LREC-COLING 2024

MathNLP: The 2nd Workshop on Mathematical Natural Language Processing @LREC-COLING 2024

Workshop Proceedings

Editors
Marco Valentino

21 May, 2024
Torino, Italia

Proceedings of MathNLP: The 2nd Workshop on Mathematical Natural Language Processing @ LREC-COLING 2024

ISBN 978-2-493814-22-7
ISSN 2951-2093 (COLING); 2522-2686 (LREC)

Jointly organized by the ELRA Language Resources Association
and the International Committee on Computational Linguistics

Message from the Workshop Organizers

The articulation of mathematical arguments is a fundamental part of scientific reasoning and communication. Across many disciplines, expressing relations and interdependencies between quantities is at the centre of scientific argumentation. Nevertheless, despite its importance, the application of contemporary NLP models for inference over mathematical text remains under-explored or subject to important limitations. MathNLP represents a forum for discussing new ideas to advance research on Mathematical Natural Language Processing, welcoming novel contributions on model architectures, evaluation methods and downstream applications. MathNLP welcomed contributions of previously unpublished papers which could be either long (8 pages) or short (4 pages). MathNLP welcomed both archival and non-archival submissions. Only archival submissions have been included in the proceedings. All submissions have been peer-reviewed by 2 independent reviewers. A total of 5 papers have been accepted for presentation at the workshop. MathNLP is particularly interested in (but is not limited to) works related to the following topics:

- Neural/Neuro-symbolic architectures to support mathematical natural language inference;
- Large Language Models for Mathematics;
- Equational embeddings;
- Autoformalisation and translation from natural language to formal languages (and vice-versa);
- Linguistic analysis of mathematical discourse and argumentation relations in the context of mathematical text;
- Probing mathematical understanding of state-of-the-art models;
- Adaptation of NLP tasks for mathematical discourse;
- NLP applied to mathematics education;
- Premise selection over mathematical text;
- Understanding and typing of variables in mathematical text;
- Retrieval of equations/formulas/expressions based on textual queries;
- Retrieval of textual context based on equational queries.

MathNLP is partially funded by the Swiss National Science Foundation (SNSF) project NeuMath (200021_204617)

Organizing Committee

- Marco Valentino, Idiap Research Institute
- Deborah Ferreira, The MathWorks
- Mokanarangan Thayaparan, The MathWorks
- Andre Freitas, Idiap Research Institute & The University of Manchester

Table of Contents

Tutorial Program

An Approach to Co-reference Resolution and Formula Grounding for Mathematical Identifiers using Large Language Models

Aamin Dev, Takuto Asakura, Rune Sætre
Technical University of Munich, University of Tokyo, Norwegian University of Science and Technology (NTNU)
Germany, Japan, Norway
aamin.dev@cs.tum.edu, takuto@is.s.u-tokyo.ac.jp, satre@ntnu.no

Abstract

The high cost of human annotation labor and the advent of low-cost Large Language Models (LLMs) offer opportunities to accelerate science. This study addresses the critical challenge of disambiguating mathematical identifiers in Mathematical Language Processing (MLP), a significant step toward the effective interpretation and utilization of mathematical documents. Unlike traditional annotation methods, which are labor-intensive and prone to inconsistencies, our approach leverages the capabilities of LLMs to automate the disambiguation process. We employ state-of-the-art LLMs, including GPT-3.5 and GPT-4, and open-source alternatives to generate a dictionary for annotating mathematical identifiers, linking each identifier to its conceivable descriptions, and then assigning these definitions to the respective identifier instances based on context. We offer a novel solution to the ambiguity problem inherent in mathematical expressions by exploiting this capability of LLMs which were unknown until now. Our extensive evaluation metrics include the CoNLL score for co-reference cluster quality and semantic correctness of the annotations. We demonstrate the effectiveness of our approach in resolving identifier ambiguities, thereby making a substantial contribution to the advancement of MLP. This work paves the way for future research in automating the interpretation of complex scientific texts, highlighting the potential of LLMs in transforming the landscape of mathematical documentation analysis, expanding model options, improving annotation coverage, and reducing annotation expenses.

1. Introduction

Scientific papers in Science, Technology, Engineering, and Mathematics (STEM) domains often comprise complex mathematical formulae. The ambiguity arising from the identical use of identifiers with varied meanings based on context can perplex readers. The manual annotation of these identifiers is a tedious process, necessitating automation to facilitate co-reference resolution and formula grounding (Asakura et al., 2020), as shown in Figure 1.

In this context, we utilize the Math Identifier-Oriented Grounding Annotation Tool, MioGatto (Asakura et al., 2021), and enhance it with automation capabilities. The proposed solution involves three key stages: pre-processing, dictionary generation, and association of each occurrence. (1) Pre-processing converts the LaTeX source into a machine-readable HTML/XML format, using LaTeXML (Ginev et al., 2011). (2) Dictionary generation leverages large language models (LLMs) to construct a dictionary with mathematical identifiers as keys and their potential descriptions as corresponding values. (3) In the association phase, each occurrence of a mathematical identifier is linked with its fitting definition from the generated dictionary. For this, we take inspiration from the task of MathAlign (Alexeeva et al., 2020).

We identify critical challenges in the current manual approach and propose automated alternatives powered by LLMs. Six diverse research questions seek to gauge the LLMs' capability to automate mathematical identifier annotations concerning efficacy, context understanding, coverage, ground truth accuracy, efficiency, and potential pitfalls.

The foundational contributions of this work encapsulate extensive annotations, performance evaluation, integration with MioGatto, ground truth annotation, and CoNLL score (Pradhan et al., 2012) estimation.

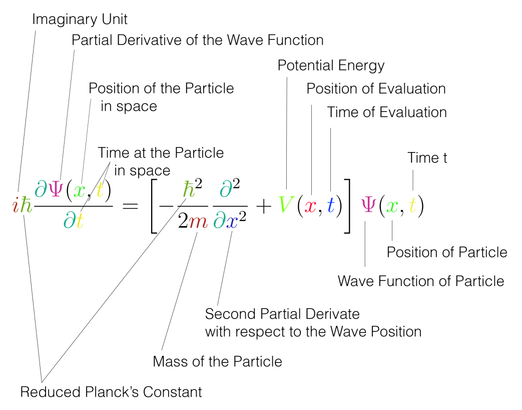
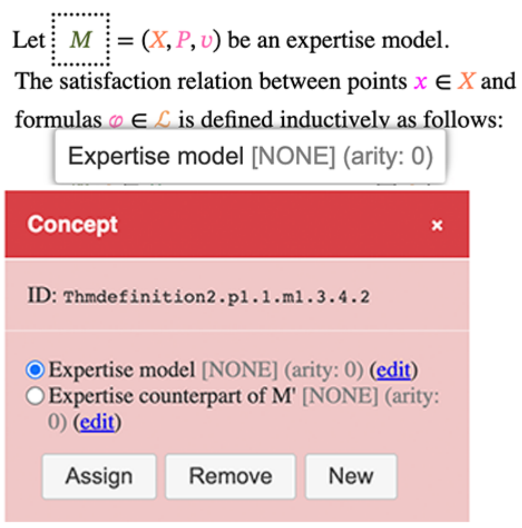

Imaginary Unit
Partial Derivative of the Wave Function
Potential Energy
Position of the Particle in space
Position of Evaluation
Time of Evaluation
Time at the Particle in space
Time t
$i\hbar \frac{\partial \Psi(x,t)}{\partial t} = \left[-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} + V(x,t)\right]\Psi(x,t)$
Position of Particle
Wave Function of Particle
Second Partial Derivate with respect to the Wave Position
Mass of the Particle
Reduced Planck's Constant

Figure 1: The challenge of disambiguating identifiers within mathematical formulae. A single variable can have multiple roles, each based on distinct definitions, creating ambiguity.


Let $M = (X, P, v)$ be an expertise model.
The satisfaction relation between points $x \in X$ and formulas $\varphi \in \mathcal{L}$ is defined inductively as follows:

Expertise model [NONE] (arity: 0)

Concept ×

ID: Thmdefinition2.p1.1.m1.3.4.2

[✓ Expertise model [NONE] (arity: 0) (edit)]
Expertise counterpart of M' [NONE] (arity: 0) (edit)

Figure 2: Screenshot of MioGatto showing its functioning where two possible annotations exist for an identifier. Hovering over the identifier shows the assigned annotation

Motivation and Problem Manual annotation is the traditional process of establishing identifier definitions, but it poses significant constraints regarding the time consumed, lack of universal accessibility, and increased financial burdens. Translating these constraints into a need for efficiency and universal availability has sparked interest in automation as the promising alternative. Nonetheless, the journey toward full automation is fraught with challenges from traditional Natural Language Processing (NLP) techniques. By investing in LLMs, we aim to harness their understanding and generation capabilities to streamline the annotation process effectively.

Research Questions Our research aims to ascertain the efficiency and feasibility of LLMs for automated mathematical identifier annotations in scientific papers. To guide the investigation, we pose six main research questions. These questions cover the effectiveness of LLMs, their ability to contextually understand mathematical identifiers, the percentage of a paper they can effectively annotate, their accuracy concerning ground truth, the impact on annotation time and cost, and the possible limitations they might present in automating this task. Answers to these research questions should offer a thorough understanding of the potential and drawbacks of LLMs in automating mathematical identifier annotations.

Contributions Our research innovatively applies LLMs like GPT-4 to automate the disambiguation and annotation of mathematical identifiers, significantly enhancing the comprehension of scientific texts. We have established a novel application of LLMs, particularly GPT-4 and its open-source counterparts, showcasing their capability to not only generate accurate and context-specific definitions for mathematical identifiers but also to do so with a high degree of semantic accuracy. We achieved remarkable semantic accuracy and CoNLL scores, demonstrating LLMs' effectiveness in complex annotation tasks beyond their initial training purposes. This work substantially reduces manual annotation efforts and costs, presenting a novel, efficient pathway for interpreting mathematical documentation. Our findings not only validate the approach with a diverse dataset of 40 scientific papers but also set a new benchmark for future explorations in Mathematical Language Processing and the automation of scientific text analysis.

2. Related Work

2.1. Formula Grounding and Tools

Our research, aimed at disambiguating mathematical identifiers, is positioned as a task within the domain of mathematical language processing (Meadows and Freitas, 2023). In particular, we focus on the automation of formula grounding, proposed by Asakura et al. (2020), using LLMs. This approach to formula grounding is distinguished by its consideration of the fact that the meanings of mathematical tokens are not constant within a document. Another known task related to formula grounding is the task called description alignment. There are several subtle variations of description alignment (Yoko et al., 2012; Stathopoulos et al., 2018; Alexeeva et al., 2020), but fundamentally, it is a simple task that involves assigning text descriptions to mathematical tokens. Solutions for description alignment have included rule-based (Alexeeva et al., 2020) and machine learning-based methods utilizing CRF, SVM (Yoko et al., 2012), Gauthian Ranking (Schubotz et al., 2016, 2017), Decision Trees, and BERT (Shan and Youssef, 2021; Lee and Na, 2022). However, the task of description alignment and its solutions typically assumes that the meaning of mathematical tokens is singular within a document, thereby failing to detect co-reference relationships among mathematical tokens. This study proposes a solution to the formula grounding task using LLM, marking the first instance to elucidate how accurately LLMs can recognize co-reference relationships among mathematical identifiers.
The proponents of the formula grounding task have introduced MioGatto (Asakura et al., 2021) as a dedicated annotation tool. Mathematical grounding involves co-reference analysis targeting mathematical expressions, but it is well-known that annotation targeting mathematical expressions and those involving co-reference rela-

tionships are both costly. Consequently, tools specialized in creating datasets for co-reference analysis (Reiter, 2018; Oberle, 2018; Bornstein et al., 2020) and those for mathematical expressions (Ginev et al., 2015; Scharpf et al., 2019) have been proposed. MioGatto is designed to efficiently perform these high-cost annotations, embodying the characteristics of both a tool for co-reference analysis annotations and one for mathematical expressions. This study proposes a method to automate this costly annotation process by generating outputs targeted at MioGatto's inputs using LLM.

2.2. The Role of LLMs and Pre-trained Frameworks

The introduction of pre-trained models like MathBERT (Peng et al., 2021) and the evaluation of GPT-3.5 (He et al., 2023) are notable developments. While MathBERT is fine-tuned for mathematical formula decoding, it does not cater to annotating mathematical identifiers, our target area. Meadows and Freitas (2023) recommended transformer models like GPT for formula retrieval. The emphasis is on quantitative reasoning using informal mathematical text, advancing the automation cause. However, our study fundamentally differs from theirs, as we primarily focus on annotation automation, not formula retrieval.

2.3. LLM Applications in the MLP Field

Recent studies, including the work by de Paiva et al. (2023), have explored the potential of LLMs in extracting mathematical concepts from textual data. Their research demonstrates the feasibility of using LLMs to automatically identify and annotate mathematical terms within a corpus of mathematical texts. By leveraging the computational power and linguistic capabilities of LLMs, researchers can improve the accuracy and efficiency of mathematical text processing, paving the way for more sophisticated applications in the field. Lai et al. (2022) and Lee and Na (2022) have attempted similarly to extract mathematical identifiers and link to their description using Named Entity Recognition (NER) and Relation Extraction (RE).

3. Methodology

This research consists of three stages for anchoring identifiers in mathematical formulae from research papers to their given descriptions: 1) preprocessing of identifiers, 2) dictionary construction, and 3) association of individual IDs to their description instances. LLMs and LaTeXML utilities are deployed for this. The result is accelerated annotation of mathematical identifiers.

1) *Pre-processing of Identifiers:* Figure 3 shows the results of LaTeX to HTML conversion using LaTeXML (Ginev et al., 2011). The format is compatible with MioGatto and allows formula grounding.

```
<p> <span> </span><span class="gd_word"
    ↪ id="S2.SS1.p1.2.2.w9">
The</span><span> </span><span class="
    ↪ gd_word" id="S2.SS1.p1.2.2.w10">
language</span><span> </span><math id="S2
    ↪ .SS1.p1.2.m2.1" class="ltx_Math"
    ↪ alttext="\mathcal{L}" display="
    ↪ inline"><semantics id="S2.SS1.p1
    ↪ .2.m2.1a"><mi class="
    ↪ ltx_font_mathcaligraphic" id="S2.
    ↪ SS1.p1.2.m2.1.1" xref="S2.SS1.p1
    ↪ .2.m2.1.1.cmml">
L</mi></semantics></math><span class="
    ↪ gd_word" id="S2.SS1.p1.2.3.w1">
is</span><span> </span><span class="
    ↪ gd_word" id="S2.SS1.p1.2.3.w2">
defined</span><span> </span><span class="
    ↪ gd_word" id="S2.SS1.p1.2.3.w3">
by</span><span> </span><span class="
    ↪ gd_word" id="S2.SS1.p1.2.3.w4">
the</span><span> </span><span class="
    ↪ gd_word" id="S2.SS1.p1.2.3.w5">
following</span><span></span><span class=
    ↪ "gd_word" id="S2.SS1.p1.2.3.w6">
grammar:</span></p>
```

Figure 3: HTML format example from "A Logic of Expertise" (Singleton, 2021) obtained by transforming the LaTeX source using LaTeXML. This machine-readable format serves as the basis for dictionary generation and for showing annotations in MioGatto.

2) *Dictionary Construction:* In the second stage, OpenAI's GPT, and other open-source LLMs are utilized to automatically generate a dictionary containing potential descriptions for each identifier. Since some lengthy papers exceed the context window of LLMs, the papers are partitioned into smaller and overlapping chunks. In this second step, the `mcdict.json` file is produced (Figure 4a) after passing each paper chunk through the LLM.
3) *Association of IDs to Description Instances:* The last stage is to deploy LLMs to annotate every instance of identifiers with suitable descriptions and store them in the `anno.json` file (Figure 4b). Quantized[1] open-source LLMs like Superhot models[2] (Chen et al., 2023) are used during this stage

[1] https://medium.com/@developer.yasir.pk/quantized-large-language-model-e80bdcb81a52
[2] https://huggingface.co/TheBloke/

```
{
    "_author": String,
    "_mcdict_version": String,
    "concepts": {
        ID: {
            "_surface": {
                "text": String,
                "unicode_name": String
            },
            "identifiers": {
                "default": [ {
                    "affixes": List,
                    "arity": Integer,
                    "description": String
                },
                ... ]
            }
        },
        ...
    }
}
```

(a) The `mcdict.json` dictionary file (shortened) contains a list of all extracted mathematical identifiers (keys) and their possible descriptions (values).

```
{
  "_anno_version": String,
  "_annotator": String,
  "mi_anno": {
    ID: {
      "concept_id": Integer, "sog": List
    },
    ...
  }
}
```

(b) The `anno.json` annotation file (shortened) holds the index of all the chosen descriptions for each identifier.

Figure 4: JSON file-structure in MioGatto.

due to their more extensive context window capabilities and well-preserved performance.
Linking between the three primary files (`source`, `mcdict`, and `anno`) is made possible by the IDs extracted during the pre-processing phase.

3.1. Pre-processing

The choice to parse the LaTeX code directly via LLMs is primarily due to the semantic richness layered over mathematical identifiers in LaTeX and its efficient token usage relative to its counterparts (see Table 1). The LaTeX code is also converted to HTML, creating a web rendering of the given paper suitable for humans but also machine-readable and useful as input to MioGatto.

`Vicuna-33B-1-3-SuperHOT-8K-GPTQ`

Despite potential setbacks due to complexities in mapping dictionary keys to their rendered instances, using LaTeXML in the pre-processing step emerged as an optimal solution after successfully isolating the mathematical symbols and finding a way to render them in a machine-readable form.

3.2. Dictionary Generation

LLMs are constructed as chat models able to output text in many languages, including some programming languages. They are also highly effective in generating a well-structured dictionary of mathematical identifiers and their possible descriptions using strategical prompting (see Figure 5). LLMs have certain limitations, notably the overflow issue—given that the length of most papers exceeds the model's context window. A master dictionary is, therefore, finally produced only after an iterative process of sub-parts generation and incorporation from all overlapping paper chunks.

```
{'role': 'system',
 'content': 'You are a helpful research
 assistant tasked with converting long
 paragraphs into a Python dictionary.
 The goal is to identify and classify
 each individual mathematical symbol,
 variable, and identifier in the text
 marked between "<| |>". The dictionary
 should store the identifiers as keys
 and their corresponding definitions as
 values in an array format.'}
```

Figure 5: System prompt for dictionary generation instructing the LLM to convert long paragraphs into a Python dictionary, emphasizing the need to identify and classify each mathematical identifier.

3.3. Association of ID to Description Occurrence

The final stage of associating extracted identifiers with their descriptions also employs LLMs, like in the dictionary generation stage. The goal is to ensure consistency and accuracy using prompting again (see Figure 6). Annotated identifiers with the chosen description act as a contextual reference for subsequent identifiers within the same context (see Figure 7).
A potential problem in providing this context could be the cascading effect of errors if a misannotation occurs. Such scenarios, although limited, are accounted for by deploying open-source LLMs that promise equal performance and extended context windows compared to the proprietary ones.
We conducted experiments with this approach on 40 academic texts using OpenAI's LLMs and other

Encoding	Formula	Tokens
LaTeX	$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$	24
ASCII Math	`x = (-b +/- sqrt{b^2 - 4ac})/(2a)`	23
XML	Too long, see Appendix A.1	387

Encoding	Formula	Tokens
LaTeX	$\oint_C \vec{B} \circ \mathrm{d}\vec{l} = \mu_0 \left(I_{enc} + \varepsilon_0 \frac{\mathrm{d}}{\mathrm{d}t} \int_S \vec{E} \circ \hat{n}\, \mathrm{d}a \right)$	84
ASCII Math	`oint_C (B . dl)=mu_0*(I_{enc} + eps_0 * d/dt * int_S (E . n_{hat}) da)`	42
XML	Too long, see Appendix A.2	929

Table 1: Token usages of three different types of encoding (LaTeX, ASCII Math, and XML). Quadratic Equation and Ampere's Circuit Law are used as examples.

```
{
 'role': 'system',
 'content': 'You are a professional
   annotator API. Your job is to select
   a fitting annotation from a dictionary
   for a mathematical identifier.'
}
```

Figure 6: System prompt for associating the identifiers to their descriptions, instructing the LLM to pick a suitable definition.

selected open-source models. One noteworthy consideration was the stochastic nature of LLMs, which necessitated the occasional repetition of experiments to obtain reliable results. The open-source models were computationally demanding despite being quantized, requiring up to 80GB of VRAM. We opted for cloud-based GPUs due to their affordability and user-friendly setup. The experiments with the open-source LLMs were conducted on pods (runpods.io) with configurations as follows:

- Vicuna-33b[3]: 1x NVidia L40 (48GB VRAM), 250GB RAM, 32vCPU at $0.69/h

- StableBeluga2[4]: 1x NVidia A100 SXM (80GB VRAM), 251GB RAM, 16vCPU at $1.84/h

Our evaluation of the models' performance was conducted using two primary metrics: the CoNLL Score for assessing the quality of co-reference resolution and a measure of semantic accuracy to evaluate the meaningfulness of the assigned definitions.

[3]https://huggingface.co/TheBloke/Vicuna-33B-1-3-SuperHOT-8K-GPTQ

[4]https://huggingface.co/TheBloke/StableBeluga2-70B-GPTQ

3.4. Reproducibility

The code can be found in Chapter 10. Use the docker command `docker run -p 4100:4100 -d ghcr.io/mathnlp-2024/miogatto:latest python -m server PAPER_ID` to launch MioGatto with the annotations produced by a given LLM. The paper IDs can be found in the `./data` folder of the repository.

4. Results

In this section, we present the results of our study on automating mathematical identifier annotations in scientific papers using LLMs. We evaluate the effectiveness of LLMs in understanding and generating descriptions for mathematical identifiers, their ability to annotate a significant portion of a paper, the accuracy of their annotations compared to ground truth, and the time and cost efficiency of the annotation process.

4.1. CoNLL Score of LLMs

We first evaluated the effectiveness of LLMs in generating descriptions for mathematical identifiers. The CoNLL metric was used to measure the quality of the co-reference clusters. The results showed that GPT-4 outperformed other models with a CoNLL score of 80.15, while other models, such as GPT-3.5-turbo and GPT-3.5-turbo-16k, had lower scores (78.51 and 79.28, respectively). Due to open-source LLMs' relatively slow speed (i.e., high run-time costs), we selected a subset of 7 of the original 40 papers. We carefully chose the papers to cover a range of attributes, including high/low CoNLL scores, high/low semantic accuracy, and short/long papers. In the smaller dataset, GPT-4 had a CoNLL score of 87.92, while StableBeluga2, an open-source LLM, had a score of 84.55, and vicuna-33b had a score of 72.44.

4.2. Coverage of Annotation

We also examined the coverage of annotation, which refers to the proportion of the paper that LLMs could successfully annotate. GPT-4 demonstrated the highest coverage, with 92.87% of the

```
{
    "role": "user",
    "content": "Given the following possible annotations: \n ```json\n"
      + definitions + "\n```
    Select the index for the most fitting description for the
    identifier <| " + match_variable + " |> from the following text."
    + possible_affixes +
    "\n Only return the value of the index and nothing else.
     Do not add any explanation otherwise the API breaks.
    The identifier has been marked with <||>.
    The text is as follows: ```txt\n" + context + "\n```"
}
```

(a) User Prompt

```
definitions = [{'index': 0,
                'identifier': 'S',
                'description': 'Soundness operator'},
               {'index': 1,
                'identifier': 'S^',
                'description': 'Dual operator of S'}]
match_variable = "S"
possible_affixes = "^"
context = "→, ↔ and truth values (⊤, ⊥) are introduced as abbreviations. We denote
    ↪ by E (Dual operator of E [^])^, <|S|>^, and A^ the dual operators corresponding
    ↪  to E,"
```

(b) User Prompt's Variables

Figure 7: Main prompt for associating the identifiers to their descriptions instructing the LLM to select the suitable definition index within the given context.

paper annotated, while GPT-3.5 had the least coverage at 90.57%. On the smaller dataset, GPT-4 had a coverage of 96.35%, StableBeluga2 a coverage of 93.17%, GPT-3.5 88.93%, and vicuna-33b a coverage of 66.18%.

4.3. Semantic Accuracy

Semantic accuracy measures the correctness of the annotations generated by LLMs. GPT-4 again outperformed other models with a weighted average semantic accuracy score of 95.70%, while other models showed lower scores, such as GPT-3.5-turbo with 84.69% accuracy. StableBeluga2 outperformed GPT-3.5 with an accuracy of 90.91%, and vicuna-33b had an accuracy of 61.58%.

4.4. Variance of Results

To account for the stochastic nature of LLMs, we conducted multiple runs of the annotation experiment on a reference paper. The results showed that GPT-3.5 had the lowest variance in CoNLL scores with a standard deviation of 1.17, indicating its stability and consistency compared to other models. GPT-3.5-turbo-16k had the highest variance with a standard deviation of 2.16,

4.5. Time and Cost Efficiency

The time and cost efficiency of the annotation process were analyzed. GPT-3.5-turbo emerged as the most time-efficient model, with an average annotation time of 2 minutes and 45 seconds per paper. However, GPT-4 had the highest cost due to its elevated token costs. The relative cost per annotation and time per annotation for each model were also calculated to provide a standardized comparison as shown in Figure 8 and 9.

5. Discussion

The results of our study demonstrate the significant potential of LLMs in automating mathematical identifier annotations in scientific papers. GPT-4, with its high CoNLL scores, comprehensive coverage of annotation, and excellent semantic accuracy, emerged as the most effective model. GPT-3.5-turbo showed the best time and cost efficiency among the models analyzed. Open-source LLMs, such as StableBeluga2, also demonstrated promising performance, sometimes even beating that of the GPT-3.5 model despite its smaller model sizes.

Open-source LLMs also have the added advantage of privacy and not having to pay for token us-

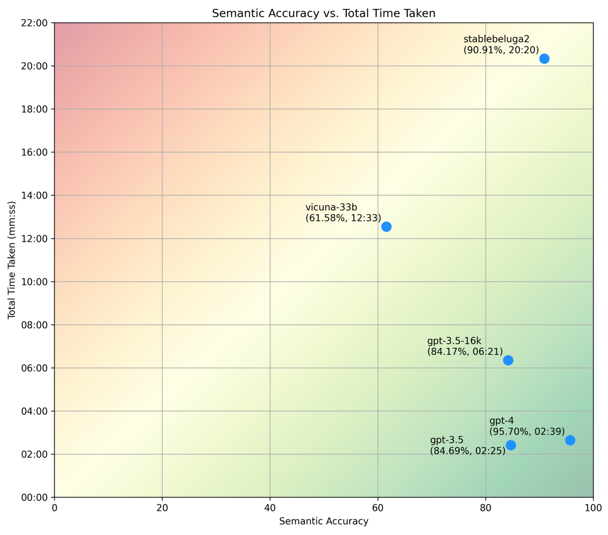


Figure 8: Scatter plots of the Total Time Taken for all five LLMs. GPT-4, despite its high costs, emerged as the most impressive model due to its superior performance, while GPT-3.5 turned out to be the most cost-effective and fastest model to operate.

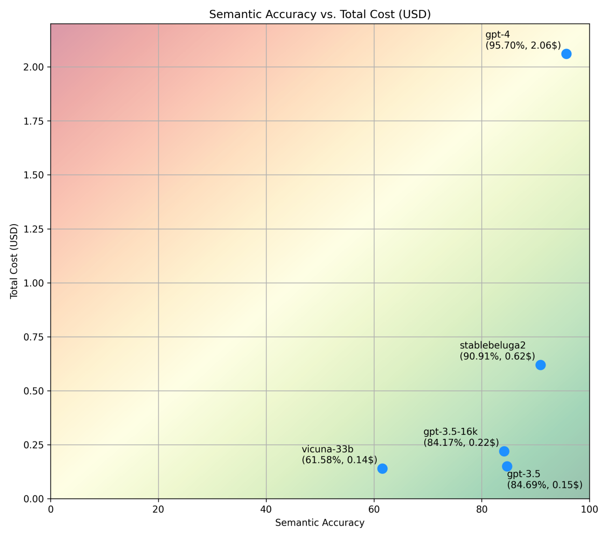


Figure 9: Total Cost (USD) vs. Semantic Accuracy for all five LLMs. GPT-4, despite its high costs, emerged as the most impressive model due to its superior performance, while GPT-3.5 turned out to be the most cost-effective and fastest model to operate.

age to OpenAI. The only cost is for their hardware and energy usage.

OpenAI's GPT models are general-purpose chat models. Their capabilities of solving nontrivial chat problems, such as formula grounding, are impressive. However, they are not very efficient at this particular purpose. Instruct models are better suited for such intricate purposes. The *instruct* nature of StableBeluga2, as opposed to the general-purpose chat model design of GPT, likely contributed to its performance in formula grounding.

While our results are promising, there are some limitations to be considered. The quality of the annotations generated by LLMs depends on the training data they were exposed to, and they may exhibit limitations when applied to domains or topics not well-represented in their training data. Additionally, the time and cost efficiency of the annotation process can vary depending on individual circumstances, such as hardware capabilities and token pricing.

In conclusion, LLMs have the potential to significantly improve the efficiency and accuracy of mathematical identifier annotations in scientific papers. Future research could focus on fine-tuning and optimizing LLMs for specific domains or developing novel techniques to further improve the automation process.

6. Conclusion

The focus of this research was to streamline the task of grounding mathematical formulae in scientific papers by automating the annotation of mathematical identifiers. This was achieved by leveraging LLMs such as GPT-3.5 and GPT-4 from OpenAI, along with open-source alternatives. Using the MioGatto Annotation Tool, we presented a technique to auto-generate a dictionary of mathematical identifiers and their associated descriptions and contextually map each identifier instance to its appropriate definition.

The evaluation metrics used in this study included the CoNLL Score for co-reference clusters' quality and semantic accuracy to measure the correctness of the annotations. Furthermore, the research examined the models for annotation coverage, annotation time, costs, and score variations due to LLMs' stochastic nature. Among the models investigated, GPT-4 excelled in its co-reference resolution ability and semantic accuracy, while GPT-3.5 was cost-effective and demonstrated the quickest performance. Open-source LLMs showed promising potential, with StableBeluga2 nearly matching the GPT models, and despite Vicuna-33 B's lower performance, it demonstrated that open-source LLMs could make meaningful contributions to this field.

The outcome of our study points towards the potential benefits of using proprietary and open-source LLMs for automating the annotation of mathematical identifiers, thus enhancing the efficiency of co-reference resolution and formula grounding. At the same time, the results highlight some challenges in this field, including the unpredictable nature of LLMs, the contextual complexity of mathematical identifiers, and the absence of a

universally accepted measure of semantic accuracy.
While the contributions of this research critically impact the mathematical language processing domain, future research based on the foundation of our work can further refine the methods, explore the use of different LLMs, and develop more sophisticated measures for semantic accuracy.

7. Future Work

Even though this study achieved high accuracy and coverage, there is room for further improvement. More sophisticated measures of semantic accuracy could be developed, and better annotation accuracy and coverage should both be achievable well below humans' price/performance ratio. Better methods to enhance the correctness of annotations should also be looked into. Introducing feedback mechanisms into the annotation process would allow continuous improvements in the annotation quality.
Our future work will explore creating a cost-effective solution for formula grounding by combining dictionary generation via open-source LLMs and better auto-association through machine learning models.

8. Optional Supplementary Materials: Appendices, Software and Data

We provide a link to an anonymous GitHub Repository but do not expect the reviewers to check it. It is, however, possible, and other researchers might want to do so (ArXiv, etc). See Section 10.

9. Bibliographical References

Maria Alexeeva, Rebecca Sharp, Marco A Valenzuela-Escárcega, Jennifer Kadowaki, Adarsh Pyarelal, and Clayton Morrison. 2020. Mathalign: Linking formula identifiers to their contextual natural language descriptions. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 2204–2212.

Takuto Asakura, André Greiner-Petter, Akiko Aizawa, and Yusuke Miyao. 2020. Towards grounding of formulae. In *Proceedings of the First Workshop on Scholarly Document Processing*, pages 138–147.

Takuto Asakura, Yusuke Miyao, Akiko Aizawa, and Michael Kohlhase. 2021. Miogatto: A math identifier-oriented grounding annotation tool. In *13th MathUI Workshop at 14th Conference on Intelligent Computer Mathematics (MathUI 2021)*.

Aaron Bornstein, Arie Cattan, and Ido Dagan. 2020. CoRefi: A crowd sourcing suite for coreference annotation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations (EMNLP 2020)*, pages 205–215.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*.

Valeria de Paiva, Qiyue Gao, Pavel Kovalev, and Lawrence S Moss. 2023. Extracting mathematical concepts with large language models. *arXiv preprint arXiv:2309.00642*.

Deyan Ginev, Sourabh Lal, Michael Kohlhase, and Tom Wiesing. 2015. Kat: an annotation tool for stem documents. In *Mathematical user interfaces workshop at CICM. Andrea Kohlhase and Paul Libbrecht, editors*.

Deyan Ginev, Heinrich Stamerjohanns, Bruce R Miller, and Michael Kohlhase. 2011. The latexml daemon: Editable math on the collaborative web. In *Intelligent Computer Mathematics: 18th Symposium, Calculemus 2011, and 10th International Conference, MKM 2011, Bertinoro, Italy, July 18-23, 2011. Proceedings 4*, pages 292–294. Springer.

Xingwei He, Zhenghao Lin, Yeyun Gong, Alex Jin, Hang Zhang, Chen Lin, Jian Jiao, Siu Ming Yiu, Nan Duan, Weizhu Chen, et al. 2023. Annollm: Making large language models to be better crowdsourced annotators. *arXiv preprint arXiv:2303.16854*.

Viet Dac Lai, Amir Pouran Ben Veyseh, Franck Dernoncourt, and Thien Huu Nguyen. 2022. Semeval 2022 task 12: Symlink-linking mathematical symbols to their descriptions. *arXiv preprint arXiv:2202.09695*.

Sung-Min Lee and Seung-Hoon Na. 2022. Jbnucclab at semeval-2022 task 12: Machine reading comprehension and span pair classification for linking mathematical symbols to their descriptions. In *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*, pages 1679–1686.

Jordan Meadows and André Freitas. 2023. Introduction to mathematical language processing: Informal proofs, word problems, and supporting tasks. *Transactions of the Association for Computational Linguistics*, 11:1162–1184.

Bruno Oberle. 2018. SACR: A drag-and-drop based tool for coreference annotation. In *Proceedings of the Eleventh International Confer-*

ence on Language Resources and Evaluation (LREC 2018).

Shuai Peng, Ke Yuan, Liangcai Gao, and Zhi Tang. 2021. Mathbert: A pre-trained model for mathematical formula understanding. *arXiv preprint arXiv:2105.00377*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Joint conference on EMNLP and CoNLL-shared task*, pages 1–40.

Nils Reiter. 2018. CorefAnnotator: a new annotation tool for entity references. In *Abstracts of EADH: Data in the Digital Humanities*.

Philipp Scharpf, Ian Mackerracher, Moritz Schubotz, Jöran Beel, Corinna Breitinger, and Bela Gipp. 2019. Annomathtex-a formula identifier annotation recommender system for stem documents. In *Proceedings of the 13th ACM conference on recommender systems*, pages 532–533.

Moritz Schubotz, Alexey Grigorev, Marcus Leich, Howard S. Cohl, Norman Meuschke, Bela Gipp, Abdou S. Youssef, and Volker Markl. 2016. Semantification of identifiers in mathematics for better math information retrieval. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '16*, pages 135–144.

Moritz Schubotz, Leonard Krämer, Norman Meuschke, Felix Hamborg, and Bela Gipp. 2017. Evaluating and improving the extraction of mathematical identifier definitions. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction: 8th International Conference of the CLEF Association, CLEF 2017, Dublin, Ireland, September 11–14, 2017, Proceedings 8*, pages 82–94. Springer.

Ruocheng Shan and Abdou Youssef. 2021. Towards math terms disambiguation using machine learning. In *Intelligent Computer Mathematics — 14th International Conference, CICM 2021, Timisoara, Romania, July 26–31, 2021, Proceedings*, volume 12833, pages 90–106. Springer International Publishing.

Joseph Singleton. 2021. A logic of expertise. *arXiv preprint arXiv:2107.10832*.

Yiannos Stathopoulos, Simon Baker, Marek Rei, and Simone Teufel. 2018. Variable typing: Assigning meaning to variables in mathematical text. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 303–312.

Kristianto Giovanni Yoko, Minh-Quoc Nghiem, Yuichiroh Matsubayashi, and Akiko Aizawa. 2012. Extracting definitions of mathematical expressions in scientific papers. *Proceedings of the 26th Annual Conference of Japanese Society for Artificial Intelligence (JSAI 2012)*.

10. Language Resource References

Code: https://anonymous.4open.science/r/grounding-of-formulae-mathnlp-2024-27C5/

A. XML Encoding Example

This section shows the complicated formatting of XML which renders it as an unsuitable type for input to LLMs.

A.1. Quadratic Equation

The XML encoding of $x = \frac{-b\pm\sqrt{b^2-4ac}}{2a}$ is as follows:

```
<math display="block"
      style="display:block math;">
  <mrow>
    <mi>x</mi>
    <mo>=</mo>
    <mfrac>
      <mrow>
        <mo>-</mo>
        <mi>b</mi>
        <mo>±</mo>
        <msqrt>
          <mrow>
            <msup>
              <mi>b</mi>
              <mn>2</mn>
            </msup>
            <mo>-</mo>
            <mn>4</mn>
            <mi>a</mi>
            <mi>c</mi>
          </mrow>
        </msqrt>
      </mrow>
      <mrow>
        <mn>2</mn>
        <mi>a</mi>
      </mrow>
    </mfrac>
  </mrow>
</math>
```

A.2. Ampere's Circuit Law

The XML encoding of $\oint_C \vec{B}.d\vec{\ell} = \mu_0(I_{enc} + \epsilon_0 \frac{d}{d_t}\int_S \vec{E}.\hat{n}\,da)$ is as follows:

```
<math>
  <mrow>
    <msub>
      <mo movablelimits="false"> </mo>
      <mi>C</mi>
    </msub>
    <mover>
      <mi>B</mi>
      <mo stretchy="false"
      style="transform:scale(0.75)
      translate(10%, 30%);">→</mo>
    </mover>
    <mo> </mo>
  </mrow>
  <mrow>
    <mrow>
      <mi mathvariant="normal">d</mi>
    </mrow>
    <mover>
      <mi>l</mi>
      <mo stretchy="false"
      style="transform:scale(0.75)
      translate(10%, 30%);">→</mo>
    </mover>
    <mo>=</mo>
  </mrow>
  <mrow>
    <msub>
      <mi> </mi>
      <mn>0</mn>
    </msub>
    <mrow>
      <mo fence="true" form="prefix">(</mo>
      <msub>
        <mi>I</mi>
        <mtext>enc</mtext>
      </msub>
      <mo>+</mo>
      <msub>
        <mi>   </mi>
        <mn>0</mn>
      </msub>
      <mfrac>
        <mrow>
          <mi mathvariant="normal">d</mi>
        </mrow>
        <mrow>
          <mrow>
            <mi mathvariant="normal">d</mi>
          </mrow>
          <mi>t</mi>
        </mrow>
      </mfrac>
      <msub>
        <mo movablelimits="false"> </mo>
        <mi>S</mi>
      </msub>
      <mover>
        <mi>E</mi>
        <mo stretchy="false"
        style="transform:scale(0.75)
        translate(10%, 30%);">→</mo>
      </mover>
      <mo> </mo>
      <mover>
        <mi>n</mi>
        <mo stretchy="false"
        style="math-style:normal;
        math-depth:0;">^</mo>
      </mover>
      <mspace width="0.2778em"></mspace>
      <mrow>
        <mi mathvariant="normal">d</mi>
      </mrow>
      <mi>a</mi>
      <mo fence="true" form="postfix">)</mo>
    </mrow>
  </mrow>
</math>
```

Fluid Dynamics-Inspired Emotional Analysis in Shakespearean Tragedies: A Novel Computational Linguistics Methodology

Davide Picca
University of Lausanne
Switzerland
davide.picca@unil.ch

Abstract

This study introduces an innovative method for analyzing emotions in texts, drawing inspiration from the principles of fluid dynamics, particularly the Navier-Stokes equations. It applies this framework to analyze Shakespeare's tragedies "Hamlet" and "Romeo and Juliet", treating emotional expressions as entities akin to fluids. By mapping linguistic characteristics onto fluid dynamics components, this approach provides a dynamic perspective on how emotions are expressed and evolve in narrative texts. The results, when compared with conventional sentiment analysis methods, reveal a more detailed and subtle grasp of the emotional arcs within these works. This interdisciplinary strategy not only enriches emotion analysis in computational linguistics but also paves the way for potential integrations with machine learning in NLP.

Keywords:
Fluid Dynamics of Emotions, Computational Linguistics, Narrative Dynamics

1. Introduction

Traditional sentiment and emotion analysis often rely on discrete classification and polarity scoring. However, these methods may not fully encapsulate the dynamic and evolving nature of language. In particular, recent developments have leveraged sophisticated machine learning models, including deep learning and neural networks, to enhance the accuracy of sentiment and emotion classification. Despite these advancements, there are notable limitations in the discrete classification and polarity scoring approach. As highlighted by (Pereira et al., 2022), such approaches struggle with understanding context and cultural nuances. Sentiments can be highly context-dependent, and what is considered positive in one culture may be negative in another. Moreover, language is dynamic and constantly evolving, with new slang, idioms, and expressions emerging regularly. Discrete classification systems can quickly become outdated, failing to capture these evolving nuances.

This paper aims to bridge the gap in existing research by utilizing a method inspired by the Navier-Stokes equations to forge a more complex examination of emotion dynamics in textual analysis. Underlying this proposal is the claim that fluid dynamics principles - such as velocity, density, and pressure - can be metaphorically adopted to shed light on the evolution of emotions in the context of narrative texts. This approach is designed to uncover those subtle shifts in emotional tides that might elude traditional emotion analysis techniques. The rest of the paper includes a presentation of the theoretical foundations and practical application, with a focus on the analysis of some of Shakespeare's tragedies, such as 'The Tragedy of Hamlet' and 'The Tragedy of Romeo and Juliet'.

Related Works

The exploration of emotion analysis in NLP through the application of the principles of fluid dynamics, in particular the Navier-Stokes equations, is an interdisciplinary research effort. Since, to the best of our knowledge, no other such attempt exists, this literature review lays the foundation for understanding both domains, emphasizing the potential of their integration.

If we look at the field of NLP, emotion and sentiment analysis have emerged as a significant area, focusing broadly on the computational process of identifying and categorizing emotions in text spanning from social media (Drus and Khalid, 2019; Rodríguez-Ibánez et al., 2023) to classic texts (Picca and Richard, 2023; Pavlopoulos et al., 2022; Picca and Pavlopoulos, 2024). Numerous studies have employed machine learning techniques for sentiment classification, with early efforts leading the way in this approach (Pang et al., 2002). Researchers have also explored the use of recursive deep models, such as those highlighted by (Socher et al., 2013), to enhance the depth of sentiment analysis at the sentence level. More recent comprehensive reviews suggest a trend towards multimodal sentiment analysis, which combines methods from single modes into more elaborate frameworks. These trends point to an evolving landscape in sentiment analysis, now including various modalities and integrating advanced technologies like GPT algorithms (Lu et al., 2023). Moreover, Emotion Recognition in Conversations (ERC) has evolved rapidly, integrating advances from various disciplines. Pereira *et al.* (Pereira et al., 2022) surveyed text-based ERC, focusing on challenges such as conversational context modeling and emotion shifts in multiparty interactions. The paper shows that, in the field of sentiment analysis, accurately capturing the dynamic nature of sentiments in narratives remains a challenge. Traditional methods often treat sentiments within texts as unchanging, overlooking their evolving characteristics (Liu, 2012).

On a different note, the field of fluid dynamics extensively uses the Navier-Stokes equations to simulate the behavior of fluids under various forces. These equations are crucial for understanding fluid motion and have been applied across diverse disciplines, including meteorology and biomedical engineering (Acheson, 1990). Their ability to accurately describe complex fluid behaviors, such as turbulence and flow patterns, is well-recognized (Tennekes and Lumley, 1972). Recent developments in fluid dynamics have expanded the scope of Navier-Stokes theory to higher-rate conditions, enhancing our comprehension of fluid behaviors under different flow scenarios (Pahlani et al., 2023).

The idea of adapting these equations to analyze emotional flow in textual content is a novel approach. While direct empirical studies in this specific area are relatively scarce, the conceptual similarities are notable. For instance, just as external factors influence fluid motion in fluid dynamics, external variables may similarly affect the flow of emotions in text. This perspective aligns with research exploring how cultural and contextual elements influence emotion formation and expression.

2. Conceptual Mapping

The initial phase of the experiment involves establishing a conceptual parallel between the elements of fluid dynamics and linguistic properties in text. The key elements and their linguistic analogs are:

1. **Velocity ($\vec{u}$)**: This could represent the "speed" at which emotion is propagating through the text. For example, a rapid shift from positive to negative sentiment could be considered a high "velocity". It represents how the current state of emotion (*e*) influences its rate of change (or flow). A rapid change in emotion in the text could lead to a significant value in this term, analogous to high velocity in fluid dynamics.

2. **Time (t)**: This remains as the position of a sentence or word in the text, serving as a temporal marker. Time is represented by the position of words in the text.

3. **Density (ρ)**: This could represent the "density" of specific emotions in a given section of text. A paragraph filled with positive words would have a high "density" of positive emotion. In our context, the "density" of sentiment is computed by summing the absolute values of sentiment scores as provided by the SenticNet lexicon (Cambria et al., 2022).

4. **Pressure (p)**: This could be analogous to the intensity of an emotion. Stronger words ("love," "hate") exert more "pressure" than weaker ones ("like," "dislike"). In this context, the "pressure" exerted by specific words in the text is based on the sum of their sentiment scores if the target word is present in a user-generated list.

5. **Viscosity (ν)**: This could represent the resistance to the flow of emotion, perhaps due to the complexity or ambiguity of the text. In this context, the "viscosity" is evaluated by the standard deviation of sentiment scores as provided by the SenticNet lexicon, indicating resistance in emotional flow.

6. **External Force ($\vec{g}$)**: This could be external factors like cultural context or the influence of preceding text segments. For our experiment, this measures the "external force" based on the polarity value of the text as provided by the SenticNet lexicon.

The Navier-Stokes-inspired equation is then modified to align with these linguistic properties. The modified equation for sentiment flow ($\vec{e}$) in the text is:

$$\frac{\partial \vec{e}}{\partial t} + (\vec{e} \cdot \nabla)\vec{e} = -\frac{1}{\rho_{\text{sent}}} \nabla p_{\text{sent}} + \nu_{\text{sent}} \nabla^2 \vec{e} + \vec{g}_{\text{context}} \quad (1)$$

Where ρ_{sent}, p_{sent}, and ν_{sent} represent the density, pressure, and viscosity of sentiment, respectively, and $\vec{g}_{\text{context}}$ is the external contextual force.

2.1. Operationalization of Sentiment Analysis Variables

This section delineates the operationalization of variables for analyzing sentiment dynamics in texts, as implemented for this experiment.

Density(ρ): The function computes the "density" of emotions in a segment of text by summing the absolute values of sentiment scores. This approach quantifies the overall emotional "weight" or intensity in a section of text without regard to the sentiment's polarity (positive or negative). The density is higher when there are strong emotional words, regardless of whether they convey positive or negative sentiments. So for example, in analyzing a paragraph filled with intensely emotional words (both positive and negative), the density calculation aggregates these sentiment scores to reflect a high "density" of emotional content.

Pressure(p): This "pressure" is determined by checking for the presence of predefined keywords that are expected to have a strong emotional impact (e.g., "love," "hate") and summing up their sentiment scores. The presence of these keywords in the text increases the "pressure" of the sentiment, analogous to how stronger words exert more influence on the emotional tone of the text. For example, a sentence that includes the word "hate" (assuming "hate" is part of the keywords list), and if the sentiment score associated with "hate" is high, the sentiment pressure for that sentence increases, reflecting the intensity of the emotion conveyed.

Viscosity(ν): "Viscosity", in this context, represents the variability or dispersion of sentiment scores, which could be interpreted as the text's resistance to a uniform emotional flow. High viscosity indicates a wide range of sentiment scores, suggesting complexity or ambiguity in the text's emotional content. For example, in our context, in a narrative with a mix of high and low sentiment scores—indicating fluctuating emotions—the viscosity measurement will be high, indicating a "resistant" emotional flow, possibly due to complex or ambiguous emotional expressions.

External Force($\vec{g}$): This function treats the polarity value as a direct representation of external influences on the text's emotional tone. The polarity could encompass various external factors like cultural context or influences from preceding text segments that affect the overall sentiment direction. For example, in this context, in a text where the overall polarity is positive, the external contextual force is represented by this positive value, suggesting that external influences (e.g., narrative context, cultural nuances) are pushing the emotional tone in a positive direction.

It is important to stress that the use of fluid dynamics concepts—such as velocity, pressure, and external forces—serves as a metaphorical framework to understand the dynamics of emotions in text. It's crucial to recognize that these variables can take on diverse meanings, extending beyond their initial definitions. For instance, the concept of "external forces" ($\vec{g}$) is not limited to the general sentiment of a sentence. It could encompass, if appropriately measured and measurable, a variety of elements outside the immediate context of the text, such as cultural nuances, historical references, or even the prevailing social or political climate in which the text was written or is being read.

Furthermore, these variables could be dynamic, changing with the reader's perspective or the broader societal context. For example, a novel's emotional "velocity" or "pressure" could be interpreted differently by readers from varying cultural backgrounds, or it might shift over time as societal norms and values evolve.

This flexibility in defining and interpreting these variables opens up a vast array of possibilities for deep and contextually rich sentiment analysis, allowing for a more comprehensive understanding of emotional dynamics in text that goes beyond mere word-level sentiment scoring.

3. Methodology and Results

To test the validity of such a novel approach, we analyzed a dataset based on three well-known Shakespeare's tragedies, such as "The Tragedy of Hamlet" and "The Tragedy of Juliet and Romeo". Our methodology incorporated a multidimensional approach, leveraging both fluid dynamics-inspired modeling and the advanced capabilities of the SenticNet lexicon (Cambria et al., 2022). Moreover, in the construction of the analytical framework for this paper, the study leveraged the comprehensive literary resources available at SparkNotes[1]. Such a platform provided essential interpretive guidance and served as a reference point for the contextualization of our emotional analysis.

3.1. SenticNet: A Tool for Sentiment Analysis and Opinion Mining

SenticNet is an advanced semantic and affective resource for sentiment analysis and opinion mining that leverages both artificial intelligence and semantic web techniques to better understand the nuances of natural language. Developed by Cambria et al. (Cambria et al., 2022), SenticNet combines common-sense reasoning tools and deep learning models to extract the polarity of texts, providing a more nuanced interpretation of emotions, sentiments, and opinions expressed in language.

SenticNet relies on The Hourglass of Emotions (Susanto et al., 2020) model which is an affective computing model that conceptualizes a multidimensional representation of human emotions. This model captures the complexity and interconnectivity of emotional states. At its core, the model posits that emotions can be classified along four primary axes or dimensions, each representing a different aspect of human affective experience (see Figure 1).

These dimensions are:

- *Introspection*: This axis gauges the positive or negative valence of emotions, representing a spectrum from ecstasy to grief.

[1] https://www.sparknotes.com/about/

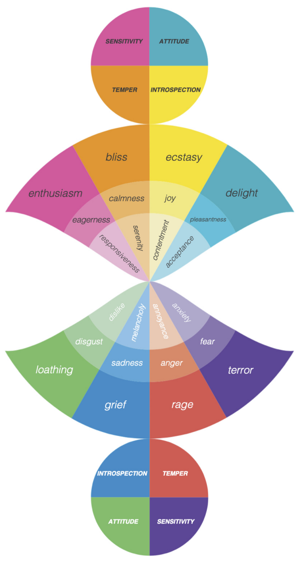


Figure 1: SenticNet Hourglass (taken from (Susanto et al., 2020))

- *Temper*: This dimension measures the level of engagement or disinterest, ranging from bliss to rage.
- *Sensitivity*: This axis reflects the degree of control or susceptibility in emotional responses, spanning from enthusiasm to terror.
- *Attitude*: This dimension accounts for the cognitive and motivational aspects of emotions, extending from empowerment to loathing.

SenticNet's utilization in our study is precisely motivated by its capability to transcend simple word-level analysis, enabling the capture of semantic and affective information associated with natural language concepts rather than merely focusing on keywords or isolated phrases. This feature renders SenticNet particularly suitable for time series analysis in research contexts that necessitate a deep interpretation of text context and emotional tone.

3.2. Data processing

We collected dialogues from texts[2] and preprocessed them using BookNLP (Bamman et al., 2014). Successively, using the SenticNet framework, each word in the dialogues was tagged for emotional categorizations and polarity scores. We initiated the analysis by identifying the most significant emotional dimension for each entity in the dataset as discussed in Section 3.1. This was achieved by computing the average values across all dimensions for each entity, and subsequently pinpointing the dimension with the highest average, as represented mathematically:

$$\text{max\_avg\_dim} = \arg\max(\text{avg\_values})$$

A comparative analysis was conducted between the results from our fluid dynamics model and traditional sentiment analysis methods consisting of a time series plotting of the computing of the average values across all SenticNet macro dimensions (Sensitivity, Attitude, Temper, Introspection) (Susanto et al., 2020) for each character of the novel using pure SenticNet scores to calculate the dimension with the highest average, as we did for the fluid dynamics model.

The first approach, inspired by fluid dynamics, was particularly effective in uncovering the ATTITUDE dimension, frequently emerging as a dominant factor in the simulations. This dimension served as a lens through which we could observe the continuum of emotional shifts pivotal to the unfolding drama in Shakespeare's works.

The second approach, relying exclusively on the SenticNet framework, is a critical component of our analysis. By utilizing SenticNet scores, we tagged the text to extract and analyze the time-series representation of the emotional lexicon, with a specific focus on the INTROSPECTION dimension since it emerged as the highest average value across all dimensions for each character of the novel.

This dimension was instrumental in highlighting the reflective and internal aspects of the characters' emotional experiences. It allowed us to delve into the frequency and intensity of introspective language within the texts, thereby revealing the prominence of self-reflection and the depth of internal emotional states characteristic of Shakespearean protagonists.

The combination of these approaches – the fluid dynamics-inspired modeling and the application of SenticNet – enabled a comprehensive and multifaceted analysis. While the fluid dynamics model illuminated the broader emotional trajectory across the narrative, SenticNet provided a detailed insight into the subtler aspects of emotional expression, particularly introspection. This dual approach not only validated the feasibility of our unique methodology but also enriched our understanding of the complex emotional tapestry woven into Shakespeare's tragedies.

3.3. Discussion of Results

In our sentiment analysis, we focus on ATTITUDE and INTROSPECTION because they often yield the highest scores in the texts we examine as provided by the *argmax* formula discussed in Section 3.2. This indicates that these dimensions are most representative of the emotional content and dynamics

[2]we downloaded the Sheakspeare's work from the digital library http://gutenberg.org

present in the text. By highlighting these dimensions, we can provide a more accurate and in-depth analysis of the emotional arcs and character development, which is particularly valuable in the study of literature, where characters' inner worlds and attitudes are often central to the narrative.

ATTITUDE, as defined in psychological terms and shown in Figure 1, refers to how a character in Shakespeare's plays evaluates and reacts to different elements like events, other characters, or their internal conflicts. It's a combination of their beliefs, emotional responses, and behaviors toward these elements. A character's attitude isn't fixed; it evolves based on their experiences, social context, and can change with different situations or moods (Bohner and Wanke, 2002). Within Shakespeare's texts, this translates to a character's evaluative stance towards the unfolding events, other characters, or internal conflicts.

When the fluid simulations identify ATTITUDE as the predominant dimension, it signifies that these evaluative stances—comprising cognitive, affective, and behavioral components—are the most salient features in influencing the play's narrative trajectory. These attitudes manifest through the characters' decisions and actions, reflecting the persistent yet context-dependent nature of their dispositions. The “distant reading” (Moretti, 2013) facilitated by this dimension enables us to view the broader emotional influences that drive the dramatic events, offering insights into how characters organize complex information and how their evaluations guide their behaviors.

Conversely, INTROSPECTION, as revealed through purely SenticNet time series analysis and shown in Figure 1, delves into the characters' self-reflective processes, mapping the psychological landscape where they contemplate their thoughts, feelings, and place within the larger narrative. This dimension captures the characters' internal dialogues, as they navigate through and make sense of their experiences, fulfilling expressive functions, affirming personal values, maintaining social identity, and regulating emotions.

The constant occurrence of INTROSPECTION in SenticNet's analysis emphasizes the capacity for self-reflection that characterizes Shakespeare's characters, offering a window into their inner world. This approach highlights the intimate and psychological spaces in which characters struggle with their attitudes, which are complex constructs acquired and modified through life experiences and socialization.

By applying computational techniques to measure ATTITUDE and INTROSPECTION, we shed light on the multifaceted ways in which Shakespeare's characters experience and express their emotions. ATTITUDE encapsulates evaluative stances that are both persistent and contextually adaptive, while INTROSPECTION provides a structural schema to organize and reflect upon these attitudes.

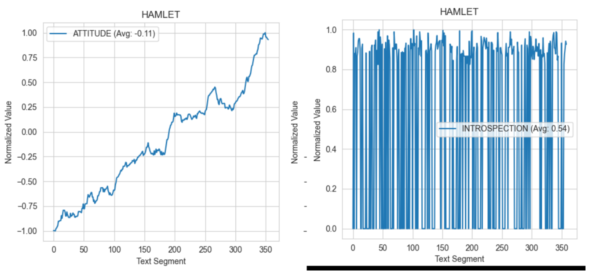


Figure 2: Emotional trajectory in 'Hamlet'. The left graph depicts the nuanced flow of emotional sentiment in 'Hamlet' through a fluid dynamics-inspired model. The right graph, utilizing data from SenticNet, quantifies the frequency of introspective moments.

This approach allows us to not only identify the presence of emotions but also to understand their evolution throughout the text. In fact, by comparing the findings from SenticNet with those from our method, we aim to demonstrate the enhancements our methodology brings to the field. While SenticNet provides a robust baseline by identifying the dominant emotional dimensions within a text, our method seeks to map these findings onto a temporal framework, thereby offering a dynamic vision of emotional trajectories.

This comparison is likely to illustrate the added value of our approach, which potentially uncovers the finer gradations and fluctuations of sentiment that may remain underrepresented in a static analysis. Consequently, our methodology could reveal an improvement over the use of SenticNet alone, offering a more granular and temporally sensitive understanding of emotional expressions in texts.

The tragedy of Hamlet, Prince of Denmark In Figure 2, we have two graphs that offer a deep dive into the emotional structure of the play.

The graph on the left shows the flow of emotion ATTITUDE across the play using a fluid dynamics-inspired approach, with the line averaging at -0.11. This suggests a generally negative undercurrent to the play's emotional tone, correlating with the narrative's progression into darker themes such as betrayal, revenge, and existential crisis.

The upward trajectory towards the end might indicate a momentary shift in emotional intensity, possibly reflecting Hamlet's resolve to avenge his father's death or his acceptance of the tragic inevitability of his circumstances. The fluid model captures the complexities of Hamlet's psychological state and the nuanced changes in the play's emotional atmosphere. It doesn't just map individual emotional incidents, as it is done by the SenticNet mapping (see right-side of Figure 2), but reflects an ongoing and accumulated emotional path, which is characteristic of the exploration of themes such as madness, indecision, and tragedy.

On the right, the INTROSPECTION graph sourced from SenticNet data illustrates the frequency of introspective sentiment within the play, with an average value of 0.54. This high average underscores the introspective nature of "Hamlet", as the title character is known for his soliloquies that explore profound philosophical questions. The spikes in the graph could represent Hamlet's soliloquies or other reflective moments within the play. While this graph captures the frequency and variability of introspective moments, it lacks the narrative context that the left graph provides, showing the moments of reflection as isolated peaks rather than as part of a continuum.

Comparing the two, the left graph offers a more holistic view of the emotional trajectory in "Hamlet". It encapsulates the build-up of tension and the psychological depth that defines the play. The right graph, although informative about the frequency of introspective moments, does not convey the emotional journey or the cumulative effect of the narrative. The left graph's approach to emotional flow allows us to visualize the overarching emotional descent that is central to the tragedy of Hamlet, which the right graph's momentary peaks cannot fully express.

In essence, the left graph's representation seems to offer a more integrated and comprehensive understanding of the emotional fabric of "Hamlet", aligning with Shakespeare's intent to create a complex portrait of a character caught in a web of existential quandaries and moral dilemmas. It captures not just the emotional states themselves, but how these states flow and interact throughout the narrative, providing a dynamic view of the emotional landscape that is as turbulent and layered as the play itself.

The tragedy of Juliet and Romeo In Figure 3, we have a vertical arrangement of graphs representing the emotional progression of the characters Romeo and Juliet.

For Romeo, the graph on the left presents the emotional sentiment throughout the play, showing a significant fluctuation that peaks and dips, with an overall average attitude of 0.21. This represents a slightly positive emotional baseline for Romeo. The initial upward trend could mirror Romeo's escalating joy as he falls in love with Juliet. This is followed by a series of sharp downturns, likely reflecting the tumultuous events he faces, such as the banishment after Tybalt's death and the tragic

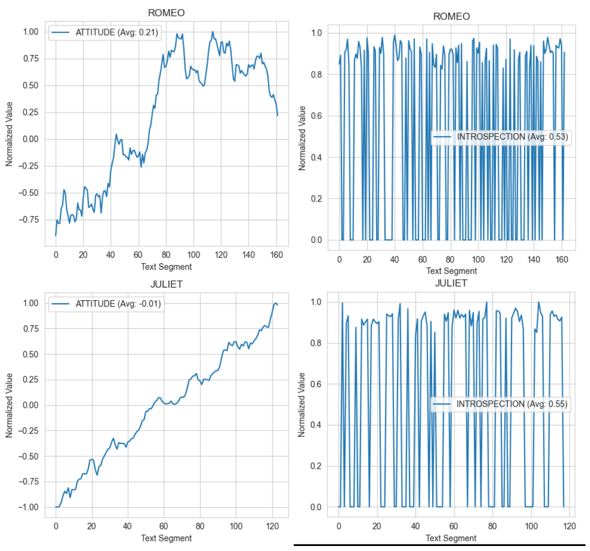


Figure 3: Emotional trajectory in 'Romeo and Juliet'. The left graph depicts the nuanced flow of emotional sentiment in 'Romeo and Juliet' through a fluid dynamics-inspired model. The right graph, utilizing data from SenticNet, quantifies the frequency of introspective moments.

news of Juliet's death.

The corresponding graph on the right, derived from SenticNet data, shows a high frequency of introspective moments, with an average of 0.53. The numerous peaks suggest that Romeo's character is marked by frequent introspective reflection, which could correlate with the many soliloquies and emotional dialogues where he contemplates his love for Juliet and his misfortunes.

The contrast between the fluid-dynamic representation of Romeo's emotional pat and the SenticNet graph is quite stark. The left graph suggests a more complex and less binary emotional landscape, offering insight into the nuanced progression of Romeo's feelings as the narrative unfolds.

Turning to Juliet, the left graph depicts her emotional sentiment throughout the play with less volatility than Romeo's, starting at a lower point but gradually increasing over time, despite a few dips. The average attitude is slightly negative at -0.01, which could reflect the constant pressures and challenges Juliet faces, including her family's expectations and the conflict with the Montagues.

The right graph shows Juliet's introspection levels, similar to Romeo's, with high frequency and intensity, averaging at 0.55. This underscores Juliet's reflective nature and the depth of her inner life as she grapples with her feelings for Romeo, and the dire circumstances that unfold.

In the case of both Romeo and Juliet, the left graphs provide a narrative of emotional development that is more aligned with the structure and thematic elements of the play. While the SenticNet graphs capture the presence of introspective and emotional language, the fluid dynamics-inspired graphs encapsulate the broader emotional arcs of the characters, effectively mapping the Shakespearean tragedy's dramatic and emotional rhythm.

3.4. Qualitative vs. Quantitative evaluation

The novel approach proposed in this paper involves a qualitative rather than quantitative evaluation of sentiment dynamics in textual narratives. This choice is driven by several compelling reasons, grounded in both the nature of our methodology and the characteristics of the subject texts.

Firstly, our approach conceptualizes emotions in the text as fluid-like entities. This fluid dynamics-inspired model inherently demands a qualitative assessment, as it aims to capture the subtle, nuanced shifts in emotional tides over time. A quantitative analysis, while valuable in its right, might not fully encapsulate these intricate dynamics.

Secondly, the richness and complexity of Shakespeare's tragedies warrant a qualitative approach. These narratives are characterized by layered emotional landscapes, where emotions are influenced by a multitude of factors, including character development, plot progression, and thematic elements. A purely quantitative evaluation might overlook these crucial aspects, thereby diminishing the depth and accuracy of the analysis.

4. Conclusions

This paper presented a novel approach to sentiment analysis in textual narratives, specifically through the application of modified Navier-Stokes equations, a fundamental concept in fluid dynamics. By conceptualizing sentiments and emotions in the text as fluid-like entities, this study aimed to capture the dynamic, evolving nature of emotional expression and propagation in narrative texts. The methodology involved mapping linguistic properties to elements of fluid dynamics, such as velocity, density, and pressure, and modifying the Navier-Stokes equations to align with these properties.

Our experimentation focused on analyzing the sentiment flow in three of Shakespeare's tragedies: "Hamlet" and "Romeo and Juliet". The results demonstrated that this fluid dynamics-inspired approach provides a more nuanced understanding of the emotional trajectory within these texts, as compared to traditional sentiment analysis methods. The application of these modified equations allowed for the visualization of sentiment dynamics, illustrating how emotions ebb and flow throughout the narrative.

The comparative analysis between the fluid dynamics-based model and the SenticNet framework revealed distinct insights. While the SenticNet approach provided valuable data on the frequency of introspective moments, the fluid dynamics model offered a broader perspective on the emotional journey throughout the plays. This comprehensive view highlighted not only individual emotional incidents but also the accumulative emotional progression characteristic of Shakespearean drama.

The findings suggest that this interdisciplinary approach holds significant promise for advancing emotional analysis in computational linguistics. By integrating principles from fluid dynamics, we can depict the complex emotional landscape of narrative texts more holistically and dynamically. This methodology has the potential to enhance our understanding of sentiment flow in a variety of textual forms, from classical literature to contemporary digital narratives.

Future work could extend this methodology to different genres and types of texts, further exploring the applicability and scalability of this approach. The potential integration of this model with advanced machine learning techniques also presents a promising direction for expanding the capabilities and applications of sentiment analysis in NLP.

D. J. Acheson. 1990. *Elementary Fluid Dynamics*. Oxford University Press.

David Bamman, Ted Underwood, and Noah A. Smith. 2014. A Bayesian mixed effects model of literary character. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 370–379, Baltimore, Maryland. Association for Computational Linguistics.

G. Bohner and M. Wanke. 2002. *Attitudes and Attitude Change*, 1st edition. Psychology Press.

Erik Cambria, Qian Liu, Sergio Decherchi, Frank Xing, and Kenneth Kwok. 2022. SenticNet 7: A commonsense-based neurosymbolic AI framework for explainable sentiment analysis. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 3829–3839, Marseille, France. European Language Resources Association.

Zulfadzli Drus and Haliyana Khalid. 2019. Sentiment analysis in social media and its application: Systematic literature review. *Procedia Computer Science*, 161:707–714. The Fifth Information Systems International Conference, 23-24 July 2019, Surabaya, Indonesia.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5:1–167.

Qiang Lu, Xia Sun, Yunfei Long, Zhizezhang Gao, Jun Feng, and Tao Sun. 2023. Sentiment analysis: Comprehensive reviews, recent advances, and open challenges. *IEEE Transactions on Neural Networks and Learning Systems*.

Franco Moretti. 2013. *Distant reading*. Verso Books.

Gunjan Pahlani, Thomas Schwartzentruber, and Richard D. James. 2023. A constitutive relation generalizing the navier–stokes theory to high-rate regimes. *Journal of Fluid Mechanics*, 974:A30.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 79–86. Association for Computational Linguistics.

John Pavlopoulos, Alexandros Xenos, and Davide Picca. 2022. Sentiment analysis of homeric text: The 1st book of iliad. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, Marseille, France. European Language Resources Association.

Patrícia Pereira, Helena Moniz, and Joao Paulo Carvalho. 2022. Deep emotion recognition in textual conversations: A survey. *arXiv preprint arXiv:2211.09172v1*.

Davide Picca and John Pavlopoulos. 2024. Deciphering emotional landscapes in the iliad: A novel french-annotated dataset for emotion recognition. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation*, Turin, Italy. European Language Resources Association.

Davide Picca and Caroline Richard. 2023. Unveiling emotional landscapes in plautus and terentius comedies: A computational approach for qualitative analysis. In *Proceedings of the Ancient Language Processing Workshop*, Varna, Bulgaria. INCOMA Ltd.

Margarita Rodríguez-Ibánez, Antonio Casánez-Ventura, Félix Castejón-Mateos, and Pedro-Manuel Cuenca-Jiménez. 2023. A review on sentiment analysis from social media platforms. *Expert Systems with Applications*, 223:119862.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Yosephine Susanto, Andrew G. Livingstone, Bee Chin Ng, and Erik Cambria. 2020. The Hourglass Model Revisited. *IEEE Intelligent Systems*, 35(5):96–102. 105 citations (Crossref) [2023-10-18].

Henk Tennekes and John L. Lumley. 1972. *A First Course in Turbulence*. MIT Press.

Math Problem Solving: Enhancing Large Language Models with Semantically Rich Symbolic Variables

Ali Emre Narin
Kabatas Erkek High School
Istanbul, Turkey
aliemre2024@gmail.com

Abstract

The advent of Large Language Models (LLMs) based on the Transformer architecture has led to remarkable advancements in various domains, including reasoning tasks. However, accurately assessing the performance of Large Language Models, particularly in the reasoning domain, remains a challenge. In this paper, we propose the Semantically Rich Variable Substitution Method (SemRiVas) as an enhancement to existing symbolic methodologies for evaluating LLMs on Mathematical Word Problems (MWPs). Unlike previous approaches that utilize generic symbols for variable substitution, SemRiVas employs descriptive variable names, aiming to improve the problem-solving abilities of LLMs. Our method aims to eliminate the need for LLMs to possess programming proficiency and perform arithmetic operations, to be universally applicable. Our experimental results demonstrate the superior accuracy of SemRiVas compared to prior symbolic methods, particularly in resolving longer and more complex MWP questions. However, LLMs' performance with SemRiVas and symbolic methods that utilize one-character variables still falls short compared to notable techniques like CoT and PaL.

Keywords: Math Word Problems, Large Language Models

1. Introduction

The Transformer architecture (Vaswani et al., 2023) has facilitated the development of Large Language Models (LLMs) capable of achieving exceptional performance. Models like PaLM 540B (Chowdhery et al., 2023) and GPT-4 (Achiam et al., 2023), with billions of parameters, have undergone training on vast amounts of textual data using the Transformer architecture. These models exhibit outstanding capabilities across various domains, including reasoning, coding, common sense, translation, and planning, often reaching or surpassing human-level performance (Achiam et al., 2023). The remarkable performance of these models has sparked significant interest among researchers, leading to the creation of numerous LLMs such as Llama (Touvron et al., 2023) and Phi (Li et al., 2023).

The proliferation of models has motivated researchers to devise various methodologies for assessing model performance and to curate datasets for evaluating these methodologies. Typically, researchers evaluate the responses of LLMs against questions in datasets, assessing the accuracy of these responses compared to ground truth answers (Mialon et al., 2023; Cobbe et al., 2021).

One domain where such evaluations are conducted is reasoning. In this domain, models are tasked with detecting and executing various operations based on provided text. Achieving satisfactory results in the reasoning domain has proven challenging for LLMs, with many models exhibiting notably low accuracy rates in this area (Cobbe et al., 2021).

To measure accuracy in the reasoning domain, many researchers use Mathematical Word Problems (MWP) (Li et al., 2023; Chowdhery et al., 2023). These MWPs present mathematical challenges using everyday language and numerical data. Successfully solving the problems demands models to possess a strong proficiency in Natural Language Understanding (NLU), as they must identify both the problem scenario and the route to the solution beforehand. This dual requirement of comprehending the context and deducing the problem's intent closely aligns with human reasoning capabilities, making MWPs a suitable evaluation method in the reasoning domain.

Various datasets and methods are employed by researchers to gauge skills using MWPs. Examples of widely used datasets in the literature include GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), and SVAMP (Patel et al., 2021).

In addition to datasets, various methods have been developed to assist Large Language Models (LLMs) in accurately answering questions, particularly for Mathematical Word Problems (MWPs). Predominant among these are Chain-of-Thought Prompting (CoT) (Wei et al., 2023) and Program Aided Language Models (PaL) (Gao et al., 2023). CoT employs an intuitive methodology, guiding AI models through progressive problem-solving steps, while PaL necessitates models to generate program code for solution computation.

These two methods have been observed to significantly increase the accuracy rates of responses to questions in datasets used for measuring reasoning in LLMs. However, both approaches have spe-

Question
Dan plants 3 rose bushes. Each rose bush has 25 roses. Each rose has 8 thorns. How many thorns are there total?
Answer
Dan plants 3 rose bushes. Each rose bush has 25 roses. Each rose has 8 thorns. So **3 x 25 x 8 = 300**. The answer is 300.

Figure 1: Examples of Errors Made by Large Language Models in Arithmetic (Wei et al., 2023)

cific problems. In the CoT approach, LLMs are expected to perform arithmetic operations. However, LLMs are not models highly-competent in performing arithmetic operations (Nogueira et al., 2021; Lu et al., 2023; Frieder et al., 2023; Stolfo et al., 2023; Meadows et al., 2023). The primary function of LLMs in solving MWPs should be to determine which operations to perform among numbers based on the given scenario. As shown in Figure 1, models correctly define operations, but inaccuracies in performing operations have been observed, leading to decreased accuracy rates. Therefore, measurements made with these approaches only partially reflect reality. The PaL method is not effected by errors that can occur due to arithmetic operations, since a third party code interpreter runs the calculations, however it is limited for models that are proficient in using programming languages. Therefore, PaL is limited in terms of the LLMs it can be applied to.

New methods have emerged to address these challenges, leveraging symbolic variables to delegate arithmetic computations to an external calculator (He-Yueya et al., 2023). Additionally, these methodologies does not require any programming proficiency. One particular method, replaced variables in a question with “w,x,y,z” variables, and then asked LLM to solve the question using self-prompting strategies (Gaur and Saunshi, 2023). We refer to this kinds of approaches in our paper as "One-Character Substitution Method", as they exchange numbers with one-character variables.

This strategy effectively tackles issues related to arithmetic errors and programming expertise limitations. In our study, we investigate the potential refinement of this technique by replacing these generic placeholders with semantically rich counterparts. Instead of employing generic symbols, we advocate for descriptive variable names such as "number-of-books-James-has" as exemplified in Figure 2. We hypothesize that this modification will aid LLMs in problem-solving tasks, as managing numerous one-character variables, especially in contexts involving multiple numerical values, poses a significant challenge even for humans.

We introduce the Semantically Rich Variable Substitution Method (SemRiVas) as an improvement of prior work that leverages symbolic variables to aid LLMs in solving MWPs. Our research contributes to the literature in the following ways:

1. Our approach eliminates the need for LLMs to possess programming proficiency and perform arithmetic operations, making it universally applicable as an evaluation method.
2. We commit to releasing all code and datasets associated with our method, facilitating its adoption for evaluating LLM performance on MWPs.
3. Our method demonstrates superior accuracy compared to previously employed One-Character Substitution Methods.

2. Background

2.1. Large Language Models

Large Language Models(LLMs) are advanced natural language processing systems that utilize different machine learning techniques to comprehend and generate text in human language. With the development of Transformer architecture (Vaswani et al., 2023), massive models with billions of parameters have been proposed. Some of the notable examples include ChatGPT (OpenAI, 2022) , Phi1.5 (Li et al., 2023), and Llama2-7B (Touvron et al., 2023). These models demonstrated remarkable performances in several natural language processing tasks, including machine translation, common sense reasoning, summarizing, planning, reasoning, and coding.

2.2. Math Word Problems

Math Word Problems (MWPs) are mathematical problem-solving scenarios within a contextualized linguistic framework. Solving these problems necessitates the translation of the verbal description into mathematical expressions (Meadows and Freitas, 2023), such as equations or inequalities, which requires analytical thinking and reasoning skills. Recent works have prevalently utilized math word problems to evaluate LLM's reasoning capabilities (Li et al., 2023; Chowdhery et al., 2023).

2.3. Chain of Thought Prompting

Chain of Thought Prompting (CoT) (Wei et al., 2023) is a strategy that aims to decompose a problem into several intermediary operations to reduce the problem's complexity, resulting in better performances. Specifically, when employing CoT, we prompt the

SemRiVas

Question
James has number-of-books-James-has books. If he buys number-of-books-James-buys books, how many books he has?
Answer
James will have number-of-books-James-has + number-of-books-James-buys books.

One-Character Substitution

Question
James has w books. If he buys 2x books, how many books he has?
Answer
James will have w + 2x books.

Figure 2: Comparison of Methods

LLM with instructions such as "Let's think step by step," directing the LLM to follow a divide-and-conquer approach. Researchers have shown that this intuitive and easy-to-implement approach exhibits remarkable accuracies when tested with math word problems.

2.4. Program-Aided Language Models

When solving math word problems, LLMs have been shown to encounter difficulties in performing arithmetic. With Program-Aided Language Models (PaL)(Gao et al., 2023), researchers aimed to utilize LLMs' coding abilities to address math word problems indirectly. Instead of directly providing the answer, the approach involves making the model generate a Python code that would, in turn, produce the correct result. In this method, once the Python code is generated, it is given to a third-party Python interpreter to execute the operations. By employing this strategy, they aimed to minimize the errors stemming from LLMs' inability to perform simple math. They demonstrated that PaL achieved high accuracies in their tests with LLMs capable of coding.

2.5. Solving Math Word Problem's Symbolically

Symbolic methods (Ferreira et al., 2022) represent quantities and relationships using symbols or variables, allowing for systematic manipulation and solution. Symbolic methods make sense because they reduce the likelihood of arithmetic errors and enable the application of mathematical reasoning and algorithms to complex scenarios.

The most notable work concerning our domain in this research has used (w, x, y, z) (Gaur and Saunshi, 2023) variables to substitute numerical values in the SVAMP (Patel et al., 2021) Dataset, then tested LLM's ability on this new augmented dataset. We will cover more about that method in the Baselines subsection.

3. Methods

In this study, we introduce SemRiVas: Semantically Rich Variable Substitution method. The SemRiVas method primarily aims to replace numbers in Math Word Problems (MWPs) with self-explanatory variables. This substitution facilitates differentiation between variables, particularly in lengthy questions, and delegates calculations to a third-party calculator to reduce arithmetic errors. We utilize the few-shot prompting technique for consistent and accurate responses.

3.1. Semantically Rich Symbolic Variables

Using symbolic variables instead of numerical values can enhance model performance in solving MWPs. Models often commit arithmetic errors while maintaining correct reasoning, as illustrated in Figure 1. Employing a simple third-party calculator can rectify these errors efficiently. Hence, substituting variables in questions and later solving them appears logical to improve accuracy. Previous methods have utilized one-character variables such as [w, x, y, z], [p, q, r, s], however, we posit that one-character variables might confuse models, especially when there are many of them (Gaur and Saunshi, 2023).

Instead of replacing numbers with one-character variables, we advocate substituting variables that describe the number's purpose. For instance, as depicted in Figure 2, we replace numbers with variables like "number-of-books-James-has." We hypothesize that in a question with multiple numbers, it would be challenging to track one-character variables and recall each variable's significance. Conversely, using self-explanatory variables could aid model comprehension.

3.2. Few-Shot Prompting

Few-Shot Prompting is a strategy where question-answer pairs are compiled in the desired format of model responses. Subsequently, the model is prompted with these question-answer examples, followed by the original query. This approach guides the model to adhere to specified structural properties and provides concrete examples. It is particularly valuable for evaluation, facilitating the generation of extractable and quantifiable examples.

We have opted for 8-shot prompting in all our evaluations. These prompts were drawn from GSM-8K (Cobbe et al., 2021) and customized for each method addressed in this study. All prompts will be made available as open-source resources to facilitate the adoption of our method.

4. Results and Discussion

4.1. Baselines

In this study, we opted for the GSM8K benchmark (Cobbe et al., 2021) for evaluations, given its widespread use in the literature. The GSM8K dataset consists of high/middle-school level Mathematical Word Problems (MWP) of high quality. Another reason for selecting GSM8K was to examine the impacts of variable-substitution methods on both short and long-context questions. Previous studies utilized the SVAMP (Patel et al., 2021) dataset, which is constrained to a maximum of 4 numbers, whereas GSM-8K encompasses numbers ranging up to 8.

We chose GPT-3.5, specifically "gpt-3.5-turbo," as our base LLM due to its strong coding skills, enabling accurate PaL (Gao et al., 2023) calculation and prompt-following abilities for smoother processes.

To evaluate our method against previous approaches, we assessed PaL, CoT, and a One-Character Substitution Method similar to prior work that utilized [w,x,y,z].

4.2. Evaluation Strategy

We evaluated all four methods on 200 randomly selected QA pairs from GSM-8K. (Due to financial constraints, only 200 were selected). The process of number replacement to generate the dataset for the SemRiVas method was initially assisted by an LLM and later supervised by humans to address any errors. We used a simple Python program to generate the dataset for the One-Character Substitution method, as it didn't require specific variable names for each number.

The models were prompted with similar prompts differing only in response format, not in questions.

For the PaL method, we offloaded the generated responses into a python interpreter for evaluation. For the One-Character Substitution, and SemRiVas method, we simply exchanged variables with the original numbers, and evaluated from there using a simple python calculator. CoT didn't require any additional steps to evaluate, we simply took the final answer it wrote.

We maintained consistent hyperparameters for all methods: temperature parameter at 0.7 and Top-P parameter at 1. We observed in a smaller subset of data that these parameter values have proven to be highly robust compared to higher or lower settings.

Table 1: Accuracy Rates of Each Method (8-Shot)

PaL	CoT	SemRiVas	One-Character S.
48%	84%	44%	34.5%

4.3. Results

According to our research results (Table 1), the GPT-3.5 model prompted with the SemRiVas method achieved a 44% success rate across 200 questions. PaL achieved 48% accuracy, CoT achieved 84%, and the One-Character Variable Substitution method achieved 34.5%.

We have also analyzed the success rates of SemRiVas and the One-Character variable method on the 50 longest/shortest questions to see whether our hypothesis that using semantically rich variables enhances performance in long-context settings. We found that SemRiVas was 38% more successful than the One-Character Variable Substitution method on the longest questions, while it was marginally less successful (7%) than the one-character variable substitution method on the shortest questions.

4.4. Discussion

Our method, designed to measure LLMs' logical and reasoning abilities, achieved higher accuracy than the one-character variable substitution method, demonstrating the effectiveness of semantically rich variables. This increase in accuracy underscores the effectiveness of our approach for solving MWPs symbolically using large language models.

However, we can see that LLMs seem to be struggling to solve MWPs using symbolic variables, as both symbolic methods didn't surpass the accuracy rate of CoT or PaL.

That said, due to budget and manpower limitations, our experiments were confined to the GPT-3.5 model and a smaller subset of the GSM-8K (Cobbe et al., 2021) benchmark. Further testing on the entire GSM-8K dataset and with different models could provide a clearer picture of our approach's success rate.

5. Conclusion

In conclusion, our research introduces the Semantically Rich Variable Substitution Method (SemRiVas) as a novel approach to evaluate Large Language Models (LLMs) on Mathematical Word Problems (MWPs). By utilizing descriptive variable names instead of generic symbols, SemRiVas aims to improve LLMs' problem-solving abilities, particularly in long-context settings. Our experimental results demonstrate the superior accuracy of SemRiVas compared to previous symbolic methods, highlighting its effectiveness in enhancing LLM reasoning capabilities. However, our findings also suggest that LLMs still face challenges when solving MWPs using symbolic variables, as they did not surpass the accuracy rates achieved by methods such as CoT or PaL. Further research is needed to understand and address these challenges comprehensively.

6. Acknowledgements

The author would like to thank Clayton Greenberg for his valuable mentorship.

7. Bibliographical References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mo Bavarian, Jeff Belgum, Irwan Bello, and ... Barret Zoph. 2023. Gpt-4 technical report.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, and ... Noah Fiedel. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems.

Deborah Ferreira, Mokanarangan Thayaparan, Marco Valentino, Julia Rozanova, and Andre Freitas. 2022. To be or not to be an integer? encoding variables for mathematical text. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 938–948, Dublin, Ireland. Association for Computational Linguistics.

Simon Frieder, Luca Pinchetti, , Ryan-Rhys Griffiths, Tommaso Salvatori, Thomas Lukasiewicz, Philipp Petersen, and Julius Berner. 2023. Mathematical capabilities of chatgpt. In *Advances in Neural Information Processing Systems*, volume 36, pages 27699–27744. Curran Associates, Inc.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Vedant Gaur and Nikunj Saunshi. 2023. Reasoning in large language models through symbolic math word problems.

Joy He-Yueya, Gabriel Poesia, Rose E. Wang, and Noah D. Goodman. 2023. Solving math word problems by combining language models with symbolic solvers.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1. Curran.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. 2023. Textbooks are all you need ii: phi-1.5 technical report.

Pan Lu, Liang Qiu, Wenhao Yu, Sean Welleck, and Kai-Wei Chang. 2023. A survey of deep learning for mathematical reasoning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14605–14631, Toronto, Canada. Association for Computational Linguistics.

Jordan Meadows and André Freitas. 2023. Introduction to Mathematical Language Processing: Informal Proofs, Word Problems, and Supporting Tasks. *Transactions of the Association for Computational Linguistics*, 11:1162–1184.

Jordan Meadows, Marco Valentino, Damien Teney, and Andre Freitas. 2023. A symbolic framework for systematic evaluation of mathematical reasoning with transformers.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2023. Gaia: a benchmark for general ai assistants.

Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. 2021. Investigating the limitations of transformers with simple arithmetic tasks.

OpenAI. 2022. Introducing chatgpt.

Tim Orchard and Leszek Tasiemski. 2023. The rise of generative ai and possible effects on the economy. *Economics and Business Review*, 9:9.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online. Association for Computational Linguistics.

Alessandro Stolfo, Zhijing Jin, Kumar Shridhar, Bernhard Schölkopf, and Mrinmaya Sachan. 2023. A causal framework to quantify the robustness of mathematical reasoning with language models. pages 545–561.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and ... Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models.

Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. 2023. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment.

Data Driven Approach for Mathematical Problem Solving

Byungju Kim, Wonseok Lee, Jaehong Kim and Jungbin Im
Mathpresso Inc.
Seoul, Republic of Korea
{peyton.kim, jack.lee, julio.kim, marvin.im}@mathpresso.com

Abstract

In this paper, we investigate and introduce a novel Llama-2 based model, fine-tuned with an original dataset designed to mirror real-world mathematical challenges. The dataset was collected through a question-answering platform, incorporating solutions generated by both rule-based solver and question answering, to cover a broad spectrum of mathematical concepts and problem-solving techniques. Experimental results demonstrate significant performance improvements when the models are fine-tuned with our dataset. The results suggest that the integration of contextually rich and diverse problem sets into the training substantially enhances the problem-solving capability of language models across various mathematical domains. This study showcases the critical role of curated educational content in advancing AI research.

Keywords: Mathematical problem solving, Data-driven, Large language model

1. Introduction

In the field of machine learning, the ability to solve complex mathematical problems is often used as a measure of a model's reasoning abilities, understanding of natural language, and its capacity to engage in abstract thinking. From the basic arithmetic operations to more complex numerical challenges, the capacity of machine learning models to navigate and resolve mathematical problems lays the groundwork for advanced applications in fields such as data analysis, and education. This significance is due to the fundamental nature of mathematics as a form of structured problem-solving. The endeavor to enhance machine learning models' capability in mathematical problem-solving is driven by the dual goals of improving their analytical capabilities and enabling them to handle real-world tasks that require precise numerical computations. This pursuit involves not only refining the models' ability to understand and analyze numerical data but also their capability to interpret contextual information and properly apply mathematical concepts in varied scenarios.

On the other hand, the advent of large language models (LLMs) has marked a significant milestone in demonstrating the potential of data-driven approaches. Numerous LLMs, such as GPT (OpenAI et al., 2024), Gemini (Team et al., 2023), Llama-2 (Touvron et al., 2023; Rozière et al., 2024) and Orca-2 (Mitra et al., 2023), have demonstrated an ability to understand and generate human-like text. They have achieved exceptional performance across a variety of tasks, including mathematical problem-solving. This capability arises from their extensive training on diverse datasets, and sophisticated training algorithms to process and learn from the data. The remarkable performance of these models has shown that with sufficiently many data, it is possible to achieve levels of understanding and interpreting capabilities that closely mimic human cognitive processes.

Moreover, the scalability of large language models shows that their performance often improves with the addition of more data. This phenomenon is referred to as "scaling laws" (Johnson et al., 2018; Kaplan et al., 2020; Fernandes et al., 2023; Isik et al., 2024). This suggests that the limits of these models' capabilities are continually expanding, as more data becomes available for training.

Similar approaches have been attempted in the field of mathematical problem-solving. Llemma-2 (Azerbayev et al., 2023), part of the Llama-2 (Touvron et al., 2023) series, have been trained on a mixture of publicly available data, and achieved remarkable performances in various mathematical tasks (Hendrycks et al., 2021b; Cobbe et al., 2021a; Lewkowycz et al., 2022; Hendrycks et al., 2021a). Their performance enhancement implies that the current transformer-based (Vaswani et al., 2023) LLMs can learn mathematical induction from the large corpus of data.

Another research stream of data-driven training is to utilize several existing LLMs to synthesize new datasets (Yu et al., 2023; Toshniwal et al., 2024; Yue et al., 2023) or to teach other models (Burns et al., 2023; Luo et al., 2023). Their primary aim is to curate a variety of math problems with rich step-by-step solutions, enabling a LLM to effectively learn the logic underlying the progression of mathematical steps.

In this work, we introduce a new model based on Llama-2, trained using our dataset. Drawing inspiration from previous research, we have collected a novel dataset for math problem-solving. Our dataset collection methodology ensures that it mirrors the distribution of mathematical challenges encountered in the real world. This model demonstrates consistent performance improvements on math problem-solving tasks. To benchmark its performance, we conducted evaluations of our trained models against the MATH and GSM8k datasets (Hendrycks et al., 2021b; Cobbe et al., 2021b). Further analysis of our dataset's composition reveals that the observed performance improvements align with its distribution. The alignment of our dataset's distribution with the performance improvements suggests that performance could be further enhanced by expanding our data. This implies that as we enrich our dataset with a broader range of problems, the model's ability to tackle diverse mathematical tasks is likely to improve. It highlights the importance of a comprehensive dataset for optimizing performance in math problem-solving tasks.

2. Dataset Construction

To construct a dataset of math problems with explanatory solutions, we have utilized our question answering platform, [1]QANDA. Our platform serves as a math question-and-answer app, designed to bridge the gap between students facing mathematical challenges and teachers equipped to provide solutions. The interactive environment allows students to pose math problems, to which the platform's network of qualified teachers responds with detailed answers, explanations, and step-by-step guidance. Since the math problems are originated from the users who pose a question to the platform, the distribution of the problems in our dataset reflects the diverse mathematical challenges encountered by students. To secure the broader diversity in solutions, we have collected two distinct types of solutions offered by the platform: question answering and rule-based math solver.

2.1. Rule-based Math Solver

Similar to the Sympy (Meurer et al., 2017), a python library for symbolic mathematics, the platform provides a solution to a mathematical problem expressed through symbolic expressions. By adopting a rule-based approach, the solver ensures a high degree of accuracy and reliability, offering solutions that mimic the methodical process a human may use. Moreover, it offers step-by-step solutions to the given problems, thereby enabling a deeper understanding of the problem-solving process. By leveraging the platform, we have obtained detailed solutions, which demonstrate the procedural steps involved in resolving complex mathematical expressions.

[1]https://mathpresso.com/en

Problem :
Solve the equation $\frac{x+3}{x+2} - \frac{x}{x-2} = \frac{5x}{x^2-4}$
Solution :
If $\frac{a(x)}{b(x)} = c(x)$ is valid, it is $\begin{cases} a(x) = b(x)c(x) \\ b(x) \neq 0 \end{cases}$

$$\begin{cases} 5x = (x^2-4)\left(\frac{x+3}{x+2} - \frac{x}{x-2}\right) \\ x^2-4 \neq 0 \end{cases}$$

Simplify the expression

$$\begin{cases} 5x = -x-6 \\ x^2-4 \neq 0 \end{cases}$$

Solve a solution to x

$$\begin{cases} x = -1 \\ x^2-4 \neq 0 \end{cases}$$

Substitute $x = -1$

$$\begin{cases} x = -1 \\ (-1)^2-4 \neq 0 \end{cases}$$

Therefore, $x = -1$

(a)

Problem :
Given two sets $X = \{-1, 0, 1\}$ and $Y = \{-2, -1, 0, 1, 2\}$, there exists a function f from X to Y. For every element x in X, when $xf(x)$ is a constant function, what is the value of $f(1) + f(-1)$?
Solution :
Given $xf(x)$ is a constant function, we have $xf(x) = c$. When $x = 0$, we have $0 \cdot f(0) = 0$. Therefore, $c = 0$. Hence, for all x, we have $xf(x) = 0$, which implies $f(x) = 0$. Therefore, $f(1) + f(-1) = 0 + 0 = 0$.

(b)

Figure 1: Data points sampled from our dataset. Given math problems, their solutions are collected from either (a) rule-based math solver and (b) question answering. Rule-base solver provides detailed step-by-step solutions, while the other covers more complex problems

The enriched datasets offer a substantial advantage in training LLMs. The detailed solutions are inherently superior for training purposes since they encompass an extensive range of information about mathematical concepts and the procedures involved in problem-solving (Lightman et al., 2023; Wang et al., 2024). Such an approach ensures the models not to estimate the mathematical reasoning but to replicate the logical deductions required to solve complex problems. Consequently, these enriched training datasets are instrumental in enhancing the capability of LLMs to solve mathematical

Training Procedure	Prealg.	Algebra	Intermediate Algebra	Number Theory	Counting & Probability	Geometry	Precalculus
				Level 1			
Llama + M	59.3	64.4	50.0	40.0	38.5	31.6	42.1
Llama + Q + M	73.3	79.3	63.5	43.3	**69.2**	55.3	61.4
Llemma + M	80.2	85.2	69.2	**63.3**	61.5	**57.9**	49.1
Llemma + Q + M	**81.4**	**87.4**	**75.0**	56.7	64.1	**57.9**	**64.9**
				Level 2			
Llama + M	48.0	49.3	17.2	21.7	25.7	30.5	15.0
Llama + Q + M	64.4	**74.6**	35.9	**39.1**	37.6	**47.6**	28.3
Llemma + M	63.3	67.7	25.0	22.8	34.7	39.0	29.2
Llemma + Q + M	**70.1**	73.1	**38.3**	37.0	**40.6**	45.1	**35.4**
				Level 3			
Llama + M	35.3	34.5	7.7	14.8	13.0	14.7	2.4
Llama + Q + M	**53.6**	55.9	14.9	**24.6**	29.0	25.5	10.2
Llemma + M	50.5	54.8	12.3	**24.6**	32.0	29.4	17.3
Llemma + Q + M	52.7	**69.0**	**21.0**	**24.6**	**36.0**	**39.2**	**21.3**
				Level 4			
Llama + M	26.7	17.3	7.3	9.2	7.2	10.4	3.5
Llama + Q + M	41.4	**49.8**	8.9	11.3	13.5	13.6	6.1
Llemma + M	38.2	34.3	7.7	**17.6**	13.5	15.2	6.1
Llemma + Q + M	**46.1**	48.1	**12.9**	**17.6**	**16.2**	**24.0**	**7.0**
				Level 5			
Llama + M	9.3	8.8	1.4	**5.8**	1.6	1.5	0.0
Llama + Q + M	17.6	21.2	2.1	5.2	5.7	1.5	1.5
Llemma + M	19.7	20.9	2.9	5.2	4.9	3.0	**4.4**
Llemma + Q + M	**21.8**	**30.6**	**3.6**	5.2	**8.1**	**3.8**	3.0
				Overall			
Llama + M	32.6	29.7	9.4	13.3	13.5	14.0	8.8
Llama + Q + M	47.1	51.3	15.1	19.1	24.5	21.9	16.3
Llemma + M	46.5	46.8	13.2	19.1	23.6	22.3	17.6
Llemma + Q + M	**50.8**	**56.9**	**18.9**	**21.1**	**27.4**	**28.0**	**21.3**

Table 1: Model performance across different mathematical domains. Models trained with our dataset show better performance in most of the mathematical domains. In the training procedure, (+Q) denotes fine-tuning with our dataset collected through QANDA, and (+M) denotes fine-tuning with Metamath dataset

problems.

Figure 1 (a) describes an exemplar data instance. In each step of the solution, the solver provides a brief explanation of the related concept before proceeding with the actual calculation.

2.2. Question Answering

Once a user pose a math problem, the platform searches from the database and curate several problem-solution pairs that match the query question. The database is constructed to aid students in understanding mathematical concepts. It makes the obtained problem and solution to be intrinsically educationally effective; the solutions are structured and detailed. These educational characteristics, such as providing step-by-step explanations and highlighting the underlying mathematical principles, benefit the training procedure of LLMs as well.

Figure 1 (b) describes an exemplar data instance collected through question answering. Comparing to data collected through rule-based math solver, data pairs gathered through question-answering mechanisms reveals a notable difference in the complexity. The problems accumulated through question answering tend to require more complex procedures to solve. It is trivial since we can easily compose a math problem with multiple expressions. This complexity demands a deeper understanding of mathematical concepts and longer deduction process. In other words, the dataset collection through question answering not only diversifies the range of problems in the dataset but also enriches it with challenges that necessitates advanced problem-solving strategies.

3. Experiments

3.1. Model Training

We fine-tuned Llama-2 7B (Touvron et al., 2023) and Llemma-2 (Azerbayev et al., 2023) 7B with our dataset. Each data instance is presented in the following prompt:

```
Problem:{math problem}
Solution:{ground truth solution}
```

To computationally evaluate performance in math problem solving, it is crucial for the model to generate an answer that is parsable. Unfortunately, as illustrated in Figure 1, achieving this is fundamentally challenging within our dataset, given that the solutions were not created in a fully controlled environment. To address this issue, we further fine-tuned our trained model using the Metamath dataset (Yu et al., 2023), enabling the model to learn the generation of well-formatted outputs. For a fair comparison, both the Llama-2 and Llemma-2 models were also trained using the Metamath dataset, utilizing the prompt described earlier.

3.2. Evaluation

To verify the effectiveness of our dataset, we evaluated the performance of MATH datasets (Hendrycks et al., 2021b). MATH is a challenging dataset designed to evaluate the mathematical problem-solving capabilities of machine learning models. It covers a wide array of domains, including prealgebrea, algebra, number theory, counting, probability, geometry, intermediate algebra, and precalculus.

Table 1 shows the performance of our approach. We break down the dataset into domains and levels to examine detailed characteristics and trends. In the training procedure, +Q denotes fine-tuning with our dataset collected through QANDA, and +M denotes fine-tuning with Metamath dataset. Note that every model undergoes fine-tuning with Metamath dataset in the end. The table distinctly demonstrates that the fine-tuning with our dataset (+Q) significantly improves the original performance, indicating a considerable improvement.

Here, it is noteworthy to focus on the difference between Llama+O and Llemma model. Llemma is initialized with CodeLlama (Rozière et al., 2024), and trained with dataset named *Proof-Pile-2* (Azerbayev et al., 2023). The dataset is composed of code (Kocetkov et al., 2022), mathematical content from web (Paster et al., 2023), and scientific papers

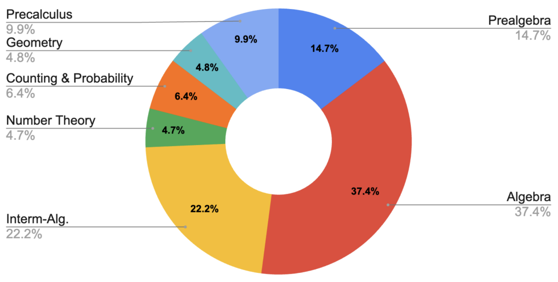


Figure 2: Domain composition of our dataset

(Computer, 2023). Our dataset, on the other hand, is fully composed of mathematical problems and their solutions.

3.3. Composition of Our Dataset

Interesting trends emerge in Algebra and Precalculus. In Algebra, the Llama+Q+M model consistently outperforms the Llemma+M model across all difficulty levels, except for the easiest set (Level 1). In contrast, within Precalculus, the Llemma+M model consistently outperforms Llama+Q+M, again with the exception of the easiest set. This suggests that the Llemma model has a stronger grasp of concepts in calculus, whereas the Llama+Q model is more adept at handling algebraic problems. Since the major difference between the two models is their training data, it is reasonable to consider that the trends implies the compositional difference between *Proof-Pile-2* and our datasets. Although not as pronounced as in the case of Precalculus, Table 1 also exhibits similar trends within the Geometry and Number Theory domains.

To explore the relationship between dataset composition and model performance, we categorized the instances in our dataset based on the domains present in the MATH dataset. Figure 2 illustrates our dataset's composition, reflecting the aforementioned trends. The majority of our dataset is classified under Algebra, Prealgebra, or Intermediate Algebra, whereas less than 10% of the samples falling into the Precalculus category. Additionally, Figure 2 shows that Geometry and Number Theory are the lesser-represented domains in our dataset. This distribution aligns with the observed performance trends.

The optimistic outlook based on this trend is that we could enhance performance in domains beyond algebra simply by collecting more data. However, this approach may potentially compromise algebra's performance.

3.4. Overall Performance

While our investigation primarily concentrated on how the composition of our dataset affects the per-

Models	GSM8k	MATH
MAmmoTH	53.6	31.5
Metamath	66.5	19.8
Llemma + M	69.2	30.0
Mistral + M	**77.7**	28.2
ToRA	68.8	**40.1**
Llama + Q + M	66.2	31.4
Llemma + Q + M	<u>71.0</u>	<u>36.1</u>

Table 2: Overall performance of various models

formance improvements within their corresponding domain, we also validated our methodology by assessing overall performance. For this experiment, we incorporated the GSM8k(Cobbe et al., 2021b) dataset. The GSM8k dataset consists of grade school math problems that require two to eight steps to solve, involving elementary-level calculations through basic arithmetic operations. For each model, we fixed their size as 7B.

Table 2 presents a comparison of the overall performance between our method and other models. It is important to note that the Metamath model is identical to Llama+M in Table 1. For both datasets, our model (Llemma+Q+M) achieves the second-highest performance. Our model notably excels in the MATH dataset over other models, with the exception of ToRA (Gou et al., 2024). Given that ToRA employs a tool-augmented, multi-step method, our findings highlight the efficacy of our data-driven approach.

4. Conclusion

This study has highlighted the potential of data-driven approaches to mimic and augment human cognitive processes in structured problem domains. By training a novel Llama-2 based model with a specially curated dataset reflective of real-world mathematical challenges, we have demonstrated significant advancements in the model's ability to tackle complex numerical tasks across a variety of mathematical domains. Our dataset, constructed from a unique blend of rule-based solutions and human-generated answers via a question-answering platform, has proven to be beneficial in achieving these improvements. The performance of our model, especially when compared against the MATH dataset, validates the efficacy of our dataset in enhancing the analytical capabilities of LLMs. The findings from this research suggest that the integration of more diverse and complex datasets would result in better performing models in mathematical domains.

References

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2023. Llemma: An open language model for mathematics.

Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, Ilya Sutskever, and Jeff Wu. 2023. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021a. Training verifiers to solve math word problems.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021b. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Together Computer. 2023. Redpajama: an open dataset for training large language models.

Patrick Fernandes, Behrooz Ghorbani, Xavier Garcia, Markus Freitag, and Orhan Firat. 2023. Scaling laws for multilingual neural machine translation.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. 2024. Tora: A tool-integrated reasoning agent for mathematical problem solving.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the math dataset.

Berivan Isik, Natalia Ponomareva, Hussein Hazimeh, Dimitris Paparas, Sergei Vassilvitskii, and Sanmi Koyejo. 2024. Scaling laws for downstream task performance of large language models.

Mark Johnson, Peter Anderson, Mark Dras, and Mark Steedman. 2018. Predicting accuracy on large datasets from smaller pilot data. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 450–455, Melbourne, Australia. Association for Computational Linguistics.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models.

Denis Kocetkov, Raymond Li, Loubna Ben Allal, Jia Li, Chenghao Mou, Carlos Muñoz Ferrandis, Yacine Jernite, Margaret Mitchell, Sean Hughes, Thomas Wolf, Dzmitry Bahdanau, Leandro von Werra, and Harm de Vries. 2022. The stack: 3 tb of permissively licensed source code.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. 2023. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct.

Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. 2017. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103.

Arindam Mitra, Luciano Del Corro, Shweti Mahajan, Andres Codas, Clarisse Simoes, Sahaj Agarwal, Xuxi Chen, Anastasia Razdaibiedina, Erik Jones, Kriti Aggarwal, Hamid Palangi, Guoqing Zheng, Corby Rosset, Hamed Khanpour, and Ahmed Awadallah. 2023. Orca 2: Teaching small language models how to reason.

OpenAI, :, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. Gpt-4 technical report.

Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, and Jimmy Ba. 2023. Openwebmath: An open dataset of high-quality mathematical web text.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2024. Code llama: Open foundation models for code.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Slav Petrov, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy Lillicrap, Angeliki Lazaridou, Orhan Firat, James Molloy, Michael Isard, Paul R. Barham, Tom Hennigan, Benjamin Lee, Fabio Viola, Malcolm Reynolds, Yuanzhong Xu, Ryan Doherty, Eli Collins, Clemens Meyer, Eliza Rutherford, Erica Moreira, Kareem Ayoub, Megha Goel, George Tucker, Enrique Piqueras, Maxim Krikun, Iain Barr, Nikolay Savinov, Ivo Danihelka, Becca Roelofs, Anaïs White, Anders Andreassen, Tamara von Glehn, Lakshman Yagati, Mehran Kazemi, Lucas Gonzalez, Misha Khalman, Jakub Sygnowski, Alexandre Frechette, Charlotte Smith, Laura Culp, Lev Proleev, Yi Luan, Xi Chen, James Lottes, Nathan Schucher, Federico Lebron, Alban Rrustemi, Natalie Clay, Phil Crone, Tomas Kocisky, Jeffrey Zhao, Bartek Perz, Dian Yu, Heidi Howard, Adam Bloniarz, Jack W. Rae, Han Lu, Laurent Sifre, Marcello Maggioni, Fred Alcober, Dan Garrette, Megan Barnes, Shantanu Thakoor, Jacob Austin, Gabriel Barth-Maron, William Wong, Rishabh Joshi, Rahma Chaabouni, Deeni Fatiha, Arun Ahuja, Ruibo Liu, Yunxuan Li, Sarah Cogan, Jeremy Chen, Chao Jia, Chenjie Gu, Qiao Zhang, Jordan Grimstad, Ale Jakse Hartman, Martin Chadwick, Gaurav Singh Tomar, Xavier Garcia, Evan Senter, Emanuel Taropa, Thanumalayan Sankaranarayana Pillai, Jacob Devlin, Michael Laskin, Diego de Las Casas, Dasha Valter, Connie Tao, Lorenzo Blanco, Adrià Puigdomènech Badia, David Reitter, Mianna Chen, Jenny Brennan, Clara Rivera, Sergey Brin, Shariq Iqbal, Gabriela Surita, Jane Labanowski, Abhi Rao, Stephanie Winkler, Emilio Parisotto, Yiming Gu, Kate Olszewska, Yujing Zhang, Ravi Addanki, Antoine Miech, Annie Louis, Laurent El Shafey, Denis Teplyashin, Geoff Brown, Elliot Catt, Nithya Attaluri, Jan Balaguer, Jackie Xiang, Pidong Wang, Zoe Ashwood, Anton Briukhov, Albert Webson, Sanjay Ganapathy, Smit Sanghavi, Ajay Kannan, Ming-Wei Chang, Axel Stjerngren, Josip Djolonga, Yuting Sun, Ankur Bapna, Matthew Aitchison, Pedram Pejman, Henryk Michalewski, Tianhe Yu, Cindy Wang, Juliette Love, Junwhan Ahn, Dawn Bloxwich, Kehang Han, Peter Humphreys, Thibault Sellam, James Bradbury, Varun Godbole, Sina Samangooei, Bogdan Damoc, Alex Kaskasoli, Sébastien M. R. Arnold, Vijay Vasudevan, Shubham Agrawal, Jason Riesa, Dmitry Lepikhin, Richard Tanburn, Srivatsan Srinivasan, Hyeontaek Lim, Sarah Hodkinson, Pranav Shyam, Johan Ferret, Steven Hand, Ankush Garg, Tom Le Paine, Jian Li, Yujia Li, Minh Giang, Alexander Neitz, Zaheer Abbas, Sarah York, Machel Reid, Elizabeth Cole, Aakanksha Chowdhery, Dipanjan Das, Dominika Rogozińska, Vitaly Nikolaev, Pablo Sprechmann, Zachary Nado, Lukas Zilka,

Flavien Prost, Luheng He, Marianne Monteiro, Gaurav Mishra, Chris Welty, Josh Newlan, Dawei Jia, Miltiadis Allamanis, Clara Huiyi Hu, Raoul de Liedekerke, Justin Gilmer, Carl Saroufim, Shruti Rijhwani, Shaobo Hou, Disha Shrivastava, Anirudh Baddepudi, Alex Goldin, Adnan Ozturel, Albin Cassirer, Yunhan Xu, Daniel Sohn, Devendra Sachan, Reinald Kim Amplayo, Craig Swanson, Dessie Petrova, Shashi Narayan, Arthur Guez, Siddhartha Brahma, Jessica Landon, Miteyan Patel, Ruizhe Zhao, Kevin Villela, Luyu Wang, Wenhao Jia, Matthew Rahtz, Mai Giménez, Legg Yeung, Hanzhao Lin, James Keeling, Petko Georgiev, Diana Mincu, Boxi Wu, Salem Haykal, Rachel Saputro, Kiran Vodrahalli, James Qin, Zeynep Cankara, Abhanshu Sharma, Nick Fernando, Will Hawkins, Behnam Neyshabur, Solomon Kim, Adrian Hutter, Priyanka Agrawal, Alex Castro-Ros, George van den Driessche, Tao Wang, Fan Yang, Shuo yiin Chang, Paul Komarek, Ross McIlroy, Mario Lučić, Guodong Zhang, Wael Farhan, Michael Sharman, Paul Natsev, Paul Michel, Yong Cheng, Yamini Bansal, Siyuan Qiao, Kris Cao, Siamak Shakeri, Christina Butterfield, Justin Chung, Paul Kishan Rubenstein, Shivani Agrawal, Arthur Mensch, Kedar Soparkar, Karel Lenc, Timothy Chung, Aedan Pope, Loren Maggiore, Jackie Kay, Priya Jhakra, Shibo Wang, Joshua Maynez, Mary Phuong, Taylor Tobin, Andrea Tacchetti, Maja Trebacz, Kevin Robinson, Yash Katariya, Sebastian Riedel, Paige Bailey, Kefan Xiao, Nimesh Ghelani, Lora Aroyo, Ambrose Slone, Neil Houlsby, Xuehan Xiong, Zhen Yang, Elena Gribovskaya, Jonas Adler, Mateo Wirth, Lisa Lee, Music Li, Thais Kagohara, Jay Pavagadhi, Sophie Bridgers, Anna Bortsova, Sanjay Ghemawat, Zafarali Ahmed, Tianqi Liu, Richard Powell, Vijay Bolina, Mariko Iinuma, Polina Zablotskaia, James Besley, Da-Woon Chung, Timothy Dozat, Ramona Comanescu, Xiance Si, Jeremy Greer, Guolong Su, Martin Polacek, Raphaël Lopez Kaufman, Simon Tokumine, Hexiang Hu, Elena Buchatskaya, Yingjie Miao, Mohamed Elhawaty, Aditya Siddhant, Nenad Tomasev, Jinwei Xing, Christina Greer, Helen Miller, Shereen Ashraf, Aurko Roy, Zizhao Zhang, Ada Ma, Angelos Filos, Milos Besta, Rory Blevins, Ted Klimenko, Chih-Kuan Yeh, Soravit Changpinyo, Jiaqi Mu, Oscar Chang, Mantas Pajarskas, Carrie Muir, Vered Cohen, Charline Le Lan, Krishna Haridasan, Amit Marathe, Steven Hansen, Sholto Douglas, Rajkumar Samuel, Mingqiu Wang, Sophia Austin, Chang Lan, Jiepu Jiang, Justin Chiu, Jaime Alonso Lorenzo, Lars Lowe Sjösund, Sébastien Cevey, Zach Gleicher, Thi Avrahami, Anudhyan Boral, Hansa Srinivasan, Vittorio Selo, Rhys May, Konstantinos Aisopos, Léonard Hussenot, Livio Baldini Soares, Kate Baumli, Michael B. Chang, Adrià Recasens, Ben Caine, Alexander Pritzel, Filip Pavetic, Fabio Pardo, Anita Gergely, Justin Frye, Vinay Ramasesh, Dan Horgan, Kartikeya Badola, Nora Kassner, Subhrajit Roy, Ethan Dyer, Víctor Campos, Alex Tomala, Yunhao Tang, Dalia El Badawy, Elspeth White, Basil Mustafa, Oran Lang, Abhishek Jindal, Sharad Vikram, Zhitao Gong, Sergi Caelles, Ross Hemsley, Gregory Thornton, Fangxiaoyu Feng, Wojciech Stokowiec, Ce Zheng, Phoebe Thacker, Çağlar Ünlü, Zhishuai Zhang, Mohammad Saleh, James Svensson, Max Bileschi, Piyush Patil, Ankesh Anand, Roman Ring, Katerina Tsihlas, Arpi Vezer, Marco Selvi, Toby Shevlane, Mikel Rodriguez, Tom Kwiatkowski, Samira Daruki, Keran Rong, Allan Dafoe, Nicholas FitzGerald, Keren Gu-Lemberg, Mina Khan, Lisa Anne Hendricks, Marie Pellat, Vladimir Feinberg, James Cobon-Kerr, Tara Sainath, Maribeth Rauh, Sayed Hadi Hashemi, Richard Ives, Yana Hasson, YaGuang Li, Eric Noland, Yuan Cao, Nathan Byrd, Le Hou, Qingze Wang, Thibault Sottiaux, Michela Paganini, Jean-Baptiste Lespiau, Alexandre Moufarek, Samer Hassan, Kaushik Shivakumar, Joost van Amersfoort, Amol Mandhane, Pratik Joshi, Anirudh Goyal, Matthew Tung, Andrew Brock, Hannah Sheahan, Vedant Misra, Cheng Li, Nemanja Rakićević, Mostafa Dehghani, Fangyu Liu, Sid Mittal, Junhyuk Oh, Seb Noury, Eren Sezener, Fantine Huot, Matthew Lamm, Nicola De Cao, Charlie Chen, Gamaleldin Elsayed, Ed Chi, Mahdis Mahdieh, Ian Tenney, Nan Hua, Ivan Petrychenko, Patrick Kane, Dylan Scandinaro, Rishub Jain, Jonathan Uesato, Romina Datta, Adam Sadovsky, Oskar Bunyan, Dominik Rabiej, Shimu Wu, John Zhang, Gautam Vasudevan, Edouard Leurent, Mahmoud Alnahlawi, Ionut Georgescu, Nan Wei, Ivy Zheng, Betty Chan, Pam G Rabinovitch, Piotr Stanczyk, Ye Zhang, David Steiner, Subhajit Naskar, Michael Azzam, Matthew Johnson, Adam Paszke, Chung-Cheng Chiu, Jaume Sanchez Elias, Afroz Mohiuddin, Faizan Muhammad, Jin Miao, Andrew Lee, Nino Vieillard, Sahitya Potluri, Jane Park, Elnaz Davoodi, Jiageng Zhang, Jeff Stanway, Drew Garmon, Abhijit Karmarkar, Zhe Dong, Jong Lee, Aviral Kumar, Luowei Zhou, Jonathan Evens, William Isaac, Zhe Chen, Johnson Jia, Anselm Levskaya, Zhenkai Zhu, Chris Gorgolewski, Peter Grabowski, Yu Mao, Alberto Magni, Kaisheng Yao, Javier Snaider, Norman Casagrande, Paul Suganthan, Evan Palmer, Geoffrey Irving, Edward Loper, Manaal Faruqui, Isha Arkatkar, Nanxin Chen, Izhak Shafran, Michael Fink, Alfonso Castaño, Irene Giannoumis, Wooyeol

Kim, Mikołaj Rybiński, Ashwin Sreevatsa, Jennifer Prendki, David Soergel, Adrian Goedeckemeyer, Willi Gierke, Mohsen Jafari, Meenu Gaba, Jeremy Wiesner, Diana Gage Wright, Yawen Wei, Harsha Vashisht, Yana Kulizhskaya, Jay Hoover, Maigo Le, Lu Li, Chimezie Iwuanyanwu, Lu Liu, Kevin Ramirez, Andrey Khorlin, Albert Cui, Tian LIN, Marin Georgiev, Marcus Wu, Ricardo Aguilar, Keith Pallo, Abhishek Chakladar, Alena Repina, Xihui Wu, Tom van der Weide, Priya Ponnapalli, Caroline Kaplan, Jiri Simsa, Shuangfeng Li, Olivier Dousse, Fan Yang, Jeff Piper, Nathan Ie, Minnie Lui, Rama Pasumarthi, Nathan Lintz, Anitha Vijayakumar, Lam Nguyen Thiet, Daniel Andor, Pedro Valenzuela, Cosmin Paduraru, Daiyi Peng, Katherine Lee, Shuyuan Zhang, Somer Greene, Duc Dung Nguyen, Paula Kurylowicz, Sarmishta Velury, Sebastian Krause, Cassidy Hardin, Lucas Dixon, Lili Janzer, Kiam Choo, Ziqiang Feng, Biao Zhang, Achintya Singhal, Tejasi Latkar, Mingyang Zhang, Quoc Le, Elena Allica Abellan, Dayou Du, Dan McKinnon, Natasha Antropova, Tolga Bolukbasi, Orgad Keller, David Reid, Daniel Finchelstein, Maria Abi Raad, Remi Crocker, Peter Hawkins, Robert Dadashi, Colin Gaffney, Sid Lall, Ken Franko, Egor Filonov, Anna Bulanova, Rémi Leblond, Vikas Yadav, Shirley Chung, Harry Askham, Luis C. Cobo, Kelvin Xu, Felix Fischer, Jun Xu, Christina Sorokin, Chris Alberti, Chu-Cheng Lin, Colin Evans, Hao Zhou, Alek Dimitriev, Hannah Forbes, Dylan Banarse, Zora Tung, Jeremiah Liu, Mark Omernick, Colton Bishop, Chintu Kumar, Rachel Sterneck, Ryan Foley, Rohan Jain, Swaroop Mishra, Jiawei Xia, Taylor Bos, Geoffrey Cideron, Ehsan Amid, Francesco Piccinno, Xingyu Wang, Praseem Banzal, Petru Gurita, Hila Noga, Premal Shah, Daniel J. Mankowitz, Alex Polozov, Nate Kushman, Victoria Krakovna, Sasha Brown, MohammadHossein Bateni, Dennis Duan, Vlad Firoiu, Meghana Thotakuri, Tom Natan, Anhad Mohananey, Matthieu Geist, Sidharth Mudgal, Sertan Girgin, Hui Li, Jiayu Ye, Ofir Roval, Reiko Tojo, Michael Kwong, James Lee-Thorp, Christopher Yew, Quan Yuan, Sumit Bagri, Danila Sinopalnikov, Sabela Ramos, John Mellor, Abhishek Sharma, Aliaksei Severyn, Jonathan Lai, Kathy Wu, Heng-Tze Cheng, David Miller, Nicolas Sonnerat, Denis Vnukov, Rory Greig, Jennifer Beattie, Emily Caveness, Libin Bai, Julian Eisenschlos, Alex Korchemniy, Tomy Tsai, Mimi Jasarevic, Weize Kong, Phuong Dao, Zeyu Zheng, Frederick Liu, Fan Yang, Rui Zhu, Mark Geller, Tian Huey Teh, Jason Sanmiya, Evgeny Gladchenko, Nejc Trdin, Andrei Sozanschi, Daniel Toyama, Evan Rosen, Sasan Tavakkol, Linting Xue, Chen Elkind, Oliver Woodman, John Carpenter, George Papamakarios, Rupert Kemp, Sushant Kafle, Tanya Grunina, Rishika Sinha, Alice Talbert, Abhimanyu Goyal, Diane Wu, Denese Owusu-Afriyie, Cosmo Du, Chloe Thornton, Jordi Pont-Tuset, Pradyumna Narayana, Jing Li, Sabaer Fatehi, John Wieting, Omar Ajmeri, Benigno Uria, Tao Zhu, Yeongil Ko, Laura Knight, Amélie Héliou, Ning Niu, Shane Gu, Chenxi Pang, Dustin Tran, Yeqing Li, Nir Levine, Ariel Stolovich, Norbert Kalb, Rebeca Santamaria-Fernandez, Sonam Goenka, Wenny Yustalim, Robin Strudel, Ali Elqursh, Balaji Lakshminarayanan, Charlie Deck, Shyam Upadhyay, Hyo Lee, Mike Dusenberry, Zonglin Li, Xuezhi Wang, Kyle Levin, Raphael Hoffmann, Dan Holtmann-Rice, Olivier Bachem, Summer Yue, Sho Arora, Eric Malmi, Daniil Mirylenka, Qijun Tan, Christy Koh, Soheil Hassas Yeganeh, Siim Põder, Steven Zheng, Francesco Pongetti, Mukarram Tariq, Yanhua Sun, Lucian Ionita, Mojtaba Seyedhosseini, Pouya Tafti, Ragha Kotikalapudi, Zhiyu Liu, Anmol Gulati, Jasmine Liu, Xinyu Ye, Bart Chrzaszcz, Lily Wang, Nikhil Sethi, Tianrun Li, Ben Brown, Shreya Singh, Wei Fan, Aaron Parisi, Joe Stanton, Chenkai Kuang, Vinod Koverkathu, Christopher A. Choquette-Choo, Yunjie Li, TJ Lu, Abe Ittycheriah, Prakash Shroff, Pei Sun, Mani Varadarajan, Sanaz Bahargam, Rob Willoughby, David Gaddy, Ishita Dasgupta, Guillaume Desjardins, Marco Cornero, Brona Robenek, Bhavishya Mittal, Ben Albrecht, Ashish Shenoy, Fedor Moiseev, Henrik Jacobsson, Alireza Ghaffarkhah, Morgane Rivière, Alanna Walton, Clément Crepy, Alicia Parrish, Yuan Liu, Zongwei Zhou, Clement Farabet, Carey Radebaugh, Praveen Srinivasan, Claudia van der Salm, Andreas Fidjeland, Salvatore Scellato, Eri Latorre-Chimoto, Hanna Klimczak-Plucińska, David Bridson, Dario de Cesare, Tom Hudson, Piermaria Mendolicchio, Lexi Walker, Alex Morris, Ivo Penchev, Matthew Mauger, Alexey Guseynov, Alison Reid, Seth Odoom, Lucia Loher, Victor Cotruta, Madhavi Yenugula, Dominik Grewe, Anastasia Petrushkina, Tom Duerig, Antonio Sanchez, Steve Yadlowsky, Amy Shen, Amir Globerson, Adam Kurzrok, Lynette Webb, Sahil Dua, Dong Li, Preethi Lahoti, Surya Bhupatiraju, Dan Hurt, Haroon Qureshi, Ananth Agarwal, Tomer Shani, Matan Eyal, Anuj Khare, Shreyas Rammohan Belle, Lei Wang, Chetan Tekur, Mihir Sanjay Kale, Jinliang Wei, Ruoxin Sang, Brennan Saeta, Tyler Liechty, Yi Sun, Yao Zhao, Stephan Lee, Pandu Nayak, Doug Fritz, Manish Reddy Vuyyuru, John Aslanides, Nidhi Vyas, Martin Wicke, Xiao Ma, Taylan Bilal, Evgenii Eltyshev, Daniel Balle, Nina Martin, Hardie Cate, James Manyika, Keyvan Amiri, Yelin Kim, Xi Xiong, Kai Kang, Florian Luisier, Nilesh Tripuraneni, David Madras, Mandy Guo, Austin Waters,

Oliver Wang, Joshua Ainslie, Jason Baldridge, Han Zhang, Garima Pruthi, Jakob Bauer, Feng Yang, Riham Mansour, Jason Gelman, Yang Xu, George Polovets, Ji Liu, Honglong Cai, Warren Chen, XiangHai Sheng, Emily Xue, Sherjil Ozair, Adams Yu, Christof Angermueller, Xiaowei Li, Weiren Wang, Julia Wiesinger, Emmanouil Koukoumidis, Yuan Tian, Anand Iyer, Madhu Gurumurthy, Mark Goldenson, Parashar Shah, MK Blake, Hongkun Yu, Anthony Urbanowicz, Jennimaria Palomaki, Chrisantha Fernando, Kevin Brooks, Ken Durden, Harsh Mehta, Nikola Momchev, Elahe Rahimtoroghi, Maria Georgaki, Amit Raul, Sebastian Ruder, Morgan Redshaw, Jinhyuk Lee, Komal Jalan, Dinghua Li, Ginger Perng, Blake Hechtman, Parker Schuh, Milad Nasr, Mia Chen, Kieran Milan, Vladimir Mikulik, Trevor Strohman, Juliana Franco, Tim Green, Demis Hassabis, Koray Kavukcuoglu, Jeffrey Dean, and Oriol Vinyals. 2023. Gemini: A family of highly capable multimodal models.

Shubham Toshniwal, Ivan Moshkov, Sean Narenthiran, Daria Gitman, Fei Jia, and Igor Gitman. 2024. Openmathinstruct-1: A 1.8 million math instruction tuning dataset.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need.

Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. 2024. Math-shepherd: Verify and reinforce llms step-by-step without human annotations.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. 2023. Mammoth: Building math generalist models through hybrid instruction tuning.

Exploring Internal Numeracy in Language Models: A Case Study on ALBERT

Ulme Wennberg, Gustav Eje Henter
Division of Speech, Music and Hearing, KTH Royal Institute of Technology
Lindstedtsvägen 24, SE-100 44 Stockholm, Sweden
ulme@kth.se, ghe@kth.se

Abstract

It has been found that Transformer-based language models have the ability to perform basic quantitative reasoning. In this paper, we propose a method for studying how these models internally represent numerical data, and use our proposal to analyze the ALBERT family of language models. Specifically, we extract the learned embeddings these models use to represent tokens that correspond to numbers and ordinals, and subject these embeddings to Principal Component Analysis (PCA). PCA results reveal that ALBERT models of different sizes, trained and initialized separately, consistently learn to use the axes of greatest variation to represent the approximate ordering of various numerical concepts. Numerals and their textual counterparts are represented in separate clusters, but increase along the same direction in 2D space. Our findings illustrate that language models, trained purely to model text, can intuit basic mathematical concepts, opening avenues for NLP applications that intersect with quantitative reasoning.

Keywords: Language models; Transformer-based models; Numerical data representation; Word embeddings; PCA; Numerals in NLP

1. Introduction

The Transformer architecture introduced by Vaswani et al. (2017) has led to major advances in computational linguistics. Transformer-based models of language like BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), ALBERT (Lan et al., 2020), and ELECTRA (Clark et al., 2020) have excelled in a range of tasks, from machine translation (Lample et al., 2018; Gu et al., 2018) to question answering (Yamada et al., 2020) and beyond. Studies have also evaluated these models' numerical reasoning, For example, Saxton et al. (2019) found a Transformer-based language model to perform at an E-grade level on a British math exam for 16-year-olds. Kalyan et al. (2021) subsequently developed benchmarks for mathematical commonsense reasoning to track the progress of models in this respect, highlighting the growing interest in this research area.

Despite demonstrable performance on numerical tasks, the origin of these abilities in text-trained models, and the root of their numeracy and quantitative understanding, remain obscure. In this paper, we aim to illuminate this aspect by analyzing the learned embeddings these models use to represent lexical tokens internally. We study how the models have learned to embed numerals and their written representations, unearthing evidence that various embedding vectors capture the essence of numerical concepts. Instead of studying whether the embeddings of different numbers are distributed close together (like one might cluster the embeddings of synonyms; cf. Mikolov et al. (2013)), we thus consider how their representations *differ*, and how the axes of greatest variation among these concepts relate to the intrinsic ordering and numeric value of the different tokens.

Our paper makes two contributions:

1. We propose a novel way to study internal numerical cognition in language models.
2. We use our proposed method to investigate how ALBERT encodes numerical and ordinal information, and how this varies across different versions of ALBERT of various sizes, independently trained.

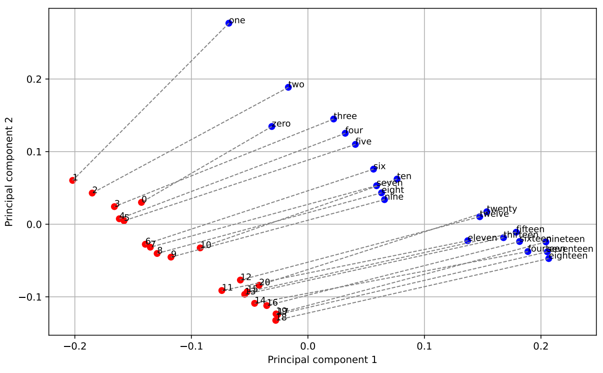
Using our proposed method, we find:

- Trained ALBERT models consistently use primary principal component axes to denote ordering and spacing of numbers, ordinals, and magnitude orders.
- The representations are closer together for higher values, suggesting a logarithmic representation of numbers.
- Numerals and their textual counterparts are represented in separate clusters, but increase along the same direction in 2D PCA space.

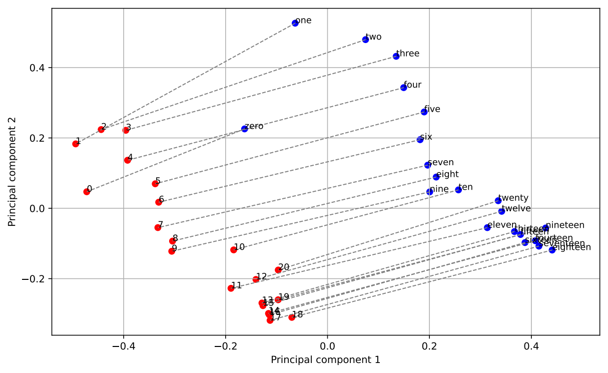
2. Background

In this section, we give a background on numeracy in language models and highlight previous works investigating their internal token representations.

Numeracy is critical for complex reasoning in NLP. Investigations into models' numerical reasoning abilities (Wallace et al., 2019; Jin et al., 2021; Thawani et al., 2021; Duan et al., 2021; Sakamoto and Aizawa, 2021) have shown promising enhancements in model numeracy. Further studies (Kim et al., 2021; Lin et al., 2020; Shah et al., 2023) have



(a) albert-base-v2



(b) albert-xxlarge-v2

Figure 1: Visualization of the two first principal components of word embeddings for numbers zero through twenty and their textual counterparts in two ALBERT models.

explored models' abilities to extrapolate and their numerical commonsense knowledge. In conjunction with these explorations, recent methodologies (Sundararaman et al., 2022; Saeed and Papotti, 2022; Jiang et al., 2020; Liang et al., 2022) introduce a variety of approaches to improve numerical representation and processing, showing ongoing efforts to refine numeracy in language models.

Internally, Transformers use self-attention to capture dependencies between all pairs of input vectors. The models use a variety of mechanisms to represent positional information for sequential input data. One common approach, used already in Vaswani et al. (2017), is to encode each token w_i as a vector that is the sum of a content-embedding vector (that depends only on the token w_i) and a position-encoding vector (that depends only on i, the position in the input sequence). The position-dependence mechanisms either explicitly represent the inherent ordering of input symbols, or, when they do not, have been shown to learn positional representations that capture both this ordering and the translation equivariance of text sequences (Wennberg and Henter, 2021).

Other research (e.g., Mikolov et al., 2013; Vylomova et al., 2016; Durrani et al., 2022) has sought to shed light on information processing in neural language models by analyzing their learned embeddings of different words, concepts, and lexical tokens. A consistent finding is that synonyms cluster together in latent space, meaning that linguistic similarity is reflected internally in the learned model. In this work, we apply a similar analysis, but to concepts that are numerical rather than linguistic. The key difference between our present study and prior work on language-model numeracy is that we look directly at the internal embeddings that Transformer-based language models have learned for numerical concepts, and investigate to what extent *differences* between these embeddings are reflective of differences in numerical value between the mathematical concepts they represent.

3. Experiments

We now describe the method and results of our study of the word embeddings inside eight different Transformer-based language models, namely the ALBERT family (Lan et al., 2020). We choose to study ALBERT because it is available in four different model sizes (starting at "base" and going up to "xxlarge"), each with checkpoints at two different points during training ("v1" vs. "v2", with v2 having been trained for longer), allowing for a comparison of embeddings in different models and their evolution. In our analysis, we specifically examine numerical ranges from zero to twenty and one to one hundred, not only to cover a broad spectrum of basic and multi-digit numerals but also because these numbers are consistently tokenized as single tokens by the ALBERT models (Lan et al., 2020). This choice aligns with our objective to study unambiguous, uncontextualized numerical representations within the model. Many submissions to the GLUE (Wang et al., 2019) and SuperGLUE (Wang et al., 2020) leaderboards are descendants of the ALBERT architecture.

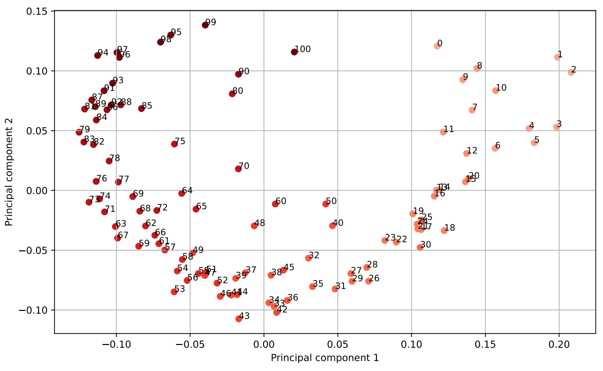
3.1. Analysis Methodology

All the analyses in this paper follow the same underlying recipe:

First, we extract uncontextualized embeddings for selected tokens (single-token words only). These word embeddings are prior to position embedding addition or self-attention layer processing.

We then conduct PCA on these embeddings to identify principal variation axes. This is a linear dimensionality reduction technique, meaning that linear structures like a number line are preserved.

Lastly, we plot embeddings along principal component axes to assess if they capture mathematical concept ordering and if distances reflect mathematical relationships, like proximity of similar numbers.


(a) albert-base-v2

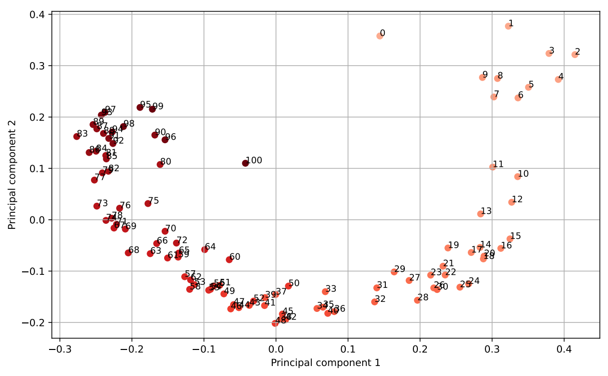

(b) albert-xxlarge-v2

Figure 2: The first and second PCA components for all numbers 1 to 100 in two different ALBERT models.

3.2. Numerical vs. Lexical Embedding

We first compare the learned embeddings of the numbers zero through twenty, juxtaposed with their word representations (e.g., “7” versus “seven”), across different ALBERT models. The results of applying our PCA-based analysis to the resulting 42 different representations is visualized in Figure 1 for the smallest and largest ALBERT models (the other models yield very similar-looking plots).

A number of observations are immediately apparent in the figure:

1. Numbers and words representing occupy two distinct, elongated clusters.
2. Within each cluster, there is a direction along which numerical values generally increase. In other words, the values are mostly in order, and we (approximately) recover a number line in the PCA space for each cluster (numbers vs. number words).
3. The direction along which the values increase is the same for both numbers and number words. It would thus easily be possible, particularly for the bigger model, to project the embeddings onto a single axis in PCA space that approximates the number line.
4. When values exceed ten, numbers begin to bunch up more.

The fact that the two different kinds of embeddings can be projected onto something like the number line strongly suggests a learned ability to link numerical symbols to their word forms, and to their approximate value and ordering.

We can also make some minor observations about individual numbers, such as the positions of the numbers and words for zero being idiosyncratic, and (more curiously) that numbers and words for twenty also consistently are out of place.

3.3. Numbers 1 Through 100

Next, we performed the same analysis on integers 0 to 100 (excluding word forms) and charted their 2D PCA distribution for the same two ALBERT models. The findings, displayed in Figure 2, mirror those from other models. We observe that:

1. As numbers increase, they approximately trace out a horseshoe shape in 2D space.
2. Larger numbers gradually compress closer together, especially for the larger model.
3. Rounded numbers (i.e., those ending in zero) lie closer to middle of the space. This is more visible for the smaller model, but true for both. 25, 75, and numbers with many powers of two are also closer to the middle. 100, with two zeroes, sticks out particularly much.

The most important conclusion is that the ability to use embeddings to order numbers by size persists into larger numbers, though the spacing gets more compressed as the numerical values increase.

3.4. Representing Orders of Magnitude

Having looked at numbers up to 100, we also studied the embeddings of words for different orders of magnitude. Specifically, we performed PCA on the embedding representations of the words “hundred,” “thousand,” “million,” “billion,” and “trillion.” Figure 3 shows these words’ positions on the first principal axis across eight ALBERT models. We see that:

1. The words always respect the expected ordering based on their numerical value.
2. The separation between “hundred” and “thousand” is consistently the shortest, typically by some margin. This evokes comparisons to the logarithmic axis at the bottom of the figure.

There is a close call between “hundred” and “thousand” for the xlarge model, but the separation increases with longer training (model v1 vs. v2).

3.5. Words for Ordinals

As our last experiment, we visualize the representation of words for ordinals rather than numerals. Specifically, we apply the same PCA-based method to the embeddings of the terms

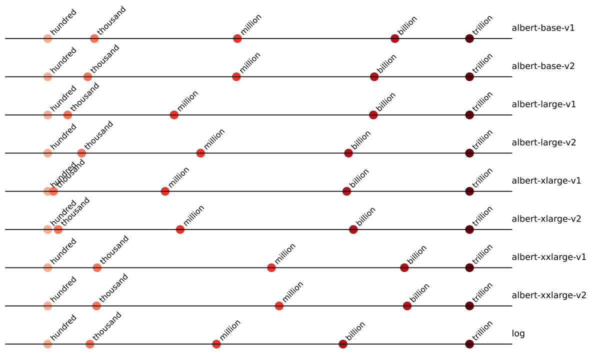


Figure 3: Orders-of-magnitude word embeddings visualized along the first PCA axis across eight ALBERT configurations. Axes have been affinely transformed so that the first and last embeddings line up vertically. The last row shows the concepts arranged on a logarithmic axis for comparison.

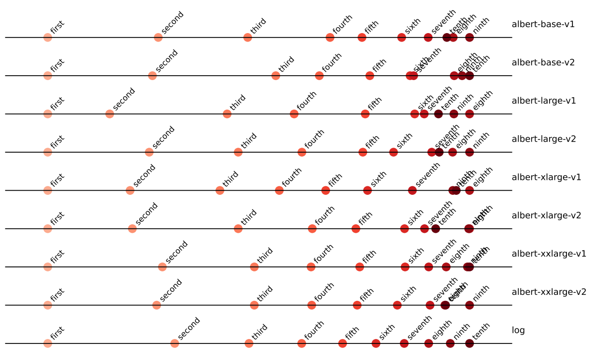


Figure 4: Visualization of ordinal term embeddings along the first PCA axis across eight ALBERT configurations. The axes have been affinely transformed so that the first and last embeddings line up vertically. The last row shows the concepts arranged on a logarithmic axis for comparison.

"first" through "tenth" and visualize the first principal axis for all eight ALBERT models, like in the previous section. The results are shown in Figure 4, from which we make the following observations:

1. Ordinals consistently appear in the correct order along the principal axis of variation up until and including "seventh".
2. The distance between ordinals gradually decreases as the numbers increase, with the last three ordinals generally being close together and often out of order.

Embeddings do not become obviously better with longer training, especially as they already mostly appear in the correct order for the v1 models.

4. Discussion

Reviewing all four analyses, it's evident that ALBERT models' internal representation of various numbers and numerical concepts in the embedding layer directly reflects their numerical value. The representations are closer to a logarithmic than a linear scale. These trends are very consistent across models of different sizes and trained for different amounts of time.

While the fact that Transformer-based language models can support simple mathematical reasoning has indicated some level of numeracy within the model, we can now open the black box and see that numerical knowledge evident within the basic vector representations inside the model. It is not at all obvious that reasonable representations of numerical concepts would arise in these models, given that they are pre-trained exclusively on text to optimize standard language-modeling objectives, with no direct mathematical training.

The observation that larger numbers cluster closer, hinting at logarithmic scaling, and the unique behavior of round numbers in Figure 2, may stem from their occurrence frequency in data, aligning with Benford's law (Benford, 1938). This law suggests that smaller leading digits are more common in real-world numerical data, yielding a near uniform distribution of digits on a logarithmic scale.

A notable limitation of our study is its focus on single-token numbers, which excludes decimals and larger numerical values from our analysis.

5. Conclusion

We have introduced a novel approach to analyzing the quality of numerical representations in language models. This offers insights into model numeracy, which matters for developing improved numerical-understanding capabilities for Transformer-based language models.

We use our method to investigate how ALBERT, an important Transformer-based language model architecture, represents different numerical and ordinal inputs. Our results demonstrate a clear concept of numerical ordering within the vector representations inside the model. Representations of larger numbers fall closer together, suggestive of models using logarithmic axis representations internally. The findings are very robust, in that they appear essentially unchanged across eight different models that differ in size and training duration.

Going beyond numerical order, future work should seek to quantify to what extent learned internal structures reflect interval and ratio scales, as well as to what extent factors like model architecture and term frequency in the corpus contribute to (or otherwise influence) these structures. Another goal is to extend the analysis to multi-token numbers and mathematical operators, and connect with emerging understanding of how models then perform stepwise mathematical processing in latent space (Lee et al., 2019; Valentino et al., 2023).

6. Acknowledgments

We thank the anonymous reviewers for constructive feedback. This research was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

7. Bibliographical References

Frank Benford. 1938. The law of anomalous numbers. *Proc. APS*, pages 551–572.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Pre-training transformers as energy-based cloze models. In *Proc. EMNLP*, pages 285–294.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. NAACL-HLT*, pages 4171–4186.

Hanyu Duan, Yi Yang, and K. Tam. 2021. Learning numeracy: A simple yet effective number embedding approach using knowledge graph. In *Proc. EMNLP*.

Nadir Durrani, Hassan Sajjad, Fahim Dalvi, and Firoj Alam. 2022. On the transformation of latent space in fine-tuned NLP models. In *Proc. EMNLP*, pages 1495–1516.

Jiatao Gu, Yong Wang, Yun Chen, Kyunghyun Cho, and Victor O. K. Li. 2018. Meta-learning for low-resource neural machine translation. In *Proc. EMNLP*, pages 3622–3631.

Chengyue Jiang, Zhonglin Nian, et al. 2020. Learning numeral embedding. *arXiv preprint arXiv:2001.00003*.

Zhihua Jin, Xin Jiang, et al. 2021. NumGPT: Improving numeracy ability of generative pre-trained models. *arXiv preprint arXiv:2101.03137*.

Ashwin Kalyan, Abhinav Kumar, Arjun Chandrasekaran, Ashish Sabharwal, and Peter Clark. 2021. How much coffee was consumed during EMNLP 2019? Fermi problems: A new reasoning challenge for AI.

Jeonghwan Kim, Giwon Hong, Kyung min Kim, Junmo Kang, and Sung-Hyon Myaeng. 2021. Have you seen that number? investigating extrapolation in question answering models. In *Proc. EMNLP*.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Phrase-based & neural unsupervised machine translation. In *Proc. EMNLP*, pages 5039–5049.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *Proc. ICLR*.

Dennis Lee, Christian Szegedy, Markus N. Rabe, Sarah M. Loos, and Kshitij Bansal. 2019. Mathematical reasoning in latent space.

Zhenwen Liang, Jipeng Zhang, et al. 2022. MWP-BERT: Numeracy-augmented pre-training for math word problem solving. In *Proc. NAACL*.

Bill Yuchen Lin, Seyeon Lee, Rahul Khanna, and Xiang Ren. 2020. Birds have four legs?! numersense: Probing numerical commonsense knowledge of pre-trained language models. In *Proc. EMNLP*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *Proc NIPS*, 26.

Mohammed Saeed and Paolo Papotti. 2022. You are my type! type embeddings for pre-trained language models. In *Proc. EMNLP*.

Taku Sakamoto and Akiko Aizawa. 2021. Predicting numerals in natural language text using a language model considering the quantitative aspects of numerals. *Proc. DEELIO*.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. 2019. Analysing mathematical reasoning abilities of neural models. In *Proc. ICLR*.

Raj Sanjay Shah, Vijay Marupudi, Reba Koenen, Khushi Bhardwaj, and S. Varma. 2023. Numeric magnitude comparison effects in large language models. In *Proc. ACL*.

Dhanasekar Sundararaman, Vivek Subramanian, Guoyin Wang, Liyan Xu, and L. Carin. 2022. Improving downstream task performance by treating numbers as entities. In *Proc. ICLR*.

Avijit Thawani, Jay Pujara, and Fredrik Ilievski. 2021. Numeracy enhances the literacy of language models. In *Proc. EMNLP*.

Marco Valentino, Jordan Meadows, Lan Zhang, and André Freitas. 2023. Multi-operational mathematical derivations in latent space.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. NIPS*, pages 5998–6008.

Ekaterina Vylomova, Laura Rimell, Trevor Cohn, and Timothy Baldwin. 2016. Take and took, gaggle and goose, book and read: Evaluating the utility of vector differences for lexical relation learning. In *Proc. ACL*, pages 1671–1682.

Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. 2019. Do NLP models know numbers? probing numeracy in embeddings. In *Proc. EMNLP*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2020. Superglue: A stickier benchmark for general-purpose language understanding systems.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Glue: A multi-task benchmark and analysis platform for natural language understanding.

Ulme Wennberg and Gustav Eje Henter. 2021. The case for translation-invariant self-attention in transformer-based language models. In *Proc. ACL*.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. LUKE: Deep contextualized entity representations with entity-aware self-attention. In *Proc. EMNLP*, pages 6442–6454.

Author Index