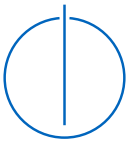
SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

# An Approach to Coreference Resolution and Formula Grounding for Mathematical Identifiers using Large Language Models

Aamin Dev



SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

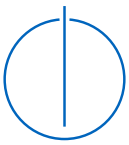
TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

# An Approach to Coreference Resolution and Formula Grounding for Mathematical Identifiers using Large Language Models

# Ein Ansatz zur Auflösung von Koreferenzen und zur Ermittlung von Formeln für mathematische Symbole mit Hilfe von Large Language Models

| | |
|---|---|
| Author: | Aamin Dev |
| Supervisor: | Prof. Dr. Georg Groh, Prof. Dr. Yusuke Miyao |
| Advisor: | Miriam Anschütz, Takuto Asakura |
| Submission Date: | 2023-09-15 |



I confirm that this bachelor's thesis is my own work and I have documented all sources and material used.

Garching bei. München, 2023-09-15 Aamin Dev

# Acknowledgments

I would like to thank all the members of Panikecke[1] for being there for me all the time and supporting me in my personal life. GPT and Rune Sætre (NTNU) for proof-reading my Thesis.

[1] https://discord.gg/RURaVwRe

# Abstract

***TODO: KEEP UPDATING ABSTRACT WHILE WRITING***

This thesis introduces a novel method for the automated annotation of mathematical identifiers in scientific papers, leveraging Large Language Models (LLMs) such as GPT-3.5 and GPT-4 from OpenAI. The approach addresses the challenges of co-reference resolution and formula grounding, traditionally handled by human annotators through costly and time-intensive procedures. Our study utilises a Math Identifier-oriented Grounding Annotation Tool (MioGatto), and explores the potential of integrating Part-of-Speech (POS) tagging and other technologies (like ?) assisting in the process. The critical component of this research is developing a procedure for generating a dictionary of mathematical identifiers, contextualising their various meanings, and enabling the language model to select the most accurate definition based on the given context. This method demonstrates the impressive capability of LLMs to disambiguate meanings based on context, a vital task due to the inherently polysemous nature of mathematical identifiers. The preliminary results of 82.36 on a Computational Natural Language Learning (CoNLL) test set makes our approach a potential game-changer in mathematical text annotation, significantly reducing time and financial costs. The findings show the (untapped) potential for using general purpose LLMs in specific mathematical language understanding.

# Contents

# 1 Introduction

Mathematical formulas, integral to Science, Technology, Engineering and Mathematics (STEM) papers, often challenge readers due to the ambiguous use of variables or identifiers and their associated meanings or definitions. The constraints of the English and Greek alphabets, being finite, frequently lead to the reuse of the same identifiers with varying definitions contingent upon context. This ambiguity can be particularly daunting for those unfamiliar with the subject matter. As the visual representation in Figure 1.1 illustrates, disambiguating identifiers within mathematical formulas presents a considerable challenge. This is particularly evident when the same variable serves multiple roles, each underpinned by a distinct definition. Such ambiguities become even more problematic for readers not profoundly acquainted with the subject matter, complicating the already demanding task of comprehending each identifier's specific meaning within the context of a complex formula.

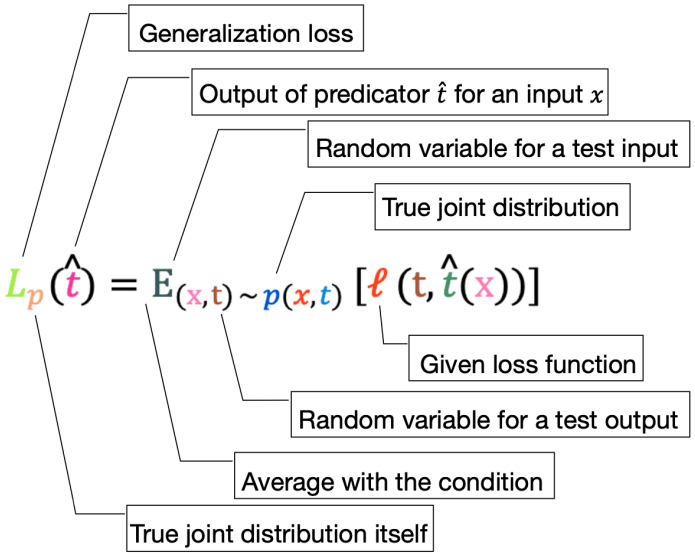



Figure 1.1: Challenges in the Disambiguation of Mathematical Formulas

MioGatto [1] (Asakura, Miyao, Aizawa, and Kohlhase, 2021), a Math Identifier-oriented Grounding Annotation Tool, was conceived to address this issue. However, manu-

[1] https://github.com/wtsnjp/MioGatto/

ally annotating papers to define each identifier is a new challenge. This process is time-consuming and resource-intensive, often requiring a whole day for annotation, depending on the paper's length.

The concept of grounding mathematical formulas (Asakura, Greiner-Petter, et al., 2020) offers a promising solution. By automating this process, we can significantly expedite the annotation process. This thesis adopts a predominantly data-driven approach to develop an automation tool, which we have realised through the following stages:

1. **Detecting/Retrieving:** This phase entails the transformation of the LaTeX source into a machine-readable HTML/XML format using LaTeXML [2] (Ginev et al., 2011).
2. **Dictionary Generation:** Leveraging LLMs, short text clustering techniques will be employed to construct a comprehensive dictionary of all mathematical identifiers as keys with their respective possible descriptions as the values.
3. **Association of Each Occurrence:** This step will involve associating every instance of a mathematical identifier with its corresponding definition, drawing inspiration from MathAlign (Alexeeva et al., 2020).

## 1.1 Motivation and Problem

Annotating mathematical identifiers in scientific papers is a cornerstone for enhancing comprehension. Traditionally, this task has been performed manually, a method that, while effective, is fraught with challenges:

- **Time-Consuming:** Manual annotation is inherently labour-intensive, often requiring hours or even days for a single paper.
- **Accessibility:** The expertise and resources required for manual annotation are not universally available, limiting its reach.
- **Cost Implications:** The adage "time is money" holds here. The extended hours spent on manual annotation translate to increased financial costs.

Given these constraints, there is a pressing need for a solution that's both efficient and universally accessible. Automation emerges as a promising alternative, potentially reducing annotation time from days to minutes and democratising access for a wider audience.

However, the path to automation has its challenges. Traditional Natural Language Processing (NLP) techniques, such as POS tagging or the establishment of formal grammar, tend to oversimplify the problem. These methods often provide a generalised

[2] `https://math.nist.gov/~BMiller/LaTeXML/`

solution, covering only a fraction of the diverse challenges presented by mathematical annotations.

In recent years, LLMs have shown immense promise in various NLP tasks. Their capacity to understand and generate context-rich text suggests they could be pivotal in automating the annotation process, provided they are harnessed effectively.

## 1.2 Research Questions

The primary objective of this research is to explore the feasibility and effectiveness of using LLMs to automate mathematical identifier annotations in scientific papers. To guide this investigation, we have formulated the following research questions:

1. **Efficacy of LLMs:** How effective are LLMs, specifically GPT-3.5 (OpenAI, 2023a) and GPT-4 (OpenAI, 2023b) and some Open Source LLMs, in generating accurate annotations for mathematical identifiers compared to traditional manual methods?
2. **Contextual Understanding:** To what extent can LLMs disambiguate mathematical identifiers based on context, given the inherent polysemy of these identifiers?
3. **Coverage of Annotation:** What percentage of a scientific paper can LLMs effectively annotate?
4. **Accuracy concerning Ground Truth:** How closely do the annotations generated by LLMs align with the ground truth provided by manual annotations?
5. **Efficiency:** How does the automation process using LLMs impact the time required for annotating scientific papers, and what are the implications for cost savings?
6. **Limitations of Automation:** What are the potential pitfalls or limitations of using LLMs for this automation task?

These questions provide a comprehensive understanding of the potential and challenges of using LLMs to automate mathematical identifier annotations.

## 1.3 Contributions

The journey of this research has led to several significant advancements in the realm of automated mathematical identifier annotations. The contributions of this thesis can be enumerated as follows:

1. **Integration with MioGatto:** Successfully incorporated GPT-based annotation capabilities into the MioGatto platform, enhancing its potential for automated paper annotations.

2. **Extensive Annotations:** Employed a range of LLMs, including GPT-3.5-turbo, GPT-3.5-turbo-16k, GPT-4, vicuna-33b (Zheng et al., 2023) [3], and StableBeluga2 (Mahan et al., n.d.; Touvron et al., 2023; Mukherjee et al., 2023) [4], to annotate a curated set of 40 scientific papers. This extensive annotation process serves as a comprehensive evaluation of the capabilities of these models.

3. **Performance Evaluation:** Conducted a thorough evaluation and analysis of the annotation performances of GPT, vicuna-33b, and StableBeluga2 LLMs, providing insights into their strengths and limitations.

4. **Ground Truth Annotation:** Personally annotated a subset of papers to establish a ground truth, ensuring a reliable benchmark for evaluating the automated annotations.

5. **CoNLL Score Approximation:** Developed a novel formula to approximate the expected CoNLL (Pradhan et al., 2012) score of a paper when annotated using GPT, offering a predictive tool for assessing annotation quality.

These contributions not only advance the field of automated annotation but also lay the groundwork for future research endeavours in this domain.

## 1.4 Outline

This thesis is structured to comprehensively understand the challenges, methodologies, and outcomes of automating mathematical identifier annotations using Large Language Models. The subsequent chapters are organised as follows:

1. **Introduction:** This chapter sets the stage by introducing the study's motivation, research questions, and contributions, offering readers a contextual foundation for the subsequent chapters.

2. **Related Work:** While the concept explored in this thesis is novel, this chapter delves into the limited existing literature that shares thematic similarities, providing a backdrop against which the current research can be contrasted.

3. **Methods:** This chapter chronicles the journey of methodological exploration. It begins with initial attempts using traditional techniques like parts of speech tagging. It transitions into the more successful GPT strategies, detailing the various prompts and markers employed to optimise results.

[3] `https://huggingface.co/TheBloke/Vicuna-33B-1-3-SuperHOT-8K-GPTQ`
[4] `https://huggingface.co/TheBloke/StableBeluga2-70B-GPTQ`

4. **Results:** Here, the empirical outcomes of the research are presented. The chapter elucidates the scores achieved by each LLM, the methodologies used to derive these scores, and the rationale behind selecting specific methods.

5. **Analysis:** This chapter delves deep into the interpretation of the results. It provides insights into the significance of the outcomes and their correlation with other findings and introduces the novel formula developed during the research.

6. **Conclusion:** This chapter focuses on synthesising the research findings, answering the questions posed at the outset and drawing conclusions on the study's implications and contributions.

7. **Future Works:** This thesis culminates with outlining potential avenues for further exploration and improvement, such as model combinations or the incorporation of emerging techniques.

This structured approach ensures a logical flow, guiding readers from the foundational concepts to the conclusions and offering a holistic understanding of the research journey.

# 2 Related Work

The burgeoning field of Mathematical Language Processing (MLP) is increasingly focused on the complexities of understanding, annotating, and disambiguating mathematical text. This chapter offers a critical review of seminal works that have shaped the trajectory of this domain, thereby contextualising the relevance of the present study.

## 2.1 Understanding Mathematical Text and Formulae

Meadows et al. (2022)'s research emphasises the importance of informal mathematical text in quantitative reasoning and advocates using transformer models like GPT in formula retrieval. This notion is further extended by the MLP project (Pagael et al., 2014), which delves into the semantics of identifiers in mathematical formulae. The project contrasts traditional pattern-matching techniques with a novel approach that employs POS tag-based distances to deduce identifier-definition probabilities. Grigore et al. (2009)'s work complements these studies by exploring the complexities of interpreting symbolic expressions in mathematical narratives, highlighting the value of linguistic context.

## 2.2 Foundational Frameworks and Automation

The pioneering work by Asakura, Greiner-Petter, et al. (2020) serves as a cornerstone in the field, delineating the importance of grounding mathematical formulae. It delineates the importance of anchoring mathematical formulae. The authors champion the indispensable role of MLP in deciphering STEM manuscripts and introduce MioGatto, a cutting-edge annotation tool. Ding et al. (2022)'s investigation complements this and Schubotz et al. (2017)'s study, which collectively underscores the transformative potential of automation in data annotation. While Ding focuses on the role of GPT-3 in reducing annotation overheads, Schubotz sets the stage for automated extraction frameworks for mathematical identifier definitions. These works collectively inform the methodology and insights that have shaped our approach. Alexeeva et al. (2020) extend this discourse by introducing machine reading techniques for extracting mathematical concepts, offering a rule-based strategy that aligns LaTeX representations with textual counterparts.

## 2.3 Role of Large Language Models and Pre-trained Frameworks

The advent of pre-trained models like "MathBERT" (Peng et al., 2021) and the evaluation of GPT-3.5 (He et al., 2023) mark significant milestones in the field. MathBERT is fine-tuned for decoding mathematical formulas and emphasises the importance of context. At the same time, the research on GPT-3.5 evaluates its efficacy as a robust annotator, questioning its potential to replace traditional crowdsourced methods. These works collectively highlight the untapped potential of Large Language Models in automating the annotation process.

## 2.4 Conclusion

The body of work reviewed herein provides a comprehensive overview of the challenges and innovations that define the landscape of mathematical language processing. As the field evolves, these cornerstone studies offer invaluable insights, setting the stage for future advancements in this domain.

# 3 Methodology

This chapter delineates the methodology employed in this research, encompassing three core stages: detection and retrieval of mathematical identifiers, dictionary construction, and association of individual instances. The process capitalises on LaTeXML utilities and explores the capabilities of LLMs to automate the annotation of mathematical identifiers. Herein, we present an exhaustive account that traces the transition from source materials to machine-readable formats and from dictionary formulation to final annotation.

Before delving into the specifics, it is crucial to understand the structure of MioGatto [1] (Asakura, Miyao, Aizawa, and Kohlhase, 2021), the tool we aim to automate. MioGatto comprises three core files:

1. `source.html`—a pre-processed HTML file suitable for web rendering, illustrated in Figure 3.1
2. `mcdict.json`—a JSON file containing a list of possible descriptions for each identifier (Figure 3.2a),
3. `anno.json`—another JSON file that holds each identifier's index of the chosen description (Figure 3.2b).

The generation of these files corresponds to different sections in this chapter. Section 3.1 focuses on creating `source.html`. Section 3.2 covers the generation of `mcdict.json`, and Section 3.3 addresses `anno.json`. These files are interlinked by IDs generated during pre-processing. The structure of the JSON files is depicted in Figure 3.2.

## 3.1 Identifier Detection and Retrieval

Initially, we considered parsing the LaTeX code directly via GPT-based LLMs, owing to their frequent training on LaTeX documents. LaTeX not only lends a more decadent semantic layer to mathematical identifiers but also offers efficiency in token usage and complexity. To illustrate the effectiveness of different encodings, Table 3.1 displays various representations of identical formulae, comparing LaTeX, ASCII Math, and MathML (XML) regarding readability and token count.

While LaTeX yielded a high-quality dictionary in our initial experiments, a challenge emerged in mapping the keys of the generated LaTeX dictionary to their rendered

[1] `https://github.com/wtsnjp/MioGatto/`

```
<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 3.1: LaTeX converted to HTML Format of "A Logic of Expertise" (Singleton, 2021)

instances in the final annotation. This necessitated an array of complex heuristics to convert the dictionary generated using LaTeX as source (Figure 3.5) to a MioGatto dictionary (Figure 3.2a). Converting complex LaTeX equations to ASCII Math would have alleviated the heuristics issue but was deemed infeasible due to ASCII Math's limitations. This also led us to rule out directly ingesting XML; not only was it prohibitive in token count, but LLMs also lacked training on XML, leading them to generate nonsensical outputs. Consequently, we opted for a pre-processing step to retrieve the identifiers in a format that obviated the need for intricate heuristics.

LaTeXML[2] (Ginev et al., 2011) served as the tool of choice for this pre-processing step. The rationale behind this conversion is clear: HTML stands as the source view for further processing with MioGatto and formula grounding. Quite conveniently, LaTeXML automatically identifies mathematical symbols and embeds them in an `<mi>` tag, making the output machine-readable. Figure 3.3 shows the command used in this process. We subsequently transform the HTML into our variant of ASCII Math. While this format conveys less information than LaTeX, the robust capabilities of LLMs compensate for this limitation, yielding comparable results. After the HTML generation, MioGatto's pre-processing tool runs to create a template for the dictionary and annotations as described in Figure 3.4.

[2] `https://math.nist.gov/~BMiller/LaTeXML/`

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

(a) mcdict.json

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

(b) anno.json

Figure 3.2: Dictionary and Annotation Files of MioGatto

| Encoding | Formula | Tokens |
|---|---|---|
| LaTeX | $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ | 24 |
| ASCII Math | $x = (-b \pm \sqrt{b^2 - 4ac}) / (2a)$ | 23 |
| XML | *too complicated to show* | 387 |

(a) Quadratic Equation

| Encoding | Formula | Tokens |
|---|---|---|
| LaTeX | $\oint_C \vec{B} \circ \mathrm{d}\vec{l} = \mu_0 \left( I_{\mathrm{enc}} + \varepsilon_0 \frac{\mathrm{d}}{\mathrm{d}t} \int_S \vec{E} \circ \hat{n} \, \mathrm{d}a \right)$ | 84 |
| ASCII Math | $oint_C(B.dl) = mu_0 * (I_{enc} + eps_0 * d/dt * int_S(E.n_{hat})da)$ | 42 |
| XML | *too complicated to show* | 929 |

(b) Ampere's Circuit Law

Table 3.1: Token usages of different types of encoding

```
latexmlc --preload=[nobibtex,ids,mathlexemes,localrawstyles]latexml.sty
        --format=html5 --pmml --cmml --mathtex --nodefaultresources
        --dest=<output HTML file> <input TeX file>
```

Figure 3.3: LaTeX Source Pre-Process to HTML using LaTeXML [2]

```
python -m tools.preprocess <HTML file>
```

Figure 3.4: HTML pre-process to generate dictionary and annotation template files for MioGatto

```
{
    "$\\equiv$": "Logical equivalence operator",
    "$\\phi$": "A formula in the language $\\cL$",
    "$\\cL$": "Language of expertise and soundness",
    "$\\prop$": "Countable set of propositional variables",
    "$\\univ$": "Universal modality",
    "$\\orr$": "Disjunction operator",
    "$\\subseteq$": "Subset or equal to",
    "$\\in$": "Element of a set",
    "$\\cap$": "Intersection of two sets",
    "$\\subseteq$": "Subset or equal to",
    "$\\emptyset$": "Empty set",
    "$\\sat$": "Satisfaction relation",
    "$\\neg$": "Negation operator",
    "$\\and$": "Conjunction operator",
    "$\\|\\phi\\|_M$": "Set of states where $\\phi$ is true in model $M$",
    "$\\forall$": "Universal quantifier"
}
```

Figure 3.5: Dictionary generated by GPT using LaTeX as Source

## 3.2 Dictionary Generation

Transforming raw HTML data into a comprehensive dictionary required several different tactics. Initial forays included POS tagging. It seemed convenient as, in most academic texts, identifier definitions are placed either before or after the identifier's introduction.

For instance, within the excerpt from the paper "A Logic of Expertise" by Singleton (2021) as shown in Figure 3.6, "Prop" is introduced and immediately followed by its definition "*A countable set of propositional variables*".

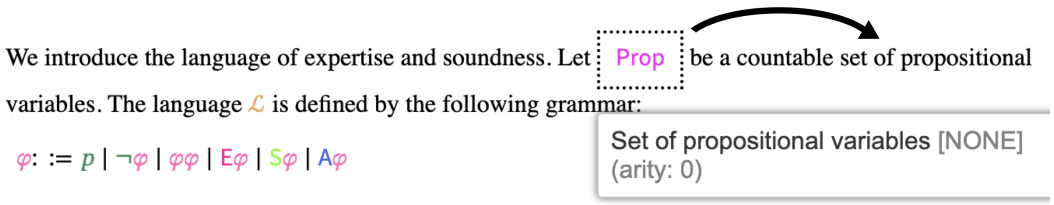



Figure 3.6: Screenshot of MioGatto showing the paper "A Logic of Expertise" with description to the right of the identifier

Similarly in Figure 3.7, '$\mathcal{L}$' is described as "*language*" immediately prior to its mention.

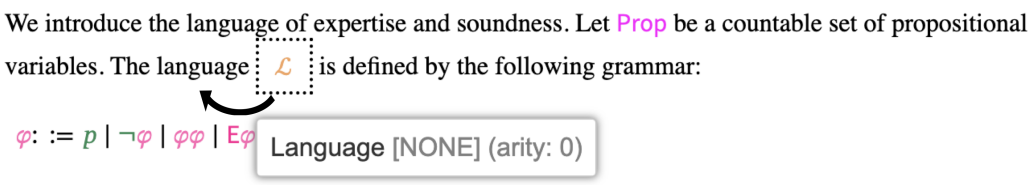



Figure 3.7: Screenshot of MioGatto showing the paper "A Logic of Expertise" with description to the left of the identifier

While seemingly effective for cases like these, this pattern only sometimes holds. In several instances across academic works, the definition does not directly proceed or follow the identifier, making POS tagging less fruitful. Take, for example, this snippet from the same paper mentioned earlier, as shown in Figure 3.8. Here, our mathematical understanding identifies "$X$" as a set, which is not readily inferred by POS tagging or formal grammar alone.

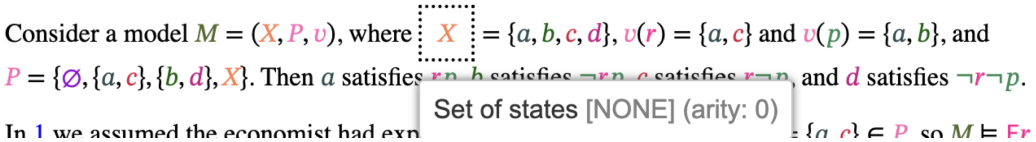



Figure 3.8: Screenshot of MioGatto showing the paper "A Logic of Expertise" without a possible POS Tagging

This complexity led to exploring the capabilities of Large Language Models (LLMs) like GPT-3.5 from OpenAI. While LLMs were generally designed as chat models and not specifically for mathematical text or dictionary generation, preliminary trials proved promising (Figure 3.5). Upon feeding a paragraph of text to the model, it returned a well-formatted dictionary (Figure 3.12) that was highly usable for annotation through OpenAI's API. However, an obstacle emerged that the model's context window is limited. Most of the papers we tested were at least 20K-40K Tokens long, and the context window of the LLMs needs to be more significant to accommodate this.

| Model | Context Window (Tokens) | Chunks Size (Tokens) |
|---|---|---|
| GPT-3.5-turbo | 4096 | 1750 |
| GPT-3.5-16k-turbo | 16384 | 2000 |
| GPT-4 | 8192 | 4000 |

Table 3.2: Token counts for different models

To circumscribe this issue, each paper was divided into chunks, approximately 50% the size of the respective model's context window. The context window varies for

different NLP models as well, and the chosen size of chunks is shown in Table 3.2: This adjustment accounted for the tokens generated by the API and made allowances for the length of our prompts. The token count was carefully selected for GPT-3.5-16k-turbo because the model tended to generalise the description of the identifiers when there were many occurrences with larger chunk sizes.

After dividing the paper into various chunks, they are inputs to the OpenAI API iteratively to generate a dictionary associated explicitly with each chunk. The chunks were carefully constructed so they did not fragment paragraphs and thus maintained the integrity of any crucial contextual information. Furthermore, to mitigate context loss when transitioning from one chunk to another, the generated dictionary was looped back into the prompt, which ensured the LLM maintained awareness of the other possible definitions of the identifiers. The system prompt was meticulously designed to provide the best possible results after experimentation, illustrated in Figure 3.9. After the system prompt, an example of the desired dictionary format was presented, as represented in Figure 3.10. Considering this approach involves neither zero-shot learning nor one-shot/few-shot learning, it is referred to as 'half-shot learning' by me.

```
{'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 3.9: System Prompt for Dictionary Generation

The prompt containing the chunks is then forwarded, as depicted in Figure 3.11. The prior generated dictionary is attached as supplementary context to help contend with the token limit content window. However, if the prompt is excessively long, the prior dictionary is omitted, and this scenario is the only occurrence of context loss from one iteration to another. The chunk is submitted, and the response received (Figure 3.12) is parsed and incorporated into the master dictionary. This procedure is repetitively enacted until all the chunks have passed through the LLM and a comprehensive master dictionary is generated. This essential JSON dictionary (Figure 3.12) then gets converted into a MioGatto-compatible dictionary (Figure 3.2a) using basic mapping techniques.

## 3.3 Association of Each Occurrence

To annotate instances of identifiers with appropriate descriptions, we again employed LLMs to select suitable annotations. We designed specific prompts to optimise the

```
{
  "role": "system",
  "name": "example_user",
  "content": "A relational model is a triple <|M|>′=(<|X|>,<|R|>,<|v|>), where
    <|X|> is a set of states, <|R|><|⊆|><|X|><|×|><|X|> is a binary relation on
    <|X|>, and <|v|>:<| P r o p |>→2<|X|> is a valuation. Given a relational
    model <|M|>′, the satisfaction relation between points <|x<|∈<|X<| and
    formulas <|φ<|∈<|ℒ<|<| K A <| is defined inductively by <|M|>′,<|x|>⊨<| K |>
    <|φ|>⇔ for all <|y|>∈<|X|>,<|x|><|R|><|y|> implies <|M|>′,<|y|>⊨<|φ|
    ><|M|>′,<|x|>⊨<| A |><|φ|><||>⇔ for all <|y|>∈<|X|>,<|M|>′,<|y|>⊨<|φ|>"
}
{
  "role": "system",
  "name": "example_assistant",
  "content": "identifiers = {
    'M`': ['Relational model', 'Expertise Model'],
    'X': 'Set of states',
    'R': 'Binary relation on X',
    'v': 'Valuation',
    ' P r o p ': 'Set of propositions',
    'X': 'A set of state',
    'M'': 'Relational model',
    'x': 'Point in X',
    'φ': 'Formula in  K A ',
    'ℒ_{ K A }': 'Set of formulas',
    ' K ': 'Modal operator K',
    ' A ': 'Modal operator A',
    'y': 'Point in X',
    '⊨': 'Satisfaction relation',
    '⇔': 'If and only if operator',
    '∈': 'Element of a set',
    '⊆': 'Subset of a set',
    '×': 'Cartesian product operator',
    '→': 'Function or implication operator',
    'for all': 'Universal quantifier'}"
}
```

Figure 3.10: Half Shot Learning Example Demonstrated in the Prompt for Dictionary Generation

```
{'role': 'system',
'content': 'Given is already a pre existing dictionary.
Your job is to extend this dictionary. Do not remove
any pre existing definitions from this dictionary. \n'
+ dictionary[index] + '.
If there is nothing to mention, reply with an empty dictionary'},
{'role': 'user', 'content': 'Generate a Python dictionary for the
following text:'
+ chunk +
'Only consider the mathematical identifiers inside "<||>"
for the dictionary.
Do not consider any other identifier other than those marked.
Consider all the identifiers individually, and do not skip
any identifier, mention all the identifiers inside
"<||>" in your dictionary. Do not include the angle
brackets in your dictionary."}
```

Figure 3.11: Main Prompt for Dictionary Generation

LLM's performance. Figure 3.13 shows the system prompt used for this purpose. We set the temperature to 0 to ensure consistency and no hallucination for the annotations.

To enable annotation by the LLM, we slightly modified the generated dictionary to include additional information. The LLM receives the identifier to annotate, a dictionary of potential annotations (including possible affixes), and the context (Figure 3.14b). The context consists of approximately 75 tokens to the left and 25 to the right of the identifier. For GPT-4, we used 40 tokens to the left and 10 to the right to reduce computational costs without sacrificing quality.

Based on this information, the LLM selects the most suitable annotation. The whole context for the annotation is presented in Figure 3.14. This process is repeated for all identifiers. If an identifier has already been annotated, its description serves as context for subsequent identifiers within the same context window (See Figure 3.14 where in context the definition of E is known due to the previous iteration where that identifier as annotated). This process is advantageous for long paragraphs of identifiers but can also lead to cascading errors if an identifier is misannotated. Special consideration is given to identifiers whose affixes match those in the dictionary. If an identifier has only one possible description, it is automatically selected, reducing the computational load on the LLM.

```
{
    "P": [
        "Expertise set"
        "Set of properties P1, P2, P3",
        "Mapping to RP"
        "Set of all unions of equivalence classes"
    ],
    "M": [
        "Expertise model",
        "Model"
        "S5 model",
        "Non-augmented model obtained from N' by dropping the RA'
         component"
    ],
    "p": "Proposition",
    "r": [
        "Economic recovery proposition",
        "Report proposition"
    ],
    "s", "Source"
    "L": "Language of expertise and soundness"
}
```

Figure 3.12: Response from GPT using the Prompt for Dictionary Generation

```
{
    "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 3.13: System Prompt for Annotation

```
{
    "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

```
match_variable = "S"
possible_affixes = "^"
definitions = [{'index': 0,
                'identifier': 'S',
                'description': 'Soundness operator'},
                {'index': 1,
                'identifier': 'S^',
                'description': 'Dual operator of S'}]
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 3.14: Main Prompt for Annotation with context
```

## 3.4 Utilising Open Source LLMs

We experimented with Open Source LLMs upon successfully leveraging GPT models for automating formula annotations. It is essential to recognise that OpenAI's models, including GPT-3.5 with 175B parameters and the rumoured GPT-4 with over 1.8T parameters, outstrip most open-source variants, which generally max out at around 70B parameters. However, GPT models are designed for general-purpose tasks, whereas our focus is primarily instructional. We hypothesised that instructive models could offer comparable performance.

Initial tests with falcon-40b-instruct [3] (Almazrouei et al., 2023; Penedo et al., 2023; Xu et al., 2023), a high-ranking model on the Hugging Face leaderboard [4] (Jain, 2022), were unsuccessful due to its limited context window. Moreover, many LLMs struggled to generate a well formatted JSON suitable to our needs. After evaluating multiple alternatives, we selected superhot models (S. Chen et al., 2023). These models offer extended context windows compatible with our over 1,000 token prompts. We also chose quantised models for improved efficiency while retaining high performance [5]. Specifically, we used vicuna-33b (Zheng et al., 2023) [6], and StableBeluga2 (Mahan et al., n.d.; Touvron et al., 2023; Mukherjee et al., 2023) [7], a fine-tuned model of LLaMa and LLaMa-2 respectively. StableBeluga2 was the top-ranked model on the Hugging Face leaderboard at the time of selection.

We reduced the chunk sizes to 750 tokens without sacrificing quality to adapt to the slower token generation rate. Although the prompts remained identical, we modified their structure from JSON to plain text to suit the transformer models, as illustrated in Figure 3.15.

These open-source LLMs occasionally produced repetitive or incomplete JSONs, necessitating an extension of our dictionary generation approach to handle such irregularities. The transformer settings were: `temperature=0.5, max_new_tokens=512, repetition_penalty=1.05`. All other settings remained consistent with previous configurations.

## 3.5 Setup for the Experiments

Experiments were conducted on a diverse selection of 40 academic papers, chosen by Asakura, Miyao, and Aizawa (2022), using both OpenAI's LLMs and our selected open-source models. Due to the stochastic nature of LLMs, on rare occasions, multiple runs were performed to obtain reliable results. The open-source models were computa-

[3] `https://huggingface.co/tiiuae/falcon-40b-instruct`
[4] `https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard`
[5] `https://medium.com/@developer.yasir.pk/quantized-large-language-model-e80bdcb81a52`
[6] `https://huggingface.co/TheBloke/Vicuna-33B-1-3-SuperHOT-8K-GPTQ`
[7] `https://huggingface.co/TheBloke/StableBeluga2-70B-GPTQ`

```
### SYSTEM: <system message>
### USER: <example user message>
### ASSISTANT: <example assistant output>
### USER: <actual user message>
### ASSISTANT:
```

Figure 3.15: System Prompt of Open Source LLMs for Formula Grounding

tionally intensive despite being quantised, requiring up to 80GB of VRAM. We used cloud GPUs for their affordability and ease of setup. Precisely, experiments involving open-source LLMs were executed on runpod.io with the following configurations:

- Vicuna-33b: 1x NVidia L40 (48GB VRAM), 250GB RAM, 32vCPU at $0.69/h
- StableBeluga2: 1x NVidia A100 SXM (80GB VRAM), 251GB RAM, 16vCPU at $1.84/h

For the OpenAI models, we conducted experiments locally on a MacBook Pro (M1 Pro) due to their lower computational requirements.

## 3.6 Evaluation Metrics

To evaluate the different models' performance rigorously, we employ two primary metrics: 1) CoNLL Score (Pradhan et al., 2012); and 2) Semantic Accuracy.

### 3.6.1 CoNLL Score

The CoNLL score serves as a standard quantitative measure for evaluating the quality of coreference resolution. We calculated this metric using the human-annotated papers generated by Asakura, Miyao, and Aizawa (2022) as the ground truth using CorefUD Scorer [8]. Traditional CoNLL scoring focuses solely on the quality of the coreference clusters—i.e., how well the model groups referring expressions together. However, it does not account for the semantic accuracy of the annotation behind these coreferent items, leading to potential issues in interpretability. For instance, Figure 3.16 illustrates that correctly identifying "Bob" and "he" as part of the same coreference cluster would result in a high CoNLL score. However, if the annotation erroneously labels them as "Alice," semantic integrity is lost, necessitating an additional metric.

[8] https://github.com/ufal/corefud-scorer
[9] https://speakerdeck.com/wtsnjp/lrec2022?slide=4


Figure 3.16: Example of Coreference Clustering (Asakura, Miyao, and Aizawa, 2022)[9]

### 3.6.2 Semantic Accuracy

Determining semantic accuracy presents several challenges. As there is no author-provided ground truth for the papers, establishing the "correctness" of an annotation becomes complex. Moreover, the subtleties in possible annotations—illustrated in Figure 3.17—make automated semantic evaluation difficult. For instance, the identifier $M$ could be annotated as either an `Expertise Model` or the `Expertise counterpart of M'`, and traditional NLP similarity metrics like cosine distance are ill-suited for this evaluation. Therefore, we resorted to manual reviews of the annotations, categorising them as "correct" or "incorrect." Given the time-consuming nature of this method, we limited our review to a representative subset of 6 of the 40 papers initially selected.

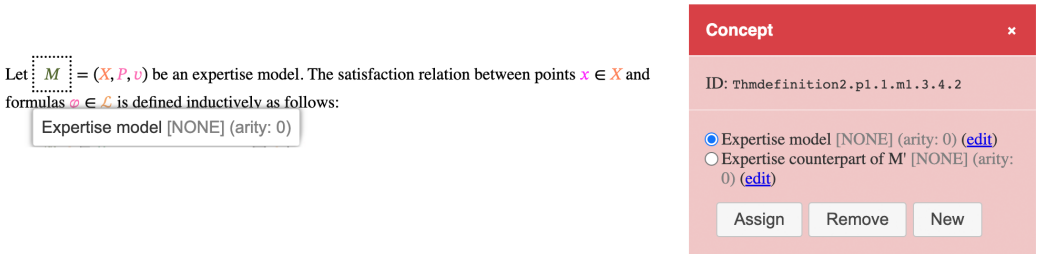



Figure 3.17: Challenges in Determining Semantic Correctness

# 4 Results

The outcomes of the annotation process using GPT and LLMs were compelling. GPT annotations exhibited superior quality and precision in explaining the identifiers. This chapter presents a comprehensive analysis of the results, including CoNLL scores, semantic accuracy scores, annotation costs, annotation duration, the extent of paper annotation by the LLMs, and the variance in scores due to the stochastic nature of LLMs.

## 4.1 GPT

We first scrutinised the annotations generated by GPT. For this analysis, we used all 40 papers from Asakura, Miyao, and Aizawa, 2022 as the ground truth and then employed GPT to generate dictionaries and annotations for all these papers.

### 4.1.1 CoNLL Score

The CoNLL score is a metric used to quantify the quality of coreference clusters. We obtained the weighted average of the CoNLL score across all the papers, with the weighting factored against the total number of annotations in the paper. Higher scores indicate better performance. As shown in Table 4.1, GPT-4 outperformed the other models, while GPT-3.5 exhibited the lowest score.

| Model | Score | Weighted Score |
|---|---|---|
| gpt (average) | 79.31 | 76.89 |
| gpt-3.5-turbo | 78.51 | 75.93 |
| gpt-3.5-turbo-16k | 79.28 | 76.67 |
| gpt-4 | 80.15 | 78.08 |

Table 4.1: Weighted average of CoNLL Scores.

### 4.1.2 Coverage of Annotation

Another critical aspect of this research was to examine the extent of the paper that the LLMs could successfully annotate. We represent this coverage in Table 4.2. GPT-4 demonstrated the highest coverage, while GPT-3.5 had the least.

| Model | Coverage |
|---|---|
| gpt (average) | 90.57% |
| gpt-3.5-turbo | 90.64% |
| gpt-3.5-turbo-16k | 88.21% |
| gpt-4 | 92.87% |

Table 4.2: coverage of the annotation of the papers achieved by GPT

### 4.1.3 Semantic Accuracy

Semantic accuracy provides a measure of the correctness of the annotations. We weighted it against the coverage to represent the total. Higher scores are preferable. As shown in Table 4.3, GPT-4 once again outperformed all the models by a significant margin.

| Model | Score | Weighted Score |
|---|---|---|
| gpt (average) | 88.19% | 79.87% |
| gpt-3.5-turbo | 84.69% | 76.76% |
| gpt-3.5-turbo-16k | 84.17% | 74.25% |
| gpt-4 | 95.70% | 88.88% |

Table 4.3: Weighted average of semantic accuracy

### 4.1.4 Variance of Data

Given the inherent stochastic nature of these models, some variation can be expected across multiple iterations. To account for this, we conducted multiple runs of the experiment on the reference paper ArXiV ID: 2107.10832 (Singleton, 2021) and tabulated the resultant variance. These outcomes are displayed in Table 4.4.

| Model | Best | Worst | Mean | Median | Std. Deviation |
|---|---|---|---|---|---|
| gpt (average) | 86.75 | 83.21 | 84.58 | 84.18 | 1.55 |
| gpt-3.5-turbo | 82.83 | 80.00 | 81.28 | 81.15 | 1.17 |
| gpt-3.5-turbo-16k | 88.38 | 83.47 | 85.29 | 84.65 | 2.16 |
| gpt-4 | 89.05 | 86.16 | 87.18 | 86.75 | 1.32 |

Table 4.4: Different CoNLL scores across four different runs

### 4.1.5 Running Time and Costs

The cost and time efficiency of the employed models are critical to the feasibility of our automated approach. Tables 4.5 and 4.6 respectively present the average time required

and average cost incurred in annotating mathematical identifiers across GPT versions. The pricing is listed on the OpenAI website[1]. While GPT-3.5 emerged as the fastest and most time-efficient model, the GPT-4 version manifested as the most costly to operate.

| Model | Dictionary Generation Time | Annotation Time | Total Time |
|---|---|---|---|
| gpt (average) | 03:49 | 03:25 | 07:14 |
| gpt-3.5-turbo | 02:00 | 02:47 | 04:48 |
| gpt-3.5-turbo-16k | 08:07 | 02:45 | 10:52 |
| gpt-4 | 01:19 | 04:43 | 06:03 |

Table 4.5: Average time taken by each model (mm:ss)

| Model | Cost | Cost / 1M Tokens |
|---|---|---|
| gpt (average) | 1.80 | 9.415 |
| gpt-3.5-turbo | 0.30 | 1.525 |
| gpt-3.5-turbo-16k | 0.52 | 2.299 |
| gpt-4 | 4.59 | 30.449 |

Table 4.6: Average cost of annotation by each model in USD

## 4.2 Open Source LLMs

Upon successfully leveraging GPT to automate formula grounding, we proceeded to automate the process using Open Source Models. We applied the same metrics to see how they compete against GPT. The results were again impressive. Due to open-source LLMs' relatively slow speed and high running costs, we selected a subset of 7 papers out of the original 40 by Asakura, Miyao, and Aizawa (2022). We carefully chose the papers to cover a range of attributes, including high/low CoNLL scores, high/low semantic accuracy, and varied lengths.

### 4.2.1 CoNLL Score

We applied the same metrics of CoNLL score to the seven selected papers annotated by Open Source LLMs. We compared them against those annotated by GPT, with the human-annotated versions serving as the reference/ground truth. The values can be seen in Table 4.7. Among the open-source models, StableBeluga2 matched the performance of GPT models, indicating its commendable competence in creating high-quality coreference clusters.

[1] `https://openai.com/pricing`

| Model | Score | Weighted Score |
|---|---|---|
| vicuna-33b | 72.44 | 77.45 |
| gpt (average) | 88.75 | 87.55 |
| gpt-3.5-turbo | 88.21 | 86.22 |
| gpt-3.5-turbo-16k | 90.12 | 89.40 |
| gpt-4 | 87.92 | 87.03 |
| StableBeluga2 | 84.55 | 81.63 |

Table 4.7: Weighted average of CoNLL Scores.

### 4.2.2 Coverage of Annotation

Open Source models also demonstrated impressive performance in their ability to annotate the papers. The values can be seen in Table 4.8. StableBeluga2 had near-identical performance to the GPT Models.

| Model | Coverage |
|---|---|
| vicuna-33b | 66.18% |
| gpt (average) | 91.97% |
| gpt-3.5-turbo | 88.93% |
| gpt-3.5-turbo-16k | 90.62% |
| gpt-4 | 96.35% |
| StableBeluga2 | 93.17% |

Table 4.8: coverage of the annotation of the papers achieved by GPT

### 4.2.3 Semantic Accuracy

For semantic accuracy, we excluded one paper (ArXiV ID: 2107.10832 (Singleton, 2021)) due to its length. It was not feasible to manually evaluate it semantically due to its extensive length. We weighted the results against the coverage again to represent the total. As detailed in Table 4.9, the semblance between the performance of StableBeluga2 and GPT models reinforces this open-source model's potential in accurately understanding and reflecting the context of scientific papers.

### 4.2.4 Variance of Data

It was impossible to calculate the data variance due to the high costs of running the Open Source Models and budget constraints. However, preliminary testing during the model selection phase indicated that Open Source Models were considerably more stable than GPT.

| Model | Score | Weighted Score |
|---|---|---|
| vicuna-33b | 61.58% | 40.75% |
| gpt (average) | 88.19% | 81.11% |
| gpt-3.5-turbo | 84.69% | 75.31% |
| gpt-3.5-turbo-16k | 84.17% | 76.27% |
| gpt-4 | 95.70% | 92.21% |
| StableBeluga2 | 90.91% | 84.70% |

Table 4.9: Weighted average of semantic accuracy

### 4.2.5 Running Time and Costs

Given the distinctive operational requirements of open-source models, the computation of time and cost efficiencies differ from those of GPT models. Unlike GPT models, the cost for open-source models revolves around GPU runtime on the servers of runpod.io and not token usage. Table 4.10 and 4.11 represent these essential aspects. The pricing of GPT is listed on the OpenAI website[2]. The cost of running vicuna-33b was 0.69USD/h, and StableBeluga2 was 1.84USD/h.

| Model | Dictionary Generation Time | Annotation Time | Total Time |
|---|---|---|---|
| vicuna-33b | 04:16 | 07:16 | 12:33 |
| gpt (average) | 01:54 | 01:53 | 03:48 |
| gpt-3.5-turbo | 01:04 | 01:21 | 02:25 |
| gpt-3.5-turbo-16k | 04:02 | 02:18 | 06:21 |
| gpt-4 | 00:38 | 02:01 | 02:39 |
| StableBeluga2 | 05:17 | 09:43 | 20:20 |

Table 4.10: Average time taken by each model (mm:ss)

| Model | Cost |
|---|---|
| vicuna-33b | 0.14 |
| gpt (average) | 1.80 |
| gpt-3.5-turbo | 0.15 |
| gpt-3.5-turbo-16k | 0.22 |
| gpt-4 | 2.06 |
| vicuna-33b | 0.62 |

Table 4.11: Average cost of annotation by each model in USD

[2] `https://openai.com/pricing`

# 5 Analysis

# 6 Conclusion

# 7 Future Works

# Abbreviations

**LLMs** Large Language Models

**NLP** Natural Language Processing

**POS** Part-of-Speech

**CoNLL** Computational Natural Language Learning

**MioGatto** Math Identifier-oriented Grounding Annotation Tool

**MLP** Mathematical Language Processing

**STEM** Science, Technology, Engineering and Mathematics

# List of Figures

# List of Tables

# Bibliography


Alexeeva, M., R. Sharp, M. A. Valenzuela-Escárcega, J. Kadowaki, A. Pyarelal, and C. Morrison (2020). "MathAlign: Linking formula identifiers to their contextual natural language descriptions." In: *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pp. 2204–2212.

Almazrouei, E., H. Alobeidli, A. Alshamsi, A. Cappelli, R. Cojocaru, M. Debbah, E. Goffinet, D. Heslow, J. Launay, Q. Malartic, B. Noune, B. Pannier, and G. Penedo (2023). "Falcon-40B: an open large language model with state-of-the-art performance." In.

Asakura, T., A. Greiner-Petter, A. Aizawa, and Y. Miyao (2020). "Towards grounding of formulae." In: *Proceedings of the First Workshop on Scholarly Document Processing*, pp. 138–147.

Asakura, T., Y. Miyao, and A. Aizawa (2022). "Building Dataset for Grounding of Formulae—Annotating Coreference Relations Among Math Identifiers." In: *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pp. 4851–4858.

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

Chen, S., S. Wong, L. Chen, and Y. Tian (2023). "Extending context window of large language models via positional interpolation." In: *arXiv preprint arXiv:2306.15595*.

Ding, B., C. Qin, L. Liu, L. Bing, S. Joty, and B. Li (2022). "Is gpt-3 a good data annotator?" In: *arXiv preprint arXiv:2212.10450*.

Ginev, D., H. Stamerjohanns, B. R. Miller, and M. 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*. Springer, pp. 292–294.

Grigore, M., M. Wolska, and M. Kohlhase (2009). "Towards context-based disambiguation of mathematical expressions." In: *The joint conference of ASCM*.

He, X., Z. Lin, Y. Gong, A. Jin, H. Zhang, C. Lin, J. Jiao, S. M. Yiu, N. Duan, W. Chen, et al. (2023). "Annollm: Making large language models to be better crowdsourced annotators." In: *arXiv preprint arXiv:2303.16854*.

Jain, S. M. (2022). "Hugging face." In: *Introduction to Transformers for NLP: With the Hugging Face Library and Models to Solve Problems*. Springer, pp. 51–67.

Mahan, D., R. Carlow, L. Castricato, N. Cooper, and C. Laforte (n.d.). *Stable Beluga models*. URL: %5Bhttps://huggingface.co/stabilityai/StableBeluga2%5D(https://huggingface.co/stabilityai/StableBeluga2).

Meadows, J. and A. Freitas (2022). "A survey in mathematical language processing." In: *arXiv preprint arXiv:2205.15231*.

Mukherjee, S., A. Mitra, G. Jawahar, S. Agarwal, H. Palangi, and A. Awadallah (2023). "Orca: Progressive learning from complex explanation traces of gpt-4." In: *arXiv preprint arXiv:2306.02707*.

OpenAI (2023a). *GPT-3.5*. URL: https://platform.openai.com/docs/models (visited on 09/15/2023).

– (2023b). *GPT-4 Technical Report*. eprint: arXiv:2303.08774.

Pagael, R. and M. Schubotz (2014). "Mathematical language processing project." In: *arXiv preprint arXiv:1407.0167*.

Penedo, G., Q. Malartic, D. Hesslow, R. Cojocaru, A. Cappelli, H. Alobeidli, B. Pannier, E. Almazrouei, and J. Launay (2023). "The RefinedWeb dataset for Falcon LLM: outperforming curated corpora with web data, and web data only." In: *arXiv preprint arXiv:2306.01116*. arXiv: 2306.01116. URL: https://arxiv.org/abs/2306.01116.

Peng, S., K. Yuan, L. Gao, and Z. Tang (2021). "Mathbert: A pre-trained model for mathematical formula understanding." In: *arXiv preprint arXiv:2105.00377*.

Pradhan, S., A. Moschitti, N. Xue, O. Uryupina, and Y. Zhang (2012). "CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes." In: *Joint conference on EMNLP and CoNLL-shared task*, pp. 1–40.

Schubotz, M., L. Krämer, N. Meuschke, F. Hamborg, and B. 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*. Springer, pp. 82–94.

Singleton, J. (2021). "A Logic of Expertise." In: *arXiv preprint arXiv:2107.10832*.

Touvron, H., L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. (2023). "Llama 2: Open foundation and fine-tuned chat models." In: *arXiv preprint arXiv:2307.09288*.

Xu, C., D. Guo, N. Duan, and J. McAuley (2023). "Baize: An Open-Source Chat Model with Parameter-Efficient Tuning on Self-Chat Data." In: *arXiv preprint arXiv:2304.01196*.

Zheng, L., W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing, et al. (2023). "Judging LLM-as-a-judge with MT-Bench and Chatbot Arena." In: *arXiv preprint arXiv:2306.05685*.