Numerical Comprehension in LLMs: Reframing, Masking, and Model Analysis

Mehdi Abbassi - *DSE (947511)*

Supervision

– Professor Alfio Ferrara

Introduction

This project focuses on exploring how Large Language Models (LLMs), specifically models from the BERT family, comprehend and perform arithmetic tasks involving numerals. We fine-tuned BERT-based models on tasks of single-digit addition and subtraction and then extended the testing to include both single-digit and selective double-digit numerals (between 20 and 50). This report details our methodology, experiments, and findings across different scenarios and model configurations.

Datasets

We constructed the training and test datasets using two templates of arithmetic equations:

```
{first_operand} {operator} {second_operand} {equality} {result}.
{result} {equality} {first_operand} {operator} {second_operand}.
```

The operators used were "+" (or "plus") and "-" (or "minus"), and the equality symbols were "=" or "equals". All equations were designed to ensure non-negative results. The training dataset comprised 3600 samples, with the test datasets containing 1860 single-digit and 7200 double-digit samples. The purpose of selecting only double-digit numbers between 20 and 50 is to prevent the models from encountering them during fine-tuning and to keep the results consistently within the double-digit range, avoiding triple digits.

Fine-Tuning Parameters

Fine-tuning was conducted over 10 epochs with a batch size of 32. We used an exponentially decreasing learning rate, starting at 2.0e-4 and reducing to 2.0e-5 by the last epoch.

Experimental Setup

0.1 Reframing Numerals

Purpose and Method

The primary aim of reframing numerals is to help the model understand and interpret numerals by focusing on their positional values. In typical scenarios, numerals are treated as whole entities. However, by reframing, we break down the numerals into individual digits and separate these digits with spaces.

For example, the numeral "23" is reframed as "2 3". This breakdown implies that the model does not just see "23" as a single, indivisible token but rather sees and potentially processes the '2' and '3' independently, while also being aware that they are positioned next to each other.

Expected Benefits

Positional Understanding: By presenting digits separately, the model may start recognizing that the digit '2' in the tens position has a different value and role than the '2' in the units or hundreds position. This understanding is crucial in numeracy, where the position of a digit alters its value (e.g., '20' is different from '200').

Scalability in Learning: This method could also scale better when the model encounters larger numbers or more complex numeric structures. It trains the model to not only recognize numbers but to compute using the positional values of each digit.

Implementation in Learning

During the training phase, this preprocessing strategy could be particularly effective in teaching models basic arithmetic by reinforcing the concept of positional values and how they influence the overall value of the number.

When combined with tasks such as addition or subtraction, the model can use these separated digits to perform calculations more accurately, reflecting a deeper semantic understanding rather than mere pattern matching.

By reframing numerals in this way, the model is expected to develop a more nuanced understanding of numerals, which is fundamental for performing arithmetic operations accurately and for generalizing this knowledge to handle a variety of numeric formats and calculations.

0.2 Masking Mechanisms

Default Masking Mechanism

In the original BERT architecture, during the pre-training phase, approximately 15% of the tokens in the input data are randomly selected to be masked. Of these selected tokens, 80% are replaced with a special [MASK] token, 10% are replaced with a random token, and the remaining 10% are left unchanged. This process

helps the model learn a robust representation of the language by predicting the masked tokens based on their context.

Custom Masking for Numerals

In our project, we used a custom masking mechanism to target only numerals and not other tokens (like operators, equality sign or even the period sign at the end of the equation). Model will learn not much from trying to predict equality sign or the period, but can learn from trying to predict the numerals. Besides, since we reframe numerals into separate digits, our custom masking mechanism detects spans of consecutive digits rather than random individual tokens, and randomly chooses one of the spans to mask. This means that if a number like "2 3" appears in the input, both digits ('2' and '3') would be masked (e.g., [MASK] [MASK]).

Purpose

The idea behind this approach is to emphasize the integrity and relational understanding of whole numbers in arithmetic tasks. By masking entire numerals, the model is challenged to predict not just isolated digits but the complete numeral from its surrounding arithmetic context.

Benefits of Custom Masking

This method compels the model to understand and reconstruct entire numerals, which is crucial for performing arithmetic operations accurately. It helps the model learn the significance of numerals as cohesive entities within mathematical expressions.

Contextual Learning

By forcing the model to predict entire numbers, it has to rely more heavily on the context provided by the rest of the equation. For example, in an equation "1 2 + [MASK] [MASK] = 4 8.", the model learns to infer that the masked digits must be '3' and '6'.

Application in Arithmetic

This approach is particularly useful in arithmetic contexts where the understanding of numbers and their relationships to each other (such as in addition, subtraction) is critical. The model's ability to predict masked numbers can directly translate to better performance in solving arithmetic problems where understanding the structure and placement of digits in numerals is necessary.

By customizing BERT's masking strategy in this way, the project aims to refine the model's capabilities in numerical reasoning, making it more adept at handling the specific challenges presented by arithmetic operations. This could lead to improvements in how the model processes and understands complex

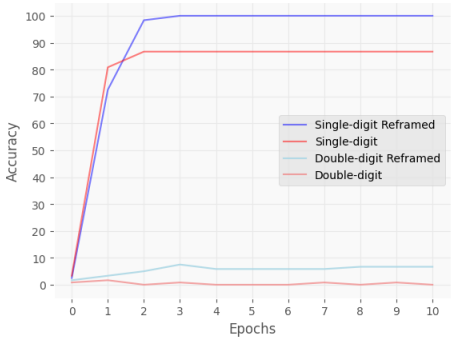

Figure 1: Impact of Numerical Reframing on Model Accuracy

numerical data, potentially enhancing its overall performance in tasks involving mathematics.

Results

Scenario 1: Impact of Reframing

Testing revealed significant differences based on whether numerals were reframed:

- Single-digits: Without reframing, accuracy reached 86.7%. With reframing, it improved to 100% by the third epoch.

- Double-digits: Without reframing, accuracy was negligible, fluctuating around 0.833%. Reframed numerals saw a modest increase to 6.667% by the eighth epoch.

Scenario 2: Custom vs. Default Masking

We compared the efficacy of our custom masking strategy against BERT's default masking:

- Single-digits: The default mechanism achieved 72.5% accuracy, while our custom approach quickly reached 100% within three epochs.

- Double-digits: Both strategies plateaued at 6.667%, but the custom masking showed more consistency.

Scenario 3: Model Comparison (BERTbase vs distilBERTbase)

We evaluated whether a larger model size correlates with better performance:

- Single-digits: BERTbase achieved a lower accuracy (91.7%) compared to distilBERTbase (100%).

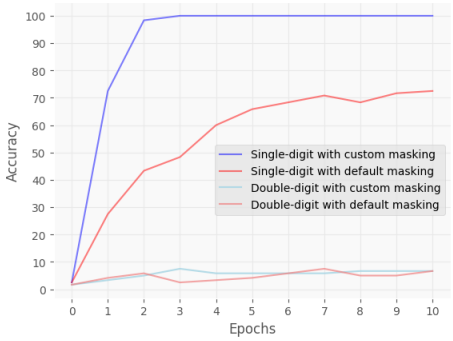

Figure 2: Accuracy Using Custom vs. Default Masking Mechanisms

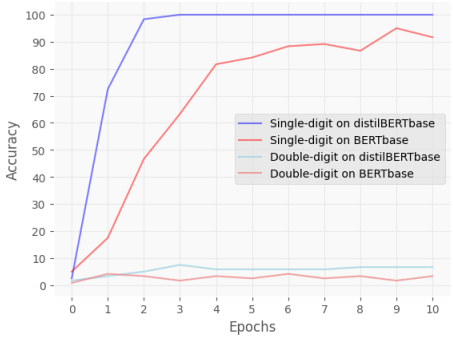

Figure 3: Performance Comparison: BERTbase vs. distilBERTbase

- Double-digits: BERTbase only managed 2.655% accuracy, significantly underperforming compared to distilBERTbase's 6.667%.

Discussion

Our findings indicate that reframing numerals significantly enhances the model's performance on familiar numerical ranges. However, the improvement is less pronounced for arithmetic operations involving unseen numeral ranges, such as double-digit numbers. This observation suggests a promising avenue for further research.

Future studies could focus on extending fine-tuning to include a broader range of numerals, such as more extensive double-digit and even triple-digit numbers. Subsequent testing could then evaluate the model's ability to generalize these skills to arithmetic operations involving previously unseen double and triple-digit numerals. This approach could potentially unlock deeper insights into the scalability of numeral comprehension and arithmetic reasoning in LLMs.

Interestingly, our custom masking strategy proved essential for achieving high accuracy, underscoring its effectiveness over the default method. Contrary to expectations, the smaller distilBERTbase model outperformed BERTbase, indicating that additional parameters do not necessarily equate to enhanced learning capability in this specific numerical context.

Conclusion

This study highlights the nuanced capabilities and limitations of LLMs in processing and performing arithmetic operations. The results will inform future research directions, particularly in refining pre-processing techniques and exploring model architectures to enhance numerical understanding in AI models.

Code Repository and Documentation

All code associated with this project, including scripts for model fine-tuning, dataset preparation, and evaluation, is available in a public GitHub repository. The repository is organized to facilitate easy navigation and replication of the project results. Each script is well-documented with comments explaining the purpose of the code blocks and instructions on how to execute them. This open access to the codebase is intended to support transparency, collaboration, and further development by the academic communities interested in advancing numerical comprehension in language models.