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.

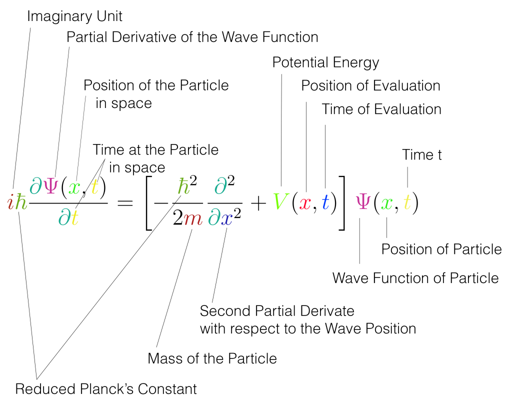
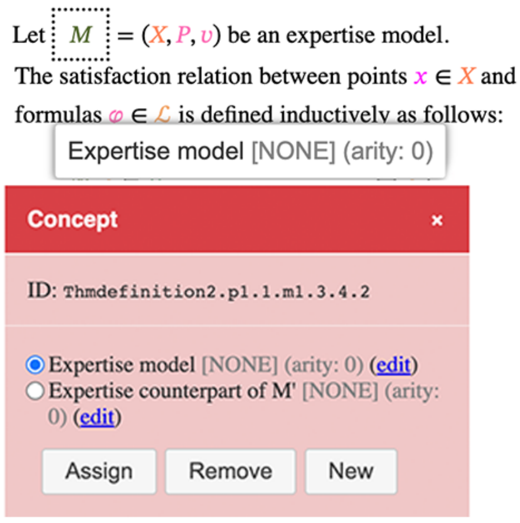

Imaginary Unit
Partial Derivative of the Wave Function
Position of the Particle in space
Potential Energy
Position of Evaluation
Time of Evaluation
Time at the Particle in space
Time t
Position of Particle
Wave Function of Particle
Second Partial Derivate with respect to the Wave Position
Mass of the 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 $\boxed{M}$ = (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) pre-processing 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.

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

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

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-

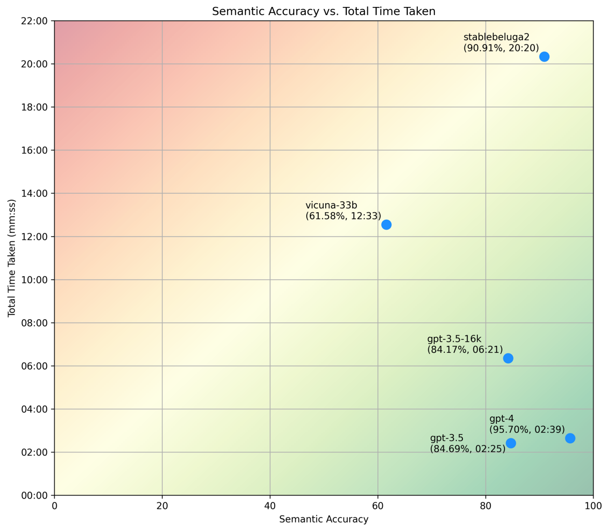


Figure 8: Scatter plots of the Total Time Taken for all five LLMs. GPT-4, despite its high costs, emerged as the most impressive model due to its superior performance, while GPT-3.5 turned out to be the most cost-effective and fastest model to operate.

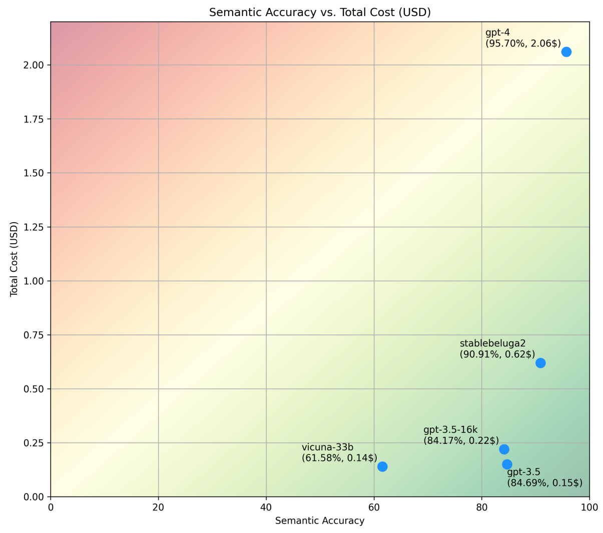


Figure 9: Total Cost (USD) vs. Semantic Accuracy for all five LLMs. GPT-4, despite its high costs, emerged as the most impressive model due to its superior performance, while GPT-3.5 turned out to be the most cost-effective and fastest model to operate.

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>
```