simplified set up we found that as question complexity increases, model accuracy decreases. Question complexity includes the number of digits involved in a question, the length of the question in terms of characters, the amount of relevant information and the number of steps needed to solve the problem. This is true not only for directly asking the question but also in extracting the steps. We have been able to observe within smaller models, specifically the Llama family, major decreases in performance with respect to answer accuracy. Deeper investigation showed that there is still contextual awareness missing in the models despite a new generation being released. More research needs to be done around understanding why this context gets missed despite increasing the size of the parameters and the training data in the smaller models. This can lead to improved datasets that hold much more varied relationships that will help future models or even fine tuning of current models to be better.

6. Evaluation, Reflections and Conclusions

While our research has provided valuable insights into the performance of smaller LLMs like Llama 2 7B and Llama 3 8B on simple word problems, there are some areas for improvement and reflection.

Our literature review gave us a wide understanding on how researchers have tried to improve outcomes of LLMs on word problems. This gave us knowledge surrounding various techniques such as using external solvers to augment the LLMs, that we have also emulated within our simplified setting. At the time of research and writing, we did feel that there was limited research around the structure of problems and the research that was done was done in isolation.

One key consideration for future work is the potential to use more standard datasets that are commonly used in the literature, such as the GSM8K dataset (Cobbe et al., 2021). Although these datasets are freely available and widely used for fine tuning, we deliberately avoided them to reduce the risk of data leaks. We wanted to ensure a simple, controlled environment. This helped us experiment but limited our ability to directly compare our results with those from other models trained on standardised data. It would be useful to carry out this research on a wider variety of lexically different problems from subsets of standard datasets. In general, it would be useful as a wider community to research more on the individual aspects such as multiplication or addition on their own, to have more information to compare against.

Using a new dataset meant verifying it as well. Manually solving and verifying the answers for 395 questions was a time consuming process, and there was a risk of introducing human error. While our calculations were done meticulously, the potential for mistakes during such an effort cannot be discounted. We could have mitigated this risk by getting external problem solvers who would solve the questions and provide an explanation. Having more people vetting the questions would reduce the risk of error.

To address subjectivity in our feature engineering, particularly in assessing "information irrelevance," we crafted thorough definitions as to why a question would be marked as part of a certain category. Our definitions can still be biassed and we could have mitigated this risk further by asking other individuals to rank the information irrelevance of questions and then take a majority vote. Given more time, we could have also conducted an in depth sensitivity analysis to assess by how much the accuracy swings by moving questions into different categories.

Our prompting techniques for extracting the right format worked well overall, and the parsing functions we developed were effective. However, there were cases where manual error correction was necessary. Manually fixing these errors was time consuming, but essential for ensuring accuracy. It might be useful for the future to have one standardised method for eliciting and parsing the outputs for LLMs. This would save researchers considerable time and reduce the fuzziness of results that could contain errors from varied and non standard approaches.

Different phrasing of the prompts might have elicited different responses from the models, and further exploration of prompt variations could have yielded more insights. Llama 2 responded well to prompting and Llama 3, in particular, tended to generate excessively long outputs, requiring us to limit token generation and add stopping prompts to ensure the model stayed focused. These adjustments helped us complete the experiments but time constraints limited our ability to test a wide range of models and prompt structures.
Sticking to the same family of models enabled us to dive deep into some architectural differences which lead to the recommendation of experimenting with a mixture of tokenizers rather than just byte pair encoding. Still, testing multiple versions of each model, including non quantized versions, could give more insights into how computation and efficiency trade offs affect performance. Future research should expand the number of models and prompts tested to give a broader picture of how different architectures handle simple word problems.

All the parsed steps from the extraction prompt were fed to a python “counting” function we created that handled the calculation. In the final version of the counter we added a non negative check to ensure calculations did not go below zero; however removing this would have allowed for greater variation in MRE scores.

An issue we encountered was the handling of large numbers, specifically questions with 17 or 18 digits. These caused errors due to floating point arithmetic when results were stored and processed through spreadsheets. In hindsight, we could have adapted these questions by reducing the number of digits or rounding them for ease, which would have prevented the floating point issues from arising. If spreadsheets are continued being used then adding a string character to the numbers will also ensure the string is kept intact. Spreadsheets, however, were useful to quickly create pivot tables and format tables that were then referenced in the research.

In conclusion, our research has demonstrated the challenges that smaller LLMs like Llama 2 and Llama 3 face when solving simple word problems, particularly in extracting addition and subtraction steps. While Llama 3 showed improvements over Llama 2 in direct prompts, both models struggled with intermediate reasoning and contextual understanding. Future work should focus on testing more standard datasets and refining and standardising prompts to improve performance and parsing. Further investigation into model architectures and tokenization techniques will be essential for enhancing small LLM capabilities.

7. Glossary

Word/Phrase	Definition
MRE	Mean Relative Error, The difference in actual and output values as a percentage of actual values
Quantization	The process to reduce the precision of the weights of a model
Accuracy	The ratio of correct answers to all answers
Zero Shot Prompts	Instructions given to a language model without examples
Few Shot Prompts	Instructions given to a language model with a few examples. Number of examples can vary.
Finetuning	Further training of a pre-trained model on a specific task
Max Tokens	A parameter that controls the maximum number of tokens a language model is allowed to generate
Temperature	A parameter that controls the randomness of predictions generated by language models
Top-P	A parameter that controls the cumulative probability of sampled tokens where tokens up to the threshold are selected from
Repetition Penalty	A parameter that controls the generation of repetitive output

8. References

1. Hosseini, M., Hajishirzi, H., Etzioni, O. and Kushman, N. (2014). Learning to Solve Arithmetic Word Problems with Verb Categorization. [online] Association for Computational Linguistics, pp.523–533. Available at: https://aclanthology.org/D14-1058.pdf.
2. Shi, F., Chen, X., Misra, K., Scales, N., Dohan, D., Chi, E., Scharli, N. and Zhou, D. (2023). Large Language Models Can Be Easily Distracted by Irrelevant Context. [online] International Conference on Machine Learning. Available at: https://www.semanticscholar.org/paper/Large-Language-Models-Can-Be-Easily-Distracted-by-Shi-Chen/3d68522abfadfc8ee6b7ec9edaaf91f1b2f38e5e
3. Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C. and Schulman, J. (2021). Training Verifiers to Solve Math Word Problems. arXiv:2110.14168 [cs]. [online] Available at: https://arxiv.org/abs/2110.14168.
4. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S. and Drame, M. (2020). Transformers: State-of-the-Art Natural Language Processing. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. doi:https://doi.org/10.18653/v1/2020.emnlp-demos.6.
5. Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M.W. and Keutzer, K. (2021). A Survey of Quantization Methods for Efficient Neural Network Inference. arXiv:2103.13630 [cs]. [online] Available at: https://arxiv.org/abs/2103.13630.
6. Dettmers, T., Pagnoni, A., Holtzman, A. and Zettlemoyer, L. (2023). QLoRA: Efficient Finetuning of Quantized LLMs. [online] arXiv.org. Available at: https://arxiv.org/abs/2305.14314.
7. 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. arXiv:2106.09685 [cs]. [online] Available at: https://arxiv.org/abs/2106.09685.
8. Wang, S., Li, S., Sun, T., Fu, J., Cheng, Q., Ye, J., Ye, J., Qiu, X. and Huang, X. (2024). LLM can Achieve Self-Regulation via Hyperparameter Aware Generation. [online] arXiv.org. Available at: https://arxiv.org/abs/2402.11251
9. Vishal Pallagani, Bharath Chandra Muppasani, Roy, K., Fabiano, F., Loreggia, A., Murugesan, K., Srivastava, B., Rossi, F., Lior Horesh and Sheth, A.P. (2024). On the Prospects of Incorporating Large Language Models (LLMs) in Automated Planning and Scheduling (APS). [online] OpenReview. Available at: https://openreview.net/forum?id=BLsvMLvuhL
10. Bastien Le Guellec, Alexandre Lefèvre, Geay, C., Shorten, L., Bruge, C., Lotfi Hacein-Bey, Philippe Amouyel, Jean-Pierre Pruvo, Kuchcinski, G. and Aghiles Hamroun (2024). Performance of an Open-Source Large Language Model in Extracting Information from Free-Text Radiology Reports. Radiology Artificial Intelligence, [online] 6(4). doi:https://doi.org/10.1148/ryai.230364.
11. Vatsal, S. and Dubey, H. (2024). A Survey of Prompt Engineering Methods in Large Language Models for Different NLP Tasks. [online] arXiv.org. Available at: https://arxiv.org/abs/2407.12994
12. Sanh, V., Webson, A., Raffel, C., Bach, S.H., Sutawika, L., Alyafeai, Z., Chaffin, A., Stiegler, A., Scao, T.L., Raja, A., Dey, M., Bari, M.S., Xu, C., Thakker, U., Sharma, S.S., Szczechla, E., Kim,

T., Chhablani, G., Nayak, N. and Datta, D. (2021). Multitask Prompted Training Enables Zero-Shot Task Generalization. arXiv:2110.08207 [cs]. [online] Available at: https://arxiv.org/abs/2110.08207.

13. Oguzhan Topsakal and T. Akinci (2023). Creating Large Language Model Applications Utilizing LangChain: A Primer on Developing LLM Apps Fast. [online] International Conference on Applied Engineering and Natural Sciences. Available at: https://www.semanticscholar.org/paper/Creating-Large-Language-Model-Applications-A-Primer-Topsakal-Akinci/c0aec04ee86c0724d61c976f19590fbe9c615723
14. Srivatsa, KV Aditya and Kochmar, E. (2024). What Makes Math Word Problems Challenging for LLMs? [online] arXiv.org. Available at: https://arxiv.org/abs/2403.11369
15. Microsoft (2024). Floating-point arithmetic may give inaccurate result in Excel - Office. [online] learn.microsoft.com. Available at: https://learn.microsoft.com/en-us/office/troubleshoot/excel/floating-point-arithmetic-inaccurate-result.
16. Hyde (2021). Google invents non-zero digits in the wee decimal places during subtraction - Google Docs Editors Community. [online] Available at: https://support.google.com/docs/thread/100646436/google-invents-non-zero-digits-in-the-wee-decimal-places-during-subtraction?hl=en
17. Xu, X., Xiao, T., Chao, Z., Huang, Z., Yang, C. and Wang, Y. (2024). Can LLMs Solve longer Math Word Problems Better? [online] arXiv.org. Available at: https://arxiv.org/abs/2405.14804
18. Anantheswaran, U., Gupta, H., Scaria, K., Verma, S., Baral, C. and Mishra, S. (2024). Investigating the Robustness of LLMs on Math Word Problems. [online] arXiv.org. Available at: https://arxiv.org/abs/2406.15444
19. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q. and Zhou, D. (2022). Chain of Thought Prompting Elicits Reasoning in Large Language Models. arXiv:2201.11903 [cs]. [online] Available at: https://arxiv.org/abs/2201.11903.
20. Qian, J., Wang, H., Li, Z., Li, S. and Yan, X. (2022). Limitations of Language Models in Arithmetic and Symbolic Induction. arXiv:2208.05051 [cs]. [online] Available at: https://arxiv.org/abs/2208.05051.
21. Ahn, J., Verma, R., Lou, R., Liu, D., Zhang, R. and Yin, W. (2024). Large Language Models for Mathematical Reasoning: Progresses and Challenges. ACL Anthology, [online] pp.225–237. Available at: https://aclanthology.org/2024.eacl-srw.17/
22. Lochan Basyal and Sanghvi, M.M. (2023). Text Summarization Using Large Language Models: A Comparative Study of MPT-7b-instruct, Falcon-7b-instruct, and OpenAI Chat-GPT Models. arXiv (Cornell University). doi:https://doi.org/10.48550/arxiv.2310.10449.
23. Jiang, J., Wang, F., Shen, J., Kim, S. and Kim, S. (2024). A Survey on Large Language Models for Code Generation. [online] arXiv.org. doi:https://doi.org/10.48550/arXiv.2406.00515.
24. Sasaki, M., Watanabe, N. and Tsukihito Komanaka (2024). Enhancing Contextual Understanding of Mistral LLM with External Knowledge Bases. Research Square (Research Square). doi:https://doi.org/10.21203/rs.3.rs-4215447/v1.

25. Li, Z., Cao, Y., Xu, X., Jiang, J., Liu, X., Teo, Y.S., Lin, S. and Liu, Y. (2024). LLMs for Relational Reasoning: How Far are We? [online] arXiv.org. Available at: https://arxiv.org/abs/2401.09042
26. Williams, S. and Huckle, J. (2024). Easy Problems That LLMs Get Wrong. [online] arXiv.org. Available at: https://arxiv.org/abs/2405.19616
27. He-Yueya, J., Poesia, G., Wang, R.E. and Goodman, N.D. (2023). Solving Math Word Problems by Combining Language Models With Symbolic Solvers. [online] arXiv.org. Available at: https://arxiv.org/abs/2304.09102
28. Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H.W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V. and Reif, E. (2022). PaLM: Scaling Language Modeling with Pathways. arXiv:2204.02311 [cs]. [online] Available at: https://arxiv.org/abs/2204.02311.
29. Dey, N., Soboleva, D., Faisal Al-Khateeb, Yang, B., Ribhu Pathria, Hemant Khachane, Muhammad, S., Chen, Z., Myers, R., Jacob Robert Steeves, Vassilieva, N., Tom, M. and Hestness, J. (2023). BTLM-3B-8K: 7B Parameter Performance in a 3B Parameter Model. [online] arXiv.org. Available at: https://www.semanticscholar.org/paper/BTLM-3B-8K%3A-7B-Parameter-Performance-in-a-3B-Model-Dey-Soboleva/44b7adbd196e69c8771734aa8c9af5fd69c04370
30. Liu, Z., Zhao, C., Iandola, F., Lai, C., Tian, Y., Fedorov, I., Xiong, Y., Chang, E., Shi, Y., Krishnamoorthi, R., Lai, L. and Chandra, V. (2024). MobileLLM: Optimizing Sub-billion Parameter Language Models for On-Device Use Cases. [online] arXiv.org. doi:https://doi.org/10.48550/arXiv.2402.14905.
31. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L. and Polosukhin, I. (2017). Attention Is All You Need. [online] arXiv.org. Available at: https://arxiv.org/abs/1706.03762.
32. Naveed, H., Khan, A.U., Qiu, S., Saqib, M., Anwar, S., Usman, M., Akhtar, N., Barnes, N. and Mian, A. (2023). A Comprehensive Overview of Large Language Models. [online] arXiv.org. doi:https://doi.org/10.48550/arXiv.2307.06435.
33. He, J., Zhou, C., Ma, X., Berg-Kirkpatrick, T. and Neubig, G. (2022). Towards a Unified View of Parameter-Efficient Transfer Learning. arXiv:2110.04366 [cs]. [online] Available at: https://arxiv.org/abs/2110.04366.
34. Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A. and Raffel, C. (2021). mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer. [online] ACLWeb. doi:https://doi.org/10.18653/v1/2021.naacl-main.41.
35. Tao, C., Hou, L., Zhang, W., Shang, L., Jiang, X., Liu, Q., Luo, P. and Wong, N. (2024). Compression of Generative Pre-trained Language Models via Quantization. [online] arXiv.org. Available at: https://arxiv.org/abs/2203.10705
36. Si, C., Gan, Z., Yang, Z., Wang, S., Wang, J., Boyd-Graber, J. and Wang, L. (2022). Prompting GPT-3 To Be Reliable. arXiv:2210.09150 [cs]. [online] Available at: https://arxiv.org/abs/2210.09150.

37. Fu, Y., Peng, H., Ou, L., Sabharwal, A. and Khot, T. (2023). Specializing Smaller Language Models towards Multi-Step Reasoning. [online] arXiv.org. Available at: https://arxiv.org/abs/2301.12726.
38. Niyogi, M. and Bhattacharya, A. (2024). PARAMANU-GANITA: Language Model with Mathematical Capabilities. [online] arXiv.org. Available at: https://arxiv.org/abs/2404.14395
39. Zhang, T., Yuan, J. and Avestimehr, S. (2024). Revisiting OPRO: The Limitations of Small-Scale LLMs as Optimizers. [online] arXiv.org. Available at: https://arxiv.org/abs/2405.10276
40. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E. and Lample, G. (2023). LLaMA: Open and Efficient Foundation Language Models. arXiv:2302.13971 [cs]. [online] Available at: https://arxiv.org/abs/2302.13971
41. Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Cappelli, A., Cojocaru, R., Hesslow, D., Julien Launay, Malartic, Q., Mazzotta, D., Badreddine Noune, Pannier, B. and Penedo, G. (2023). The Falcon Series of Open Language Models. arXiv (Cornell University). doi:https://doi.org/10.48550/arxiv.2311.16867.
42. Jiang, A.Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D.S., Casas, D. de las, 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. [online] arXiv.org. doi:https://doi.org/10.48550/arXiv.2310.06825.
43. Li, C., Wang, W., Hu, J., Wei, Y., Zheng, N., Hu, H., Zhang, Z. and Peng, H. (2024). Common 7B Language Models Already Possess Strong Math Capabilities. [online] arXiv.org. Available at: https://arxiv.org/abs/2403.04706
44. Minaee, S., Mikolov, T., Nikzad, N., Chenaghlu, M., Socher, R., Amatriain, X. and Gao, J. (2024). Large Language Models: A Survey. arXiv (Cornell University). doi:https://doi.org/10.48550/arxiv.2402.06196.
45. Jin, R., Du, J., Huang, W., Liu, W., Luan, J., Wang, B. and Xiong, D. (2024). A Comprehensive Evaluation of Quantization Strategies for Large Language Models. [online] arXiv.org. Available at: https://arxiv.org/abs/2402.16775.
46. Baras, A., Zolfi, A., Elovici, Y. and Shabtai, A. (2024). QuantAttack: Exploiting Dynamic Quantization to Attack Vision Transformers. [online] arXiv.org. Available at: https://arxiv.org/abs/2312.02220
47. Hugging Face (2023). Making LLMs even more accessible with bitsandbytes, 4-bit quantization and QLoRA. [online] Available at: https://huggingface.co/blog/4bit-transformers-bitsandbytes.
48. Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D.M., Wu, J., Winter, C. and Hesse, C. (2020). Language Models Are Few-Shot Learners. arxiv.org, [online] 4. Available at: https://arxiv.org/abs/2005.14165.
49. OpenAI (2023). GPT-4 Technical Report. arXiv:2303.08774 [cs]. [online] doi:https://doi.org/10.48550/arXiv.2303.08774.
50. Howarth, J. (2024). Number of Parameters in GPT-4 (Latest Data). [online] Exploding Topics. Available at: https://explodingtopics.com/blog/gpt-parameters.

51. Haith, A.M. and Krakauer, J.W. (2018). The multiple effects of practice: skill, habit and reduced cognitive load. Current Opinion in Behavioral Sciences, 20, pp.196–201. doi:https://doi.org/10.1016/j.cobeha.2018.01.015.
52. Yang, Z., Ding, M., Lu, Q., Jiang, Z., He, Z.-D., Guo, Y., Bai, J. and Tang, J. (2023). GPT Can Solve Mathematical Problems Without a Calculator. arXiv (Cornell University). doi:https://doi.org/10.48550/arxiv.2309.03241.
53. Nye, M., Andreassen, A., Gur-Ari, G., Michalewski, H., Austin, J., Bieber, D., Dohan, D., Aitor Lewkowycz, Bosma, M., Luan, D., Sutton, C. and Odena, A. (2021). Show Your Work: Scratchpads for Intermediate Computation with Language Models. doi:https://doi.org/10.48550/arxiv.2112.00114.
54. Zhang, M., Wang, Z., Yang, Z., Feng, W. and Lan, A. (2024). Interpretable Math Word Problem Solution Generation Via Step-by-step Planning. [online] arXiv.org. Available at: https://arxiv.org/abs/2306.00784
55. Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A. and Rodriguez, A. (2024). The Llama 3 Herd of Models. [online] arXiv.org. Available at: https://arxiv.org/abs/2407.21783.
56. Huang, W., Zheng, X., Ma, X., Qin, H., Lv, C., Chen, H., Luo, J., Qi, X., Liu, X. and Magno, M. (2024). An Empirical Study of LLaMA3 Quantization: From LLMs to MLLMs. [online] arXiv.org. Available at: https://arxiv.org/abs/2404.14047.
57. Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. de L., Hendricks, L.A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., Driessche, G. van den, Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E. and Rae, J.W. (2022). Training Compute-Optimal Large Language Models. arXiv:2203.15556 [cs]. [online] Available at: https://arxiv.org/abs/2203.15556.
58. Sardana, N., Portes, J., Doubov, S. and Frankle, J. (2023). Beyond Chinchilla-Optimal: Accounting for Inference in Language Model Scaling Laws. [online] arXiv.org. Available at: https://arxiv.org/abs/2401.00448.

9. Appendices

9.1. Appendix A - Project Proposal

MSc Data Science Dissertation Proposal

Leveraging Knowledge Graphs To Improve Large Language Models

Fasih Munir

fasih.munir@city.ac.uk

Supervisor: Tillman Weyde

1. Introduction

Large Language Models (LLM) have proven to be versatile, general purpose machines that are able to be applied to a broad range of tasks that range from text summarization, code assistance and creative work such as poetry writing. However, this broad applicability comes at the cost of being slow to train, needing extensive hardware and data and being expensive to train and operate (Naveed et al. 2023). Additionally, while LLMs are remarkable general purpose machines, their effectiveness is limited when challenged with tasks that require specific domain context such as diagnostic medicine (Ullah et al. 2024) as they are not directly trained on that data. These costs limit the extent to which a LLM can be adopted by consumers given that many consumers have specific needs which current LLMs may not be able to capture and creating their own LLM would be too costly.

Recently, Wei employed a technique called chain-of-thought (CoT) reasoning where the essential idea was for the LLM to receive prompt examples that were broken down into step by step reasoning (Wei et al. 2022). It was observed that this technique was able to broaden the ability of sufficiently large models (such as GPT-3) however limitations still exist as at times the reasoning chains may not entirely be correct for which more data would be required to fix.

Given the reasoning limitations and data requirements of LLMs, Knowledge Graphs (KG) have emerged as potential candidates that can augment the abilities of LLMs and can be grouped in 3

categories, LLM-augmented KGs, KG-enhanced LLMs and a combination of the augmentation and enhancement (Pan et al. 2023). Simply put, a KG is a graph-like structure that is built of objects and things that connect those objects (Fensel et al. 2020). For example, a KG is able to represent the idea that “a pizza has a deep pan base” (Jain et al. 2018) in the form of a triple (a structure of 3 things) where an object “pizza” is connected to an object “deep pan base” by a property, for example, “hasBase”. KGs are thus able to store knowledge and the context of that knowledge with well defined ontologies. These ontologies can be used to develop reasoning prompts, as seen by the creation of a chain-of-knowledge (CoK) (Wang et al. 2023) where the triple structure forms the base of the prompts.

1.1 Purpose and Objectives

In this paper, our purpose is to build on the idea of CoK and present an alternative solution. We will aim to answer the question:

Can we improve the quality of the output of a LLM by leveraging knowledge graphs through exploring minimal test cases that enable testing different architectures and algorithms.

The work of Wang et al. (2023) employed a generalized KG and made multiple calls to the graph for their reasoning prompts. Given that without domain specific context LLMs can struggle, we propose to use a domain specific KG, with a well structured ontology, that can be queried once to get all the related object properties. The KG we will use will be for the Dyck language which is an algebraic language with simple properties (Berstel 2007). This will be an atypical KG as the structure of Dyck words is stable with few complex relations between words such as nested words. Typical KGs have their strength in being able to capture deeper semantic relations, but the simplicity of our approach will be useful in providing a strong starting point to build from. The properties will be transformed into reasoning prompts which, an already available, LLM (such as GPT-3) will recursively ask itself and depending on either true or false, will continue or stop. This will add to the current body of research by focusing on proving or disproving the need for well constructed graphs, specific graphs for specific contexts and querying the graph once to obtain all the reasoning parameters for the LLM to prompt itself.

The following objectives and measures will outline what this research aims to accomplish:

1. Finalize a review on the current methods of enhancing LLMs with a KG

The output of this will be a thorough report on the progress of research in this domain. This will additionally help in finding other suitable evaluation metrics.

2. We will create an ontology for the Dyck language in Protege

To the best of our knowledge, no ontology formalizing the rules and grammar exists for the Dyck language. We will adhere to the principles laid out by Berstel (2007). We will then create multiple Dyck words and pass them to the reasoner in Protege and assess the assigned properties for validity. The use of this language, although not tied directly to an industry, can serve as a ground for extension into other domains such as music notation that has rules, for example, around chord structures.

3. Create a function in Python that is able to query the relevant object properties for a given LLM input and allow the pretrained LLM to recursively prompt itself

This will be considered successful if the pipeline is able to run end to end successfully with data being correctly outputted.

4. Assess different prompting use cases that include creation of words, checking the validity of words or understanding more about the language

This will be coupled with a detailed error analysis that will use examples of incorrect answers from pretrained models before employing the KG. These examples will further be tested on models enhanced by the KG to assess correctness.

Our review will enable a wider understanding of the Dyck language and its extensions, evaluation metrics and other models and domains to test on. This will lead to the creation of the first version of our ontology which in turn will be tested and evaluated in our python environment. During the process of testing we will aim to explore other architectures be it a different LLM or a different KG.

We expect this work to increase the adoption of LLMs for context specific industries such as music or medicine and further, organizations in those industries that may not have the resources to train or fine-tune their own LLM. For example, within music education, a LLM enhanced by a music theory KG would be more trusted to teach students about theory as it would need to adhere to the defined grammar within the crafted ontology. Although KGs can be difficult to create, the domain knowledge required will already be available within organizations. With training, the process of KG creation can be distributed amongst domain experts, the cost of which would be much lower compared training and maintaining a LLM.

2. Critical Context

2.1 Overview of KG LLM Combinations

Pan et al. (2023) provides a detailed overview about the different ways KGs and LLMs have been used together. Firstly, KGs can be used to improve LLMs by either being used as additional data during the training process or directly affecting the output to ensure factual accuracy. Secondly, LLMs can use their ability to parse and analyze large bodies of unstructured text to extract information that can be used to expand KGs and keep them up to date. Thirdly, both can be used in a pipeline to enhance each other where the insights from an LLM can populate a KG which can then continue providing richer context either during re-training or at the output.

Many ways to enhance KGs with LLMs have been proposed. Zhang et al. (2020) found that knowledge representations in large KGs can have degraded performance due to noisy data annotations and so they propose to use the knowledge from a pretrained language model such as BERT to enhance entity and relation representation. Xie et al. (2022) also focused on completing KGs by using pretrained language models to complete missing triples within the graph. On the

other hand, Hao et al. (2023) proposed to use a modification of the BERT model to create a KG end to end using the embeddings within the transformer model rather than using the knowledge to complete existing KGs. Wang et al. (2019) further proposed a model, KEPLER, where they use a LLM to encode entity descriptions and then simultaneously optimize the knowledge graph embeddings. While these studies are informative and have made significant advancements in problem solving such as being able to work with sparse examples (Zhang et al. 2020), all still rely on large transformer models that have compute limitations to either retrain or finetune, lack generalizability to specific contexts and have not been tested on specific contexts. These studies present valuable foundations upon which we can build our approach given we are aiming to affect only the output of an LLM which in turn should reduce the time and financial resources spent on large models.

Taking a look at how KGs enhance LLMs, this can broadly happen in 2 ways. The first is by using KGs as part of the training set for LLMs which would provide them with a deeper semantic understanding. Depending on the KGs used, these can be very useful for domain specific work. For example, Zhang et al. (2020) introduce E-BERT for e-commerce which is a LLM that, in its training phase, employs a domain specific product association knowledge graph. Their incorporation of the product KG was able to improve product classification and review question answering. However they specifically noted the difficulty in creating the product KG due to the noise (they scraped Amazon product reviews) and additionally, once again, a LLM was being trained.

In this paper, we would like to shift the focus to the second way, how KGs enhance LLMs in the way they affect the output. This method provides a layer of fact checking and context that does not come at the cost of extensive retraining. KGs can still be difficult to create due to domain knowledge and process. However, domain knowledge is less troublesome as this should be available within an industry for example medicine which leaves only the process which, as we have seen, is an open area of research. Although a manual process of creation is still possible. We will make the case for domain specific KGs that can in turn continue research in automated KG creation.

2.2. KG-Enhanced LLM Output

Before studying how KGs enhance the output of a LLM, it is important to learn how we arrived at this point. We begin by understanding the role prompting plays within LLMs. CoT prompting is a technique that significantly improves the complex reasoning performance of LLMs and can emerge naturally in "sufficiently large" models (Wei et al. 2022). The researchers were able to achieve this by simply providing the LLM with a handful of reasoning examples. These examples were manually configured and were directly able to affect the output. For example, rather than asking the model to solve a math equation, they would instead lay out the equation step by step which then allowed the model to arrive at the correct answer. Kojima et al. (2022), took a different approach to CoT, where instead of providing reasoning examples, they simply added the phrase "let's think step by step…" before each prompt. They were able to show dramatic accuracy gains on multiple benchmarks proving that simpler reasoning prompts should be explored before crafting examples or even finetuning models. Zhang et al. (2022) present AutoCoT, which builds on the usual CoT by automatically creating reasoning examples versus manually crafting them. Although they note that having diverse data is better for creating prompts with less mistakes. This brings us to Wang et al. (2023), who introduce CoK. They aimed to improve reasoning rationale by making the LLM produce evidence triples and explanation hints that come from their chosen, general, knowledge bases such as Wikidata. The triples are used to guide the LLM with factual evidence to make the reasoning chain reliable.

In this paper we will draw inspiration from CoT and CoK and present an alternative way for LLMs to reason reliably. Wang et al. (2023) noted a limitation of having finite triples. Given that they used a large and available set, and the nature of how knowledge grows, it is likely that there may never be a complete set of triples. It should further be noted that they invited domain experts to manually annotate triples to improve their reliability, highlighting the need for domain expertise. Thus, we will explore the use of a set of domain specific triples as reasoning chains. We will explore how difficult it would be to create these triples and whether they can be reliably used to improve the LLM output.

2.3 The Dyck Language

Our domain of choice, to start with, will be the Dyck language. Given that it is purely an algebraic language that consists of parentheses with simple rules such as having balanced parentheses (Berstel 2007), it will be relatively quick to build domain expertise as opposed to using a domain such as music or medicine where building the domain expertise takes years of training and finding the experts to help build an ontology can be a full research in itself. Dyck words also provide an area of sparse knowledge for which there is room to contribute with ontology creation and test the limits of LLMs with sparsity. Once we exhaust the possible use cases of this ontology, we may explore a different domain such as creating a small ontology for music theory if time permits.

3. Approaches

While a review of relevant papers has been conducted for preparing the critical context, an additional comprehensive review will be taken to establish best practices for ontology creation, to learn more about the Dyck language and its variations and assess which pretrained LLMs will work best for our research keeping in mind cost and access. We will also look at other possible avenues to assess our research outside of the ones outlined in section 3.3. Research will be conducted using online resources such as ACL Anthology and the City, University of London library. Further we will try to find other domain specific examples and compare results qualitatively.

3.1 Dyck Language Ontology

Currently, we are not aware of an ontology created for this language and will work to create it using the Protege software that is commonly used for ontologies within the KG domain. Given the relative simplicity of the language, this will form the foundation of our experimentation that can enable potential extension to other domains. Within the software we will work with Web Ontology Language (OWL) which is designed to represent knowledge and the relations between different kinds of knowledge. We will work through the explanations of Berstel (2007) and any other useful papers that we find during the comprehensive literature review and express the Dyck language using the OWL framework (Berstel's work has been used to express the thoughts around the language in this paper). This will include the creation of classes, properties and instances. Given a well defined OWL format, we will be able to then use the reasoning capabilities that are inherently present in the framework to assess the validity of Dyck words which will be useful when we are testing the accuracy of our ontology. Ontologies can be saved in various formats and can be exported to Python where we can use the ontology with RDFlib, which is a Python package designed to work with the triple structure of ontologies and graph representations. This will take our created ontology and store it in the Resource Description Framework (RDF) format. Having the ability to work with the ontology in Python will further allow us to query the graph structure using SPARQL Protocol and RDF Query Language (SPARQL) which is the language used to extract information from RDF formats. Technically it is very similar to the Structured Query Language (SQL) used to query traditional relational databases.

We will try to model the potential classes, object properties and some individuals which will inherently model the grammar of the language in some basic rules. The properties and classes will be made in the style of ontologies, for example 'locatedIn' where the first word is lowercase, there is no space and the second word is propercase (other variations exist but this is the general format). To use these properties and classes as valid LLM inputs, some simple string manipulation will be employed to turn 'locatedIn' into 'located in'.This transformation is a more common string that is used for LLM prompts and given that LLMs predict the sequence of tokens, the transformed characters are likely more useful tokens to use.

3.1.1 Potential Classes

Since the language is built on parentheses some immediate classes that can be created are for left and right parentheses. Additionally, the language can extend to other brackets and so we can include left and right curly or square brackets as well. These classes could be called ‘Characters’ with subclasses of ‘LeftParentheses’ and ‘RightParentheses’. These are also known as opening or closing symbols, which can also be possible classes.

3.1.2 Potential Object Properties

Dyck words are supposed to start and end with parentheses and these parentheses need to be balanced. This is to say that a “()” is a Dyck word but “)(“ is not a Dyck word. The reason why the second word is not a Dyck word is because the opening and closing parentheses are not balanced by their respective opening and closings. Thus, possible object properties can include ‘startsWith’ and ‘endsWith’ which relate to the opening and closing symbols respectively. Given that our ontology will also likely use brackets, other object properties such as ‘contains’ can be used to signify that Dyck words should only contain the given characters as defined by the classes. Additionally, to check for balance, a rule such as ‘isEqualTo’ could be created for correct reasoning when transforming the object property to a string input for the LLM.

3.2 Python Prompt Function and Model

Since we are trying to augment already created models, we will not be training our own. This leads us to using pretrained models such as GPT-3 which has also been used in the papers discussed in our critical context. GPT-3 is easily accessible with an OpenAI account and the free credit can be used to run small scale tests. Since it is API accessible, we will be able to work with the model in Python and have our ontology and model in the same environment. While GPT-3 is one possible model (and a useful one for the sake of comparison), our literature review will conduct a further analysis of the possible models that we can employ that will balance cost

and access. Some newer, open source models include Llama and Falcon that have the advantage of being run locally. The prompt function itself is to be designed but will likely involve the use of SPARQL and string transformations. The answer of the transformed SPARQL output will be used to then get the LLM to recursively prompt itself. This will emulate reliable reasoning. For example, when asking if a word is valid, the query may pull the related object properties for valid words, the object properties get transformed into useful string inputs which are then used as reasoning prompts for the LLM to assess the answer. Additionally, since LLM inputs can vary, there is also the possibility of converting the string input to a LLM directly to SPARQL using a tool such as AutoSPARQL. We will test with various string inputs including (but not limited to) “Is…a Dyck word?”, “What are the properties of Dyck words?” and “Can you help me write a Dyck word that has a length of 21 characters”.

3.3 Evaluation

Our evaluation will be two fold. The first will be a detailed error analysis using prompts that give incorrect responses on current models. For example, when prompting GPT-4 to define the rules of Dyck words it provided the output “Invalid Dyck Word: *XXYYXY* or (())()”. This is an incorrect output as the word it generated is in fact a valid Dyck word as it consists of parentheses that are correctly balanced. Another example is when when prompting GPT3.5 to identify if “()()()()()()()()()()()()()()((()))((((()))))()()()()()()()()()()((((()))))(())(())(())()()()()(())(())()()()()(())(())()()()()()()()()()()())(())(())(())(())((((((()))))))()()()()()()()()” was a Dyck word, it was able to say that it was not because it lacked overall balance. However, when prompted to find where it went wrong, it was not able to and explained that “Towards the end, there are multiple consecutive closing parentheses without their corresponding opening parentheses: "))))"”. This is also incorrect as can be seen in the example. Other such examples will be generated and then compared on our chosen models such as GPT-3. Our model will first be prompted on its own and then prompted with the KG in use and the outputs will be qualitatively assessed.

The second way we will assess our work is by using the same prompts that gave incorrect results but applying the techniques in the papers referenced in our critical context. Specifically, we will test with the CoT examples and single line (or zero shot) prompt of “think step by step…”. We

will look to see how well the other techniques perform in a domain specific context and how much better or worse our technique does.

If our extended literature review results in a performance metric that can be applied to our use case, that may be employed as well.

4. Work Plan

Figure 1 (right), summarizes how the work plan will flow, followed by Figure 2 (our Gantt chart below) that gives a detailed breakdown of the tasks and milestones. The project officially begins on the 1st of July 2024 and will continue for a period of 3 months. We plan to complete the final draft within Week 13 which gives an approximate 1 week buffer in case of unforeseen issues (Submission due 2nd October 2024). Bi-weekly meetings with the supervisor have been scheduled but can be ignored depending on progress. Major milestones include creating the ontology (Week 4), creating the testing environment in python (Week 5), completing evaluation (Week 9) and second draft review (Week 12).

Figure 1 - Work Flow

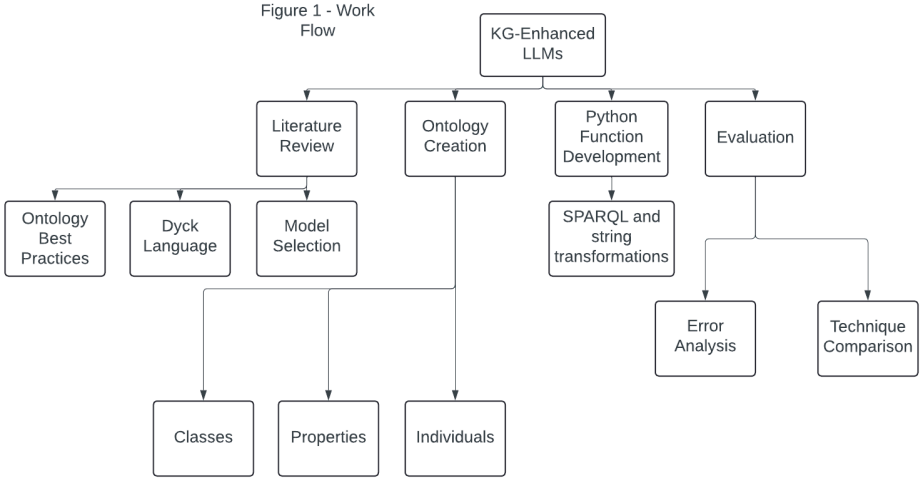


Figure 2 - Gantt Chart

Year 2024			Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14
Task	Start Date	End Date	1 Jul	8 Jul	15 Jul	22 Jul	29 Jul	5 Aug	12 Aug	19 Aug	26 Aug	2 Sep	9 Sep	16 Sep	23 Sep	30 Sep
Kickoff Meeting	1 Jul	1 Jul														
Literature Review	1 Jul	15 Jul														
Progress Meeting	15 Jul	15 Jul														
Ontology Creation	15 Jul	29 Jul														
Progress Meeting	29 Jul	29 Jul														
Python Code	29 Jul	26 Aug														
Progress Meeting	12 Aug	12 Aug														
Research Evaluation	26 Aug	2 Sep														
Progress Meeting	26 Aug	26 Aug														
First Draft	2 Sep	9 Sep														
First Draft Review	10 Sep	10 Sep														
Second Draft	11 Sep	18 Sep														
Second Draft Review	19 Sep	19 Sep														
Final Draft	20 Sep	27 Sep														
Submit Report	2 Oct	-														

5. Risks

The following risks have been identified after reading through the resource framework suggested by Dawson (2009). There are no external participants.

Risk	**Likelihood (1 - 5)**	**Consequence (1 - 5)**	**Impact (LxC)**	**Mitigation**
Unforeseen Illness	2	4	**8**	1 week buffer, extension application due to Extenuating Circumstances
Incorrect Time Estimation	2	4	**8**	1 week buffer, adapt scope of project if necessary.
Semantic Web Technology Requiring More Expertise	3	5	**15**	Thorough paper review and contact with Semantic Web Professor
API Limitations For Accessing Models	1	3	**3**	Pay subscription, use open source models on City machines.
Hardware Failure/Hardware	1	5	**5**	Cloud backups through Google

Loss				Colab and Google Workspace
Work/Data Getting Lost	1	5	5	Cloud backups through Google Colab and Google Workspace

6. References

1. Naveed, H., Khan, A.U., Qiu, S., Saqib, M., Anwar, S., Usman, M., Akhtar, N., Barnes, N. and Mian, A. (2023). A Comprehensive Overview of Large Language Models. [online] arXiv.org. doi:https://doi.org/10.48550/arXiv.2307.06435.
2. Ullah, E., Anil Parwani, Mirza Mansoor Baig and Singh, R. (2024). Challenges and barriers of using large language models (LLM) such as ChatGPT for diagnostic medicine with a focus on digital pathology – a recent scoping review. Diagnostic Pathology, 19(1). doi:https://doi.org/10.1186/s13000-024-01464-7.
3. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q. and Zhou, D. (2022). Chain of Thought Prompting Elicits Reasoning in Large Language Models. arXiv:2201.11903 [cs]. [online] Available at: https://arxiv.org/abs/2201.11903.
4. Fensel, D., Şimşek, U., Angele, K., Huaman, E., Kärle, E., Panasiuk, O., Toma, I., Umbrich, J. and Wahler, A. (2020). Introduction: What Is a Knowledge Graph? Knowledge Graphs, pp.1–10. doi:https://doi.org/10.1007/978-3-030-37439-6_1.
5. S. Jain, R. Gupta and R. K. Dwivedi, "Generating Patterns from Pizza Ontology using Protégé and Weka Tool," 2018 International Conference on System Modeling & Advancement in Research Trends (SMART), Moradabad, India, 2018, pp. 126-131, doi: 10.1109/SYSMART.2018.8746935.

6. Pan, S., Luo, L., Wang, Y., Chen, C., Wang, J. and Wu, X. (2023). Unifying Large Language Models and Knowledge Graphs: A Roadmap. [online] arXiv.org. doi:https://doi.org/10.48550/arXiv.2306.08302.
7. Wang, J., Sun, Q., Chen, N., Li, X. and Gao, M. (2023). Boosting Language Models Reasoning with Chain-of-Knowledge Prompting. [online] arXiv.org. doi:https://doi.org/10.48550/arXiv.2306.06427.
8. Jean Berstel Transductions and Context-Free Languages. (2007). Available at: https://www-igm.univ-mlv.fr/~berstel/LivreTransductions/LivreTransductions.pdf
9. Zhang, Z., Liu, X., Zhang, Y., Su, Q., Sun, X. and He, B. (2020). Pretrain-KGE: Learning Knowledge Representation from Pretrained Language Models. doi:https://doi.org/10.18653/v1/2020.findings-emnlp.25.
10. Xie, X., Zhang, N., Li, Z., Deng, S., Chen, H., Xiong, F., Chen, M. and Chen, H. (2022). From Discrimination to Generation: Knowledge Graph Completion with Generative Transformer. [online] arXiv.org. doi:https://doi.org/10.1145/3487553.3524238.
11. Hao, S., Tan, B., Tang, K., Ni, B., Shao, X., Zhang, H., Xing, E.P. and Hu, Z. (2023). BertNet: Harvesting Knowledge Graphs with Arbitrary Relations from Pretrained Language Models. [online] arXiv.org. doi:https://doi.org/10.48550/arXiv.2206.14268.
12. Wang, X., Gao, T., Zhu, Z., Zhang, Z., Liu, Z., Li, J. and Tang, J. (2019). KEPLER: A Unified Model for Knowledge Embedding and Pre-trained Language Representation. arXiv (Cornell University). doi:https://doi.org/10.48550/arxiv.1911.06136.
13. D. Zhang et al., "E-BERT: A Phrase and Product Knowledge Enhanced Language Model for E-commerce," Sep. 2020, doi: https://doi.org/10.48550/arxiv.2009.02835.
14. Kojima, T., Gu, S.S., Reid, M., Matsuo, Y. and Iwasawa, Y. (2022). Large Language Models are Zero-Shot Reasoners. arXiv:2205.11916 [cs]. Available at: https://arxiv.org/abs/2205.11916.
15. Zhang, Z., Zhang, A., Li, M. and Smola, A. (2022). Automatic Chain of Thought Prompting in Large Language Models. arXiv:2210.03493 [cs]. Available at: https://arxiv.org/abs/2210.03493.
16. Dawson, C. (2009). Projects in Computing and Information Systems A Student's Guide. [online] Available at:https://repository.dinus.ac.id/docs/ajar/Projects-in-Computing-and-Information-Systems-A-Student%e2%80%99s-Guide-2nd-Edition-April-2009.pdf.

Research Ethics, Part A: Ethics Checklist

A.1 If you answer YES to any of the questions in this block, you must apply to an appropriate external ethics committee for approval and log this approval as an External Application through Research Ethics Online - https://ethics.city.ac.uk/		*Delete as appropriate*
1.1	Does your research require approval from the National Research Ethics Service (NRES)?	**NO**
1.2	Will you recruit participants who fall under the auspices of the Mental Capacity Act?	**NO**
1.3	Will you recruit any participants who are currently under the auspices of the Criminal Justice System, for example, but not limited to, people on remand, prisoners and those on probation?	**NO**
A.2 If you answer YES to any of the questions in this block, then unless you are applying to an external ethics committee, you must apply for approval from the Senate Research Ethics Committee (SREC) through Research Ethics Online - https://ethics.city.ac.uk/		*Delete as appropriate*
2.1	Does your research involve participants who are unable to give informed consent?	**NO**
2.2	Is there a risk that your research might lead to disclosures from participants concerning their involvement in illegal activities?	**NO**
2.3	Is there a risk that obscene and or illegal material may need to be accessed for your research study (including online content and other material)?	**NO**
2.4	Does your project involve participants disclosing information about special category or sensitive subjects?	**NO**
2.5	Does your research involve you travelling to another country outside of the UK, where the Foreign & Commonwealth Office has issued a travel warning that affects the area in which you will study?	**NO**
2.6	Does your research involve invasive or intrusive procedures?	**NO**

2.7	Does your research involve animals?	**NO**
2.8	Does your research involve the administration of drugs, placebos or other substances to study participants?	**NO**
A.3 If you answer YES to any of the questions in this block, then unless you are applying to an external ethics committee or the SREC, you must apply for approval from the Computer Science Research Ethics Committee (CSREC) through Research Ethics Online - https://ethics.city.ac.uk/ **Depending on the level of risk associated with your application, it may be referred to the Senate Research Ethics Committee.**		*Delete as appropriate*
3.1	Does your research involve participants who are under the age of 18?	**NO**
3.2	Does your research involve adults who are vulnerable because of their social, psychological or medical circumstances (vulnerable adults)?	**NO**
3.3	Are participants recruited because they are staff or students of City, University of London?	**NO**
3.4	Does your research involve intentional deception of participants?	**NO**
3.5	Does your research involve participants taking part without their informed consent?	**NO**
3.6	Is the risk posed to participants greater than that in normal working life?	**NO**
3.7	Is the risk posed to you, the researcher(s), greater than that in normal working life?	**NO**
A.4 If you answer YES to the following question and your answers to all other questions in sections A1, A2 and A3 are NO, then your project is deemed to be of MINIMAL RISK. **If this is the case, then you can apply for approval through your supervisor under PROPORTIONATE REVIEW. You do so by completing PART B of this form.** **If you have answered NO to all questions on this form, then your project does not require ethical approval. You should submit and retain this form as evidence of this.**		*Delete as appropriate*

4	Does your project involve human participants or their identifiable personal data?	**NO**

9.2. Appendix B - Final Dataset

The final data containing the 395 questions and engineered features can be found in the following One Drive link

Final Dataset

9.3. Appendix C - Code

9.3.1. C1 - Initial Testing

The initial testing code can be found in the following One Drive link. It is testing code and not optimised.
Appendix C1 - Initial Testing

9.3.2. C2 - Final Experiments

The final experiments code can be found in the following One Drive link.
During our work the notebook was split into 4, these were duplicates of the same notebook with changes on how the dataframe for experimenting was filtered. This allowed us to run all experiments in parallel. Additionally, for rerunning the subset of experiments, the same notebooks were used with a different filter on the experimenting dataframe.
Only one of the notebooks is uploaded as they are essentially the same. The lessons learned from initial testing have carried forward into this notebook resulting in an optimised notebook.
Appendix C2 - Final Experiments

9.3.3. C3 - Data Cleaning Before Experimenting

This notebook was used to do initial cleaning before moving into spreadsheet software to solve the 395 questions and get them read for experimentation. It can be found in the following One Drive link.
Appendix C3 - Data Cleaning Before Experimenting

9.3.4. C4 - Significance Testing

This notebook was used just for the significance testing that we see in this report. It can be found in the following One Drive link.
Appendix C4 - Significance Testing

9.4. Appendix D - Prompts Tested

All variations and final versions of prompts can be found in the following One Drive link.

Appendix D - Prompts Tested

9.5. Appendix E - Experiment Results

All results and working can be found in the following One Drive link.

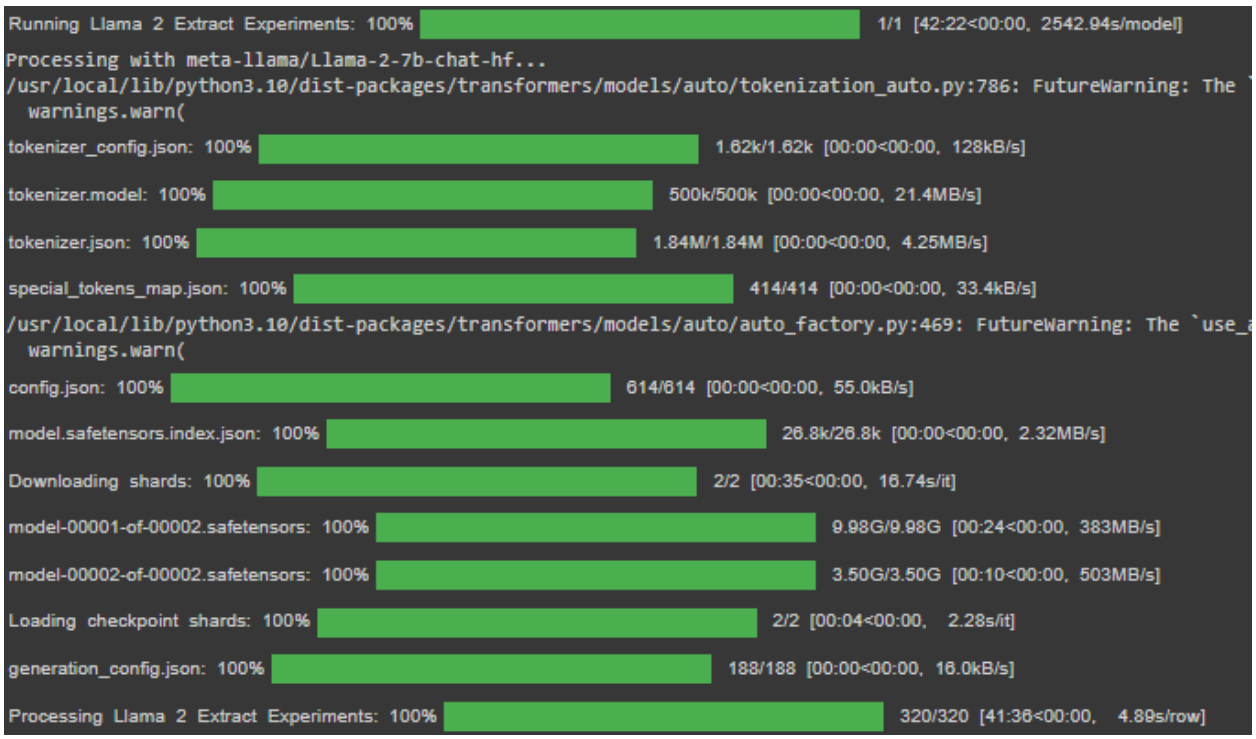
Please note that One Drive holds a link to the Google Sheet as that was the spreadsheet of choice and all formulas are available in the sheet to look at the working if needed.

To view the manual errors fixes, please filter for rows in the “All Results” tab using column B, that are marked as green or red.

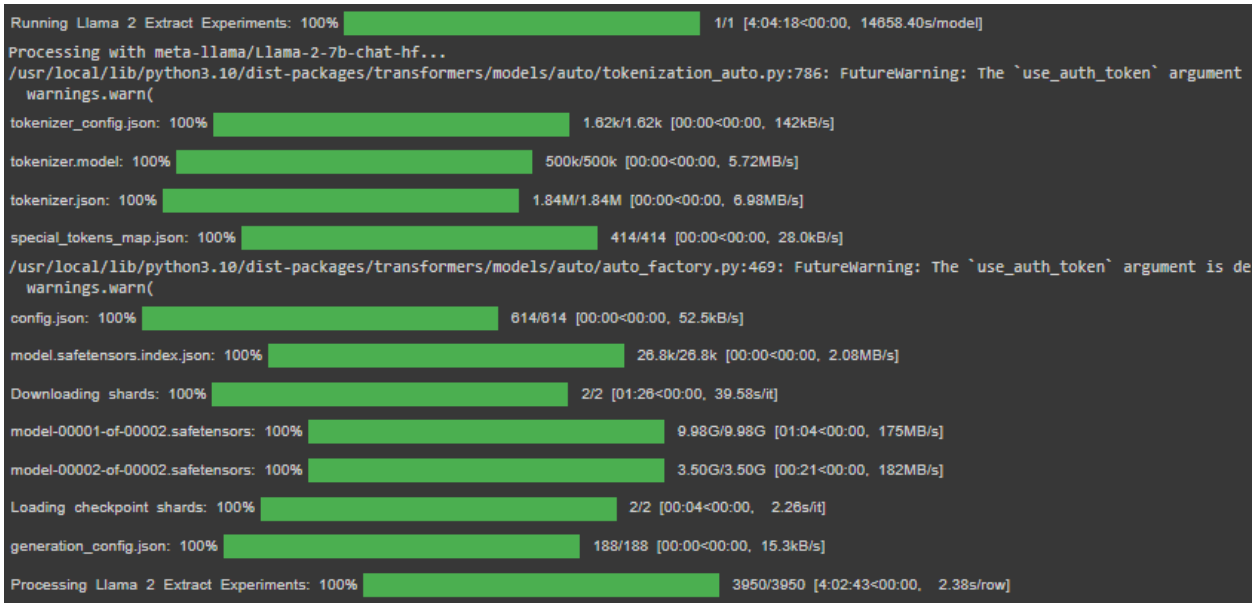
Appendix E - Experiment Results

9.6. Appendix F - Experiment Times

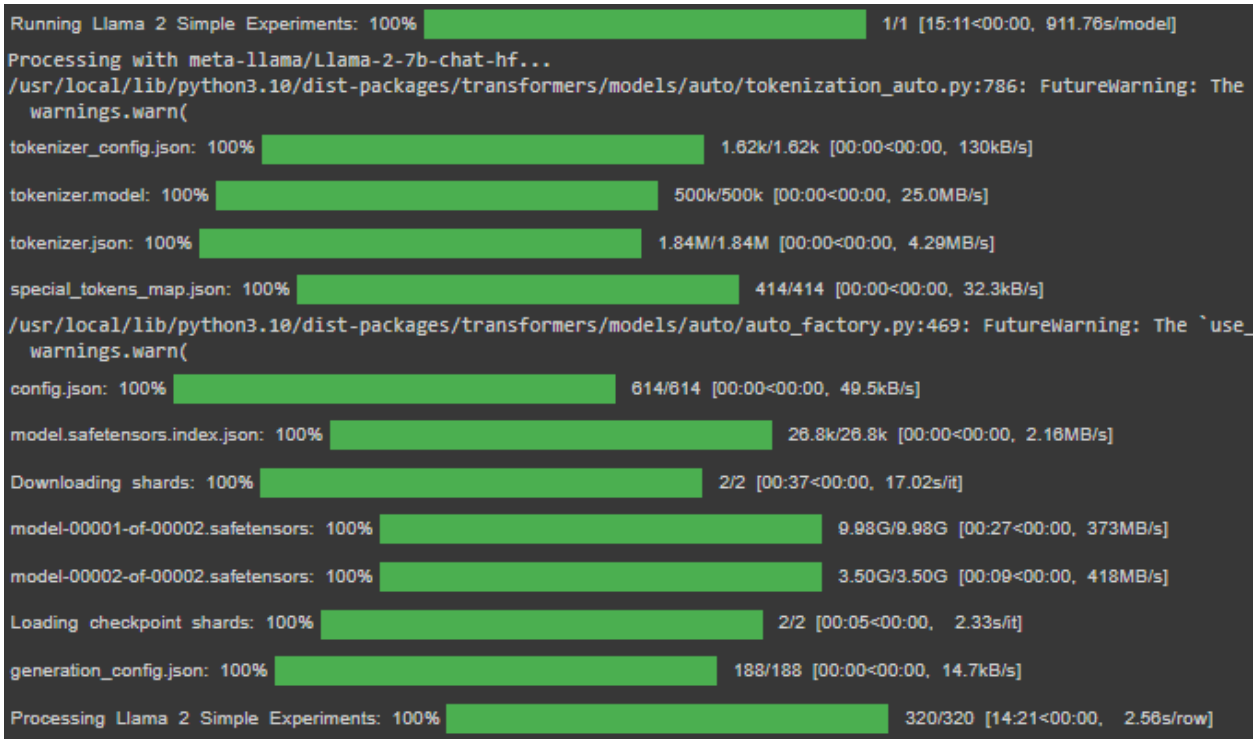
Extract experiments rerun for decimals using Llama 2



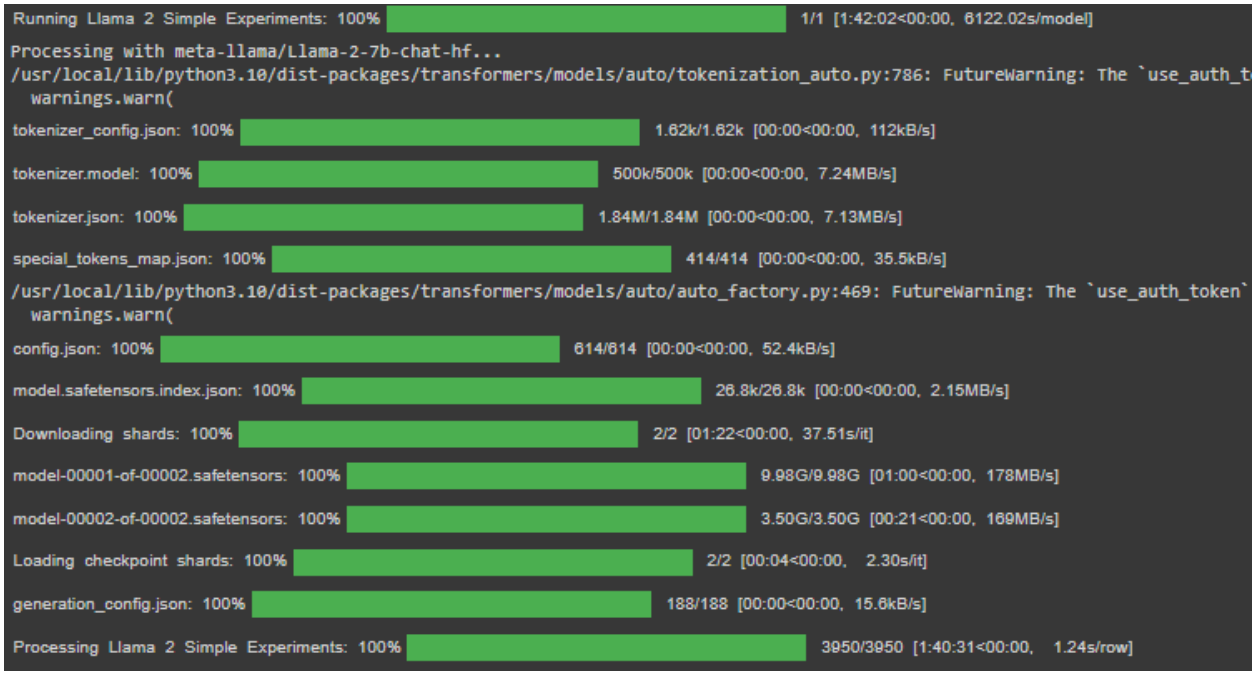
Extract experiments using Llama 2



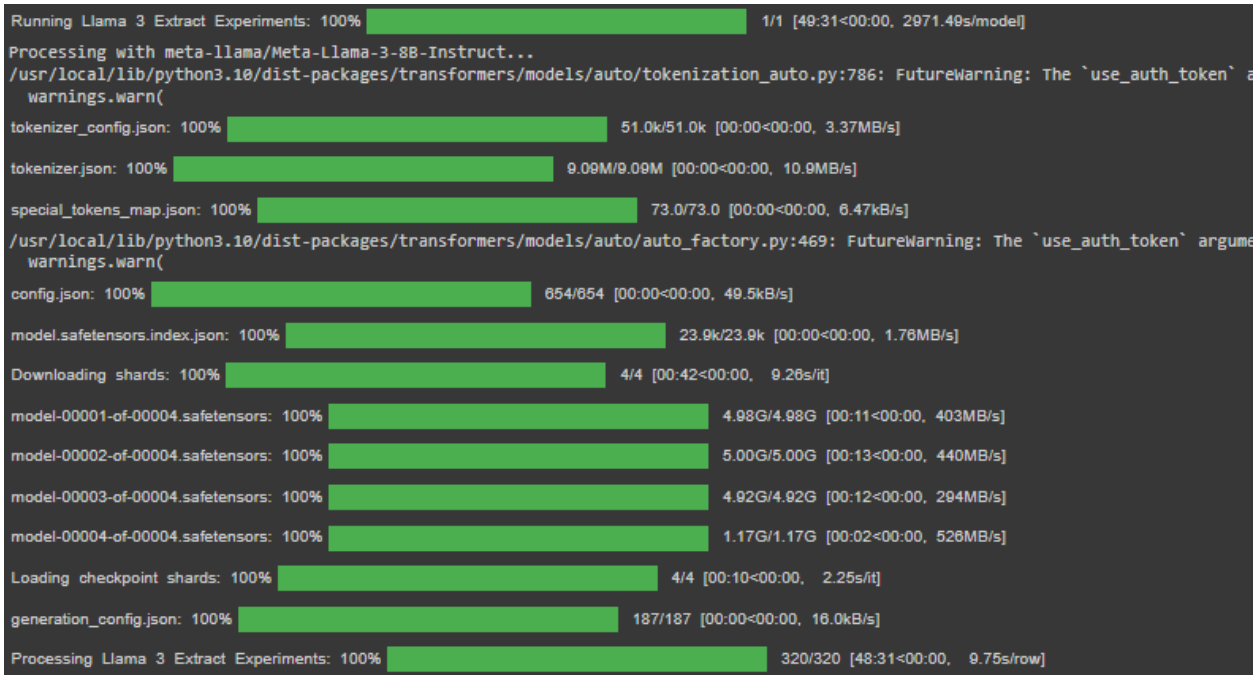
Simple experiments rerun for decimals using Llama 2



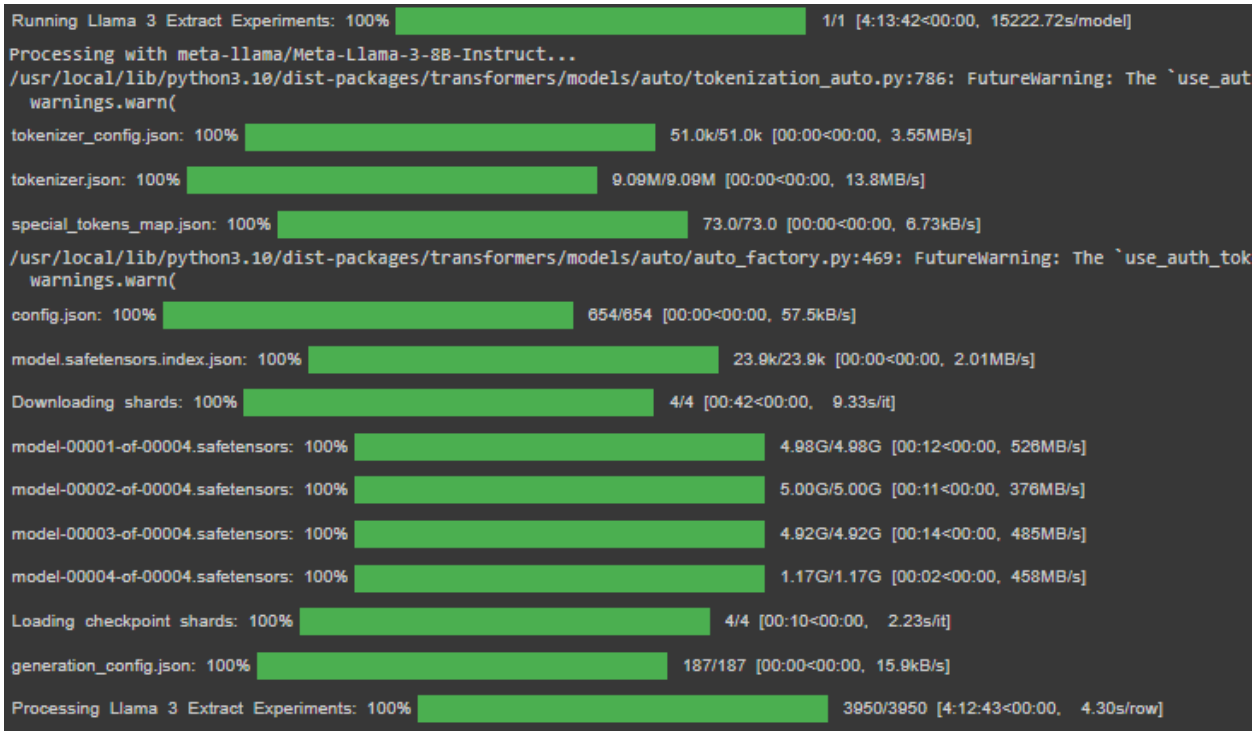
Simple experiments using Llama 2



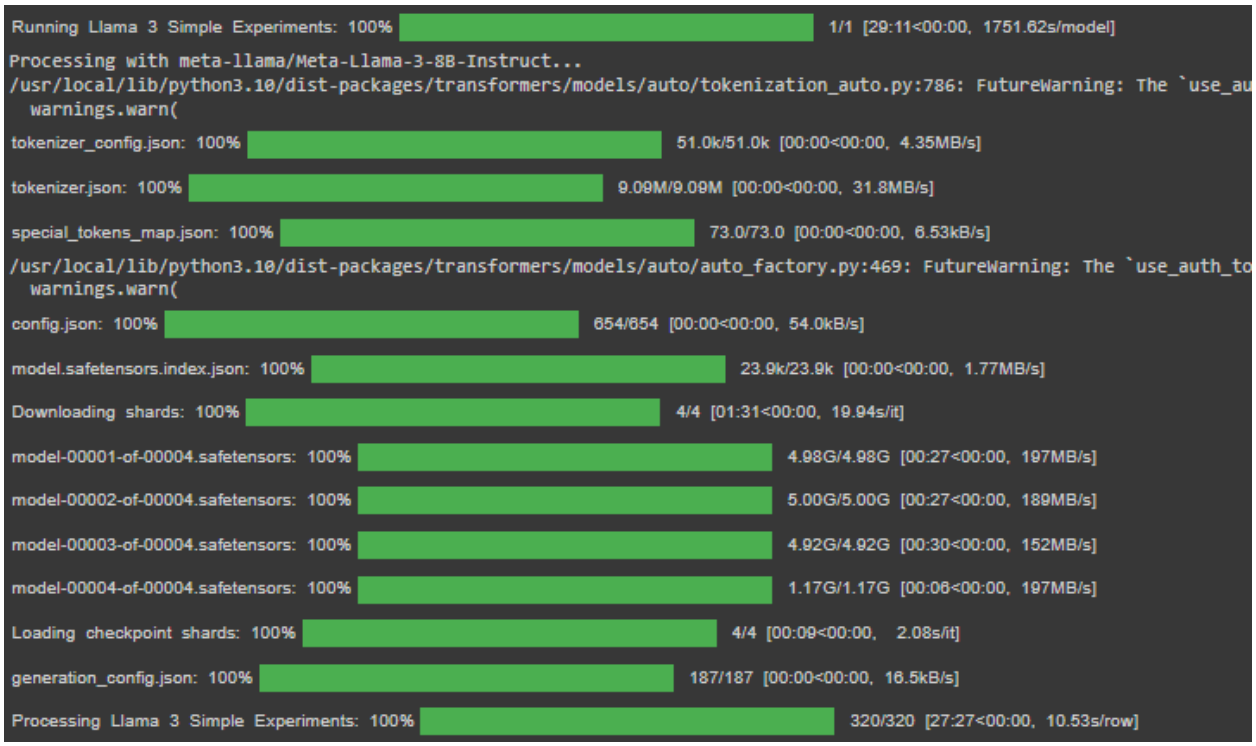
Extract experiments rerun for decimals using Llama 3



Extract experiments using Llama 3



Simple experiments rerun for decimals using Llama 3



Simple experiments using Llama 3