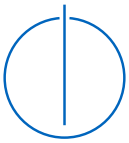
SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

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

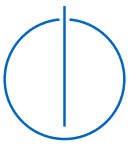
TECHNISCHE UNIVERSITÄT MÜNCHEN

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

First and foremost, I would like to express my deepest gratitude to my advisors, Takuto Asakura and Miriam Anschütz, for guiding me through this thesis's challenging yet rewarding journey. Your belief in me made all the difference.

Special thanks are also to Prof. Dr. Yusuke Miyao and Prof. Dr. Georg Groh for their supervision and for providing the facilities and environment I needed to focus on my work.

I extend my heartfelt thanks to Shoko Yamagishi, the secretary at Miyao Lab, for her incredible efficiency in navigating the bureaucracy of being an exchange student in Japan. You made my life easier in so many ways.

Words cannot adequately express my thanks to Aleksandra, who has been my pillar of strength, emotionally and physically. Your love and support have been unfaltering, even during the most stressful times.

My gratitude also goes out to Christoph, whose invaluable advice and insights made the daunting task during my entire studies seem manageable.

A warm thank you to Valentin, who was there when I needed him at the most critical moments. Your timely assistance made a world of difference.

I am also indebted to my family members Arun, Pallavi, and Parul, whose constant support and encouragement have been a steady force behind my efforts.

I would also like to extend my thanks to all the members of Panikecke[1] for their camaraderie and emotional support, as well as GPT and Rune Sætre (NTNU) for their meticulous proof-reading of my thesis.

Lastly, a big thank you to all my other friends Dhia, Philipp, Leonard, Daniel, and Mrinal, who have offered their support in many different ways. Every bit helped, and I am grateful for it.

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

Abstract

This thesis presents an innovative approach to automating the annotation of mathematical identifiers in scientific papers, a traditionally laborious and costly task. The study leverages Large Language Models (LLMs), specifically GPT-3.5 and GPT-4 from OpenAI (2023a), and open-source alternative LLMs. They are used to generate a dictionary of identifiers and their possible descriptions and to associate each instance of an identifier with its appropriate definition based on the given context.

We evaluated the results using two primary metrics: the CoNLL score, which quantifies the quality of coreference clusters, and semantic accuracy, which measures the correctness of the annotations. The study reveals that GPT-4 delivers the highest performance in both metrics with a CoNLL Score of 78.08 and semantic correctness of 88.88%, although it is also the most expensive. In contrast, GPT-3.5 emerges as the most cost-effective and fastest model. The open-source model StableBeluga2 also shows significant potential, delivering performance almost on par with the GPT models.

The findings of this study demonstrate the potential of LLMs in automating the annotation of mathematical identifiers, thereby streamlining the process of coreference resolution. Our work lays the ground for future research in this domain, with potential avenues including improving semantic accuracy, expanding model selection, improving annotation coverage, reducing annotation costs, and developing more sophisticated measures of semantic accuracy.

Contents

1. Introduction

Mathematical formulae, 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 formulae presents a considerable challenge. This challenge 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.

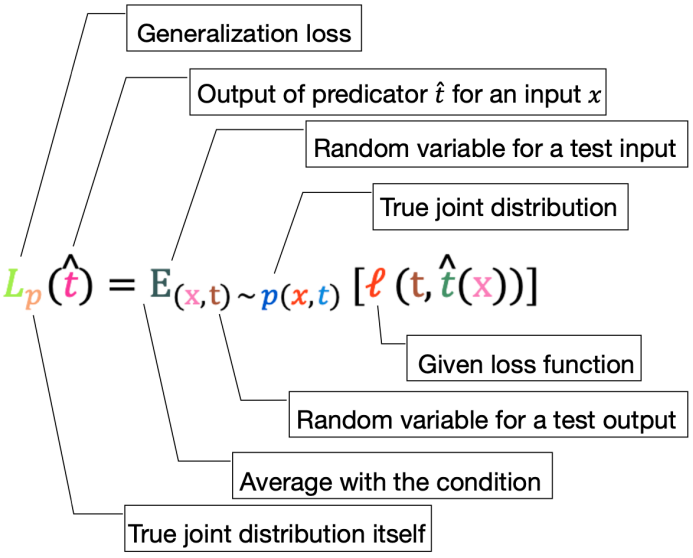


Figure 1.1.: Challenges in disambiguating identifiers within mathematical formulae. The same variable can serve multiple roles, each underpinned by a distinct definition, leading to ambiguity. [1]

MioGatto [2] (Asakura et al., 2021), a Math Identifier-oriented Grounding Annotation Tool, was conceived to address this issue. However, this tool relies on the manual annotation of papers. Moreover, manually 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 formulae (Asakura 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. **Pre-processing:** This phase entails the transformation of the LaTeX source into a machine-readable HTML/XML format using LaTeXML [3] (Ginev et al., 2011).
2. **Dictionary Generation:** Leveraging Large Language Models (LLMs), short text clustering techniques will be employed to construct a comprehensive dictionary of all mathematical identifiers as keys with their respective possible (multiple) descriptions as the corresponding values. Each paper has its dictionary. The exact structure of this dictionary is explained further in Section 3.2.
3. **Association of Each Occurrence:** This step will involve associating every instance of a mathematical identifier with its corresponding definition based on the dictionary generated. Here we are drawing inspiration from MathAlign (Alexeeva et al., 2020).

1.1. Motivation and Problem

Annotations of mathematical identifiers are critical for comprehending a formula or the text passage. 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 saying "time is money" holds here. The extended hours spent on manual annotation translate to increased financial costs.

[2] https://github.com/wtsnjp/MioGatto/
[3] https://math.nist.gov/~BMiller/LaTeXML/

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 Part-of-Speech (POS) tagging or the establishment of formal grammar, tend to oversimplify the problem (shown in Section 3.2). These methods often provide a generalised 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 used 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?

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

1.3. Contributions

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

1. **Extensive Annotations:** Tested a range of LLMs, including GPT-3.5-turbo (OpenAI, 2023a), GPT-3.5-turbo-16k, GPT-4 (OpenAI, 2023b), vicuna-33b[4](Zheng et al., 2023), and StableBeluga2[5](Mahan et al., 2023; Touvron et al., 2023; Mukherjee et al., 2023), to annotate a curated set of 40 scientific papers. This extensive annotation process serves as a comprehensive evaluation of the capabilities of these models.

2. **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.

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

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 coreference quality.

These contributions 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 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.

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

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 compared.

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

4. **Results:** Here, the empirical outcomes of the research are presented. The chapter summarises the scores achieved by each LLM, the methodologies used to derive these scores, and the rationale behind selecting the 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. It also introduces the novel formula developed during the research to estimate the CoNLL Score for a given paper.

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 Work:** The final chapter outlines potential avenues for further exploration and improvement, such as model combinations or the incorporation of emerging techniques.

This structure provides a logical flow, guiding readers from the foundational concepts to the conclusions, and offers a holistic understanding of the research journey.

2. Related Work

The field of Mathematical Language Processing (MLP) has been progressively evolving, focusing on the complexities of understanding, annotating, and disambiguating mathematical text. This chapter critically reviews key papers that have significantly influenced this domain, thereby contextualising our current study.

2.1. Understanding Mathematical Text and Formulae

The early work by Grigore et al. (2009) is a foundational study in mathematical language processing. Grigore emphasised the importance of linguistic context in deciphering mathematical formulae, primarily focusing on symbolic expressions within mathematical narratives. While this work laid the groundwork, it did not delve into automation or using machine-learning models for annotation.

Building upon this, the MLP project by Pagael et al. (2014) introduced a novel approach employing POS tag-based distances to estimate the probabilities of identifier-definition relationships. This approach provided a more nuanced understanding of mathematical identifiers than traditional pattern-matching techniques. However, it still relied on manual intervention for accurate results. In contrast, our approach automates this process using LLMs, eliminating the need for manual intervention and significantly reducing the time and cost of annotation.

Most recently, Meadows et al. (2022) supported using transformer models like GPT for formula retrieval. Meadows emphasised the role of informal mathematical text in quantitative reasoning, allowing more automated approaches. Unlike our study, Meadows' research focuses not on annotation automation but on formula retrieval.

2.2. Foundational Frameworks and Automation

The pioneering work by Asakura et al. (2020) introduced MioGatto, a cutting-edge annotation tool that anchors or grounds mathematical formulae. This tool significantly enhanced the comprehension of STEM manuscripts by resolving the ambiguity in mathematical identifiers. However, MioGatto relied on manual annotation, which is time-consuming and resource-intensive. Our research builds upon this work by automating the annotation process in MioGatto, making it more efficient and accessible.

Ding et al. (2022) and Schubotz et al. (2017) collectively highlight the potential of automation in data annotation. Ding specifically focuses on the role of GPT-3 in

reducing annotation overheads but does not delve into the mathematical language processing aspect of it. Schubotz, on the other hand, sets the stage for automated extraction frameworks for mathematical identifier definitions but does not employ LLMs for this purpose, nor concerns the task of annotations.

Another noteworthy contribution to Mathematical Language Processing is the work by Alexeeva et al. (2020). This study proposes a rule-based approach for extracting mathematical identifiers and linking them to their in-text descriptions. The domain-agnostic approach operates directly on PDFs, requiring only the location of the formula of interest. While their rule-based method offers a structured way to link identifiers to their descriptions, it does not employ LLMs or focus on coreference resolution among mathematical identifiers, a central theme in our research.

Their work also presents a novel evaluation dataset and the tool used to create it. Unlike our approach, which leverages the capabilities of LLMs for automating the annotation process, their method relies on rule-based algorithms. This makes their approach less flexible in handling the complexities and ambiguities often found in mathematical text. Moreover, their study does not delve into their method's cost or time effectiveness aspects that we consider in our research to evaluate the practical applicability of automated annotation methods. While both works aim to automate the process of annotating mathematical identifiers, our study extends the scope by employing LLMs and focusing on coreference resolution and semantic accuracy.

2.3. The Role of LLMs 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. MathBERT is fine-tuned for decoding mathematical formulae and emphasises the importance of context. However, it is not designed for annotating mathematical identifiers, our primary focus.

The research on GPT-3.5 evaluates its efficacy as a robust annotator and questions its potential to replace traditional crowdsourced methods. While this work aligns closely with our study, it does not use metrics like CoNLL or semantic accuracy, which are central to our research, nor does it tie it to mathematical text.

2.4. Summary

The body of work reviewed here offers a comprehensive landscape of the challenges and innovations in MLP. These cornerstone studies provide invaluable insights and reveal gaps our research aims to fill. As the field evolves, our study builds upon these foundational works, introducing automation and comprehensive evaluation metrics to advance the domain of mathematical identifier annotation.

3. Methodology

This chapter describes the methodology employed in this research, encompassing three core stages: 1) pre-processing of mathematical identifiers, 2) dictionary construction, and 3) association of individual IDs to description (annotation) instances. The process uses LaTeXML utilities and explores the capabilities of LLMs to automate the annotation of mathematical identifiers. Here, we present the transitions from source materials to machine-readable formats and from dictionary formulation to final annotation. The entire code is available at `https://github.com/babyygemperor/grounding-of-formulae`.

Before delving into the specifics, it is crucial to understand the structure of MioGatto[1] (Asakura et al., 2021), the tool we aim to automate. MioGatto contains three core files:

1. `source.html`—a pre-processed HTML file suitable for web rendering of the given paper, illustrated in Figure 3.1
2. `mcdict.json`—a JSON file containing a list of dictionary 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 three files corresponds to the three first 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`. The three core files are interlinked by IDs generated during pre-processing. The structure of the JSON files is depicted in Figure 3.2.

3.1. Pre-processing

Initially, we considered parsing the LaTeX code directly via GPT-based LLMs, owing to their frequent training on LaTeX documents. LaTeX has a more decent semantic layer to mathematical identifiers than ASCII Math and offers efficiency in token usage (see Definition 1) and complexity compared to MathML. 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.

[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.: HTML format of "A Logic of Expertise" (Singleton, 2021) obtained after transforming the LaTeX source using LaTeXML. This machine-readable format serves as the basis for dictionary generation and annotation in MioGatto.

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 instances in the final annotation. This necessitated an array of complex heuristics to convert the dictionary generated using LaTeX as source (Figure 3.3) to a MioGatto dictionary (Figure 3.2a). Converting complex LaTeX equations to ASCII Math would have alleviated the heuristics issue, but it was deemed infeasible due to ASCII Math's limitations of expressing rich content. This also led us to rule out directly ingesting XML because of the prohibitive token count and because LLMs 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 removed 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. We subsequently transform the HTML into our variant of ASCII Math with the following command.

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

[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.: JSON file structure in MioGatto. The dictionary file contains a list of mathematical identifiers (keys) and their possible descriptions (values). The annotation file holds the index of the chosen description for each identifier.

```
{
    "$\\equiv$": "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 3.3.: Dictionary generated by GPT-3.5 using LaTeX as the source demonstrating LLMs ability to generate a comprehensive and well-structured dictionary of mathematical identifiers and their descriptions.

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

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

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	Shown in Appendix A.2.1	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	Shown in Appendix A.2.2	929

(b) Ampere's Circuit Law

Table 3.1.: Token usages of three different types of encoding (LaTeX, ASCII Math, and XML).

3.2. Dictionary Generation

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

For instance, within the excerpt from the paper "A Logic of Expertise" by Singleton (2021) as shown in Figure 3.4, "Prop" is introduced and immediately followed by its definition "*A countable set of propositional variables*".

Similarly in Figure 3.5, '$\mathcal{L}$' is described as "*language*" immediately prior to its mention.

While seemingly effective for cases like these, this pattern only works for some cases. In several instances across academic works, the definition does not directly proceed or follow the identifier, making POS tagging less fruitful. Take, for example, the snippet shown in Figure 3.6 from the same paper mentioned earlier. Here, our mathematical understanding identifies "X" as a set, which is not readily inferred by POS tagging or formal grammar alone.

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

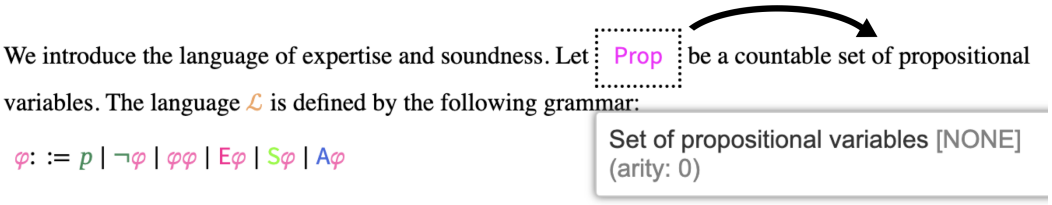


Figure 3.4.: Screenshot of MioGatto showing the paper "A Logic of Expertise" with description to the right of the identifier Prop.

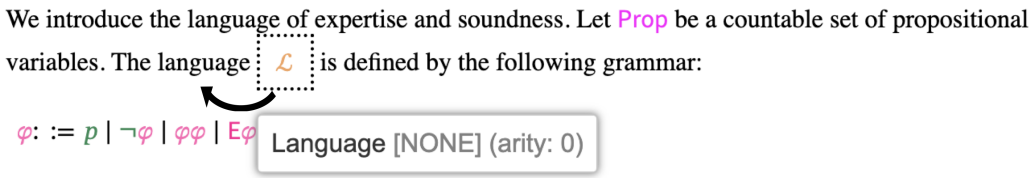


Figure 3.5.: Screenshot of MioGatto showing the paper "A Logic of Expertise" with description to the left of the identifier L.

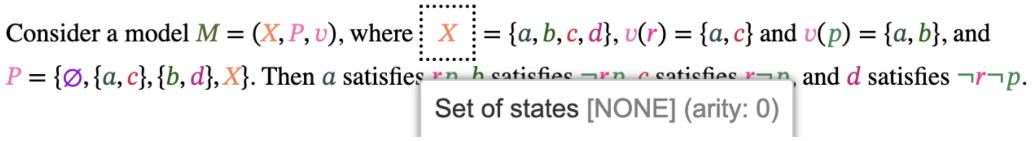


Figure 3.6.: Screenshot of MioGatto showing the paper "A Logic of Expertise" without a possible POS Tagging.

```
{
    "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 3.7.: Response JSON from GPT using the prompt for dictionary generation containing the keys as the mathematical identifiers and the values as their possible respective definitions.

not specifically for mathematical text or dictionary generation, preliminary trials proved promising. Upon feeding a paragraph of text to the model through OpenAI's API, it returned a well-formatted dictionary (Figure 3.7) that was highly usable for annotation. However, an obstacle emerged that the model's context window is limited. Most of the papers we tested were (at least) 20-40 thousand tokens long, and the context window of the LLMs is not big enough to accommodate this (see Table 3.2).

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 deal with the overflow issue, each paper was divided into smaller chunks, approximately half the size of the respective model's context window. The context window varies for different NLP models as well, and the chosen sizes of chunks are 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 reduced for GPT-3.5-16k-turbo because the model tended to generalise the description of the identifiers when there were many occurrences in larger chunk sizes.

After dividing the paper into smaller overlapping chunks, they are used as 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 (see Definition 2) (Figure 3.8) was meticulously designed, through experimentation, to provide the best possible results. After the system prompt, an example of the desired dictionary format was presented, as shown in Figure 3.9. Considering this approach involves neither zero-shot learning nor one-shot/few-shot learning, we call it 'half-shot learning'.

The prompt containing the chunks (Figure 3.10) is then forwarded to the next stage. The previously generated dictionary is attached as supplementary context to help deal with the token limit context window. However, if the prompt is excessively long, the previous 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 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 3.7) then gets converted into a MioGatto-compatible dictionary (Figure 3.2a) using basic mapping techniques.

```
{'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 3.8.: System prompt for dictionary generation instructing the LLM to convert long paragraphs into a Python dictionary, emphasising the need to identify and classify each mathematical identifier.

3.3. Association of ID to Description Occurrence

To annotate instances of identifiers with appropriate descriptions, we again employed LLMs to select suitable annotations. We designed specific prompts to optimise the LLM's performance. Figure 3.11 shows the system prompt used for this purpose. We set the temperature to 0 to ensure consistency and to avoid 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 3.12b). 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 noticeably sacrificing quality.

Based on this information, the LLM selects the most suitable annotation. The whole context for the annotation is presented in Figure 3.12. 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 3.12 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.

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 GPT-4 rumoured to have over 1.8T

```
{
  "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 3.9.: An example of the desired dictionary format provided in the prompt for dictionary generation guiding the LLM to produce a dictionary with mathematical identifiers as keys and their possible descriptions as values serving as a template for the LLM, facilitating the generation of a MioGatto-compatible dictionary.

```
{'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
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 3.10.: Main prompt for dictionary generation instructing the LLM to convert the given chunks into a Python dictionary expanding the pre-existing dictionary as extra context.

```
{
    "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 3.11.: System prompt for associating the identifiers to their descriptions instructing the LLM to pick a suitable definition.

```
{
    "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 3.12.: Main prompt for associating the identifiers to their descriptions instructing the LLM to select the suitable index of the definition within the given context.
```

parameters (almost two trillion), outstrip most open-source variants, which generally max out at around 70B (billion) parameters. However, GPT models are designed for general-purpose tasks, whereas our focus is primarily instructional. We hypothesised that "instruct" 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[6] (Zheng et al., 2023), and StableBeluga2[7] (Mahan et al., 2023; Touvron et al., 2023; Mukherjee et al., 2023), 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 noticeably sacrificing quality to adapt to the slower token generation rate and potentially smaller comprehension capabilities. Although the prompts remained identical, we modified their structure from JSON to plain text to suit the transformer models better (see Figure 3.13).

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.

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

Figure 3.13.: System prompt used for dictionary generation with open-source LLMs instructing the LLM to generate a Python dictionary for the given text.

[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

3.5. Setup for the Experiments

Experiments were conducted on 40 academic papers using OpenAI's LLMs and our selected open-source models. The papers were chosen by Asakura et al. (2022). Due to the stochastic nature of LLMs, on rare occasions, multiple runs had to be performed to obtain reliable results. The open-source models were computationally intensive despite being quantised, requiring up to 80GB of Video RAM (VRAM). We used cloud Graphical Processing Units (GPU)s for their affordability and ease of setup. Specifically, 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

3.6. Evaluation Metrics

To rigorously evaluate the different models' performance, 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 et al. (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 3.14 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.

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 3.15—make automated semantic evaluation difficult. For instance, the identifier

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

Figure 3.14.: Example of Coreference Clustering (Asakura et al., 2022)[9]

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, as the cosine distance between two similar annotations can be minimal with drastically different mathematical interpretations. 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.

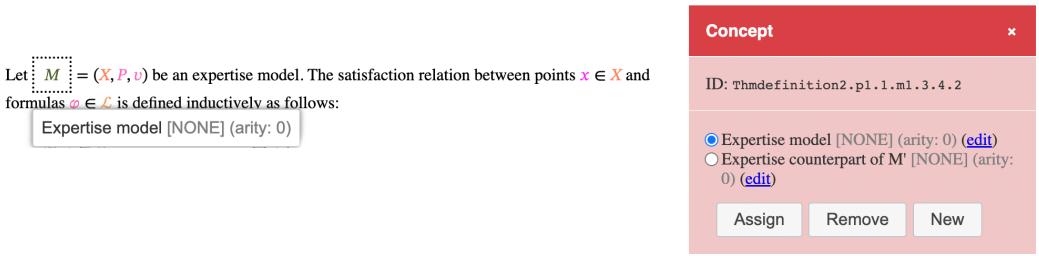


Figure 3.15.: Challenges in determining semantic correctness, where both the possible definitions look very similar but convey different meanings in a given context.

4. Results

The outcomes of the annotation process using GPT and LLMs were compelling. GPT annotations showed quality and precision in explaining the identifiers. This chapter presents 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. All the results are available at https://docs.google.com/spreadsheets/d/1v0t9q5V2j4phjZxXQFIH3b06vl8UBNrFDZlz6ajPAac/edit.

4.1. GPT

We first scrutinised the annotations generated by GPT. For this analysis, we used all 40 papers from Asakura et al., 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. This means that the calculated average score for every paper weighs the same, regardless of the number of annotations in that 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 how much of the paper the LLMs could successfully annotate. We present 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. Because of the extensive difficulty of manually reviewing semantic accuracy, we evaluated six carefully picked papers representing various low/high CoNLL scores and lengths. As shown in Table 4.3, GPT-4 again outperformed all the models by a significant margin. There was one instance where GPT-4 achieved an astonishing 100% semantic accuracy (GPT-3.5: 96.15%).

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[1] (Singleton, 2021) and tabulated the resultant variance. These outcomes are displayed in Table 4.4.

[1] `https://arxiv.org/pdf/2107.10832.pdf`

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[2]. While GPT-3.5 emerged as the fastest and most time-efficient model, the GPT-4 version manifested as the most costly. Table 4.7 describes the relative cost per concept.

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 Input 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 automation by each model in USD

4.2. Open Source LLMs

Upon successfully using GPT to achieve formula grounding, we proceeded to do the same using Open Source Models instead. We applied the same metrics to see how they compete. The results were again impressive. 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

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

Model	Cost / 1000 Annotations (USD)	Time / Annotation (seconds)
gpt (average)	2.49	0.73
gpt-3.5-turbo	**0.40**	**0.46**
gpt-3.5-turbo-16k	0.88	1.25
gpt-4	6.18	0.47

Table 4.7.: Relative cost and time taken

by Asakura et al. (2022). We carefully chose the papers to cover a range of attributes, including high/low CoNLL scores, high/low semantic accuracy, and short/long papers.

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. Table 4.8 shows the respective values. Among the open-source models, StableBeluga2 matched the performance of GPT models, indicating its ability to create high-quality coreference clusters. Vicuna-33b failed to annotate one paper entirely, so it has a score of 0 for that paper.

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 4.8.: 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 4.9. StableBeluga2 had near-identical performance to the GPT Models. Vicuna-33b failed to annotate one paper entirely, so it has a score of 0 for that paper.

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 4.9.: coverage of the annotation of the papers achieved by GPT

4.2.3. Semantic Accuracy

For semantic accuracy, we excluded the paper[3] (Singleton, 2021) due to its length. It was not feasible to manually evaluate it semantically because it was too long. Again, we weighted the results against the coverage to represent the total. As shown in Table 4.10, the resemblance between the performance of StableBeluga2 and the GPT models indicates that open-source models also have the potential to accurately "understand" (reflecting the correct context of) scientific papers.

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 4.10.: Weighted average of semantic accuracy

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 our budget constraints. However, preliminary testing during the model selection phase indicated that Open Source Models produce considerably more stable results than the GPT models.

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

[3] `https://arxiv.org/pdf/2107.10832.pdf`

cost for open-source models revolves around GPU run-time on the servers of runpod.io and not the token usage. Tables 4.11 and 4.12 represent both the time and actual cost aspects. The pricing of GPT is listed on the OpenAI website[4]. 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 4.11.: Average time taken by each model (mm:ss)

Model	Cost	Cost / 1000 Annotations (USD)	Time / Annotation (sec)
vicuna-33b	**0.14**	0.51	2.73
gpt (average)	1.80	2.70	1.06
gpt-3.5-turbo	**0.15**	**0.43**	0.60
gpt-3.5-turbo-16k	0.22	1.01	2.10
gpt-4	2.06	6.67	**0.48**
StableBeluga2	0.62	2.77	5.48

Table 4.12.: Average cost of annotation by each model

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

5. Analysis

This chapter thoroughly explores the results obtained, elucidating the significance of the observed scores and probing into the reasons behind their variation. The analysis examines the CoNLL scores, the semantic accuracy scores, annotation costs, annotation durations, the extent of paper annotation by the different LLMs, and the variance in scores due to the stochastic nature of LLMs. While specific observations, such as the superior performance of models with more parameters, might appear intuitive, the analysis also uncovers less apparent insights.

5.1. GPT

First, the annotations generated by three variations of the GPT model (GPT-3.5, GPT-3.5-16k, and GPT-4) were examined. The ground truth was derived from 40 papers selected by Asakura et al. (2022), and these were utilised as templates for GPT-generated dictionaries and annotations. An emerging pattern in these results showed that GPT-4 is a superior model compared to GPT-3.5-16k and GPT-3.5.

5.1.1. CoNLL Score

The CoNLL score measures the quality of coreference clusters, employing a weighted average to keep the varying/vast number of annotations per paper in check. The CoNLL Scores of all three models are presented in the violin plot in Figure 5.1. GPT-4 is visibly better and more consistent than its counterparts.

5.1.2. Estimation of CoNLL Score

We observe that the CoNLL scores depend upon four primary factors:

1. Topic: Our experiments revealed that papers from specific disciplines performed better — Logic is easier than NLP, with Astronomy and Mathematics following far behind. The underlying reason for this might be the training data, but due to the not-so-open nature of OpenAI, it is impossible to verify this. Moreover, the inherent nature of Language Models struggling with Mathematics is perhaps another reason GPT suffered in Mathematics papers.

2. Model: The hierarchy is clear — GPT4 outperformes GPT-3.5-16k, which in turn surpasses GPT-3.5.

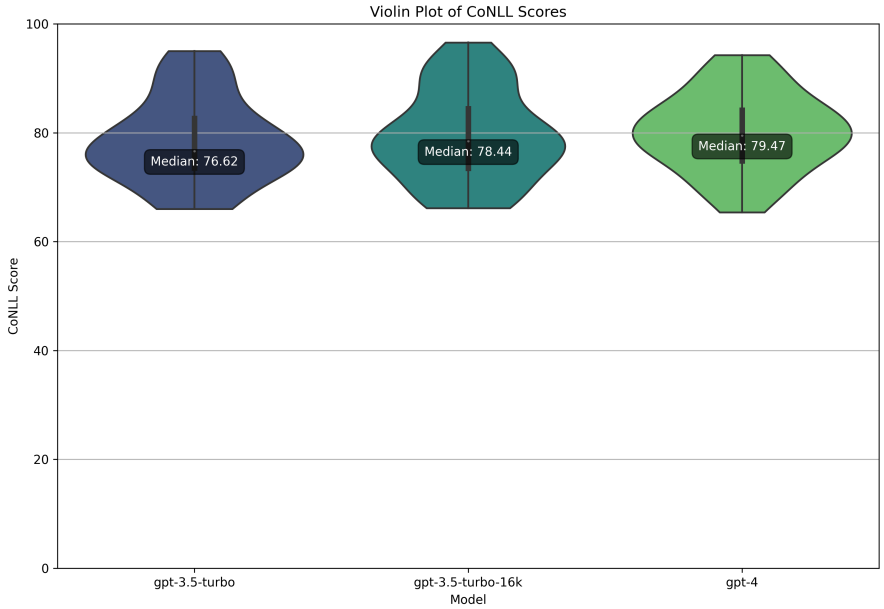


Figure 5.1.: Violin plot illustrating the disparity in CoNLL scores across three GPT models revealing that GPT-4 outperforms its counterparts in quality of coreference clusters, exhibiting higher and more consistent scores.

3. Ambiguity depends on the number of discrete identifiers in a given paper. For example, in the formula $x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$ there are 4 identifiers, x, a, b, and c. The lower the number of identifiers, the better a model typically performs, as fewer distinctions can be made.

4. Obscurity: This is estimated using the interquartile range of a given identifier's occurrences, focusing solely on the middle quartile and disregarding the outliers. From the same example of the quadratic equation of $x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$, x and c are repeated once but a and b are repeated twice. The rationale for focusing on the middle 40-percentile range is twofold. First, due to their limited occurrences, identifiers that appear infrequently (in the bottom 30th percentile) are generally easier for GPT to disambiguate. Second, persistent identifiers (in the top 30th percentile) may challenge annotation. However, our annotation approach results in a cascading effect that minimises its impact on the CoNLL score. Therefore, the identifiers falling within the middle 40 percentile truly contribute to obscurity. This subset is the primary focus when estimating the CoNLL score for annotation accuracy.

The last three variables help us devise a robust mathematical formula that can be used to quickly estimate the actual CoNLL score:

$$\text{conll\_score} \simeq \text{intercept} - \frac{2 \times \text{total\_concepts}}{5} - \frac{\text{occur\_iqr}}{5} \tag{5.1}$$

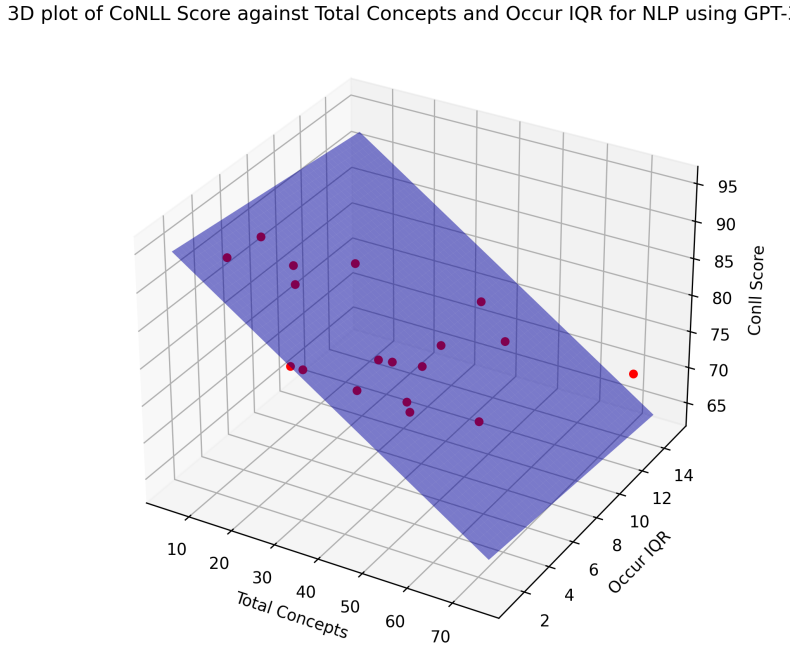
In the equation above, the variables are defined as follows:

- **conll_score** is the CoNLL score estimated (Domain dependent).
- **intercept** is the y-axis intercept, which depends on the model. GPT-3.5 = 93, GPT-3.5-16k = 95, GPT-4 = 97. (Model)
- **total_concepts** is the total number of concepts in the paper (Ambiguity).
- **occur_iqr** represents interquartile, middle 40-percentile (Obscurity).

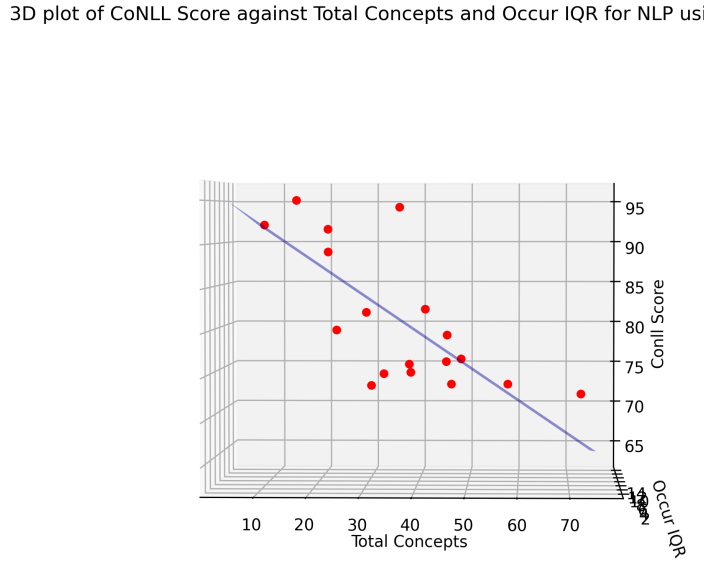
Despite the limited sample size, preliminary results in Table 5.1 demonstrate the potential of this predictive model. The formula's estimation Mean Square Error and R2 Score are shown. Since there were only 18 papers in hand for NLP, getting better results took much work. Figure 5.2 shows the visualisation of our novel formula on a 3D plane.

5.1.3. Coverage of Annotation

The coverage of annotation refers to the proportion of the paper that the LLMs successfully annotated. Consistent with the CoNLL results, GPT-4 consistently outperforms the



(a) Angled View



(b) Side view

Figure 5.2.: 3D visualisation of the CoNLL Score Estimation Formula based on the intercept, the total_concepts, and the occur_iqr, providing a quick and efficient method to predict the actual CoNLL score.

Model	Mean Square Error	R2 Score
GPT-3.5-turbo	35.636	0.136
GPT-3.5-16k-turbo	25.936	0.371
GPT-4	26.386	0.360

Table 5.1.: Mean Square Error and R2 Score of the Estimation Formula

other two GPT models. GPT-3.5-16k, in contrast, had lesser coverage than GPT-3.5 due to the 16k model's instability and propensity for repetition death. Figure 5.3 provides a visual representation of the coverage exhibited by all three models.

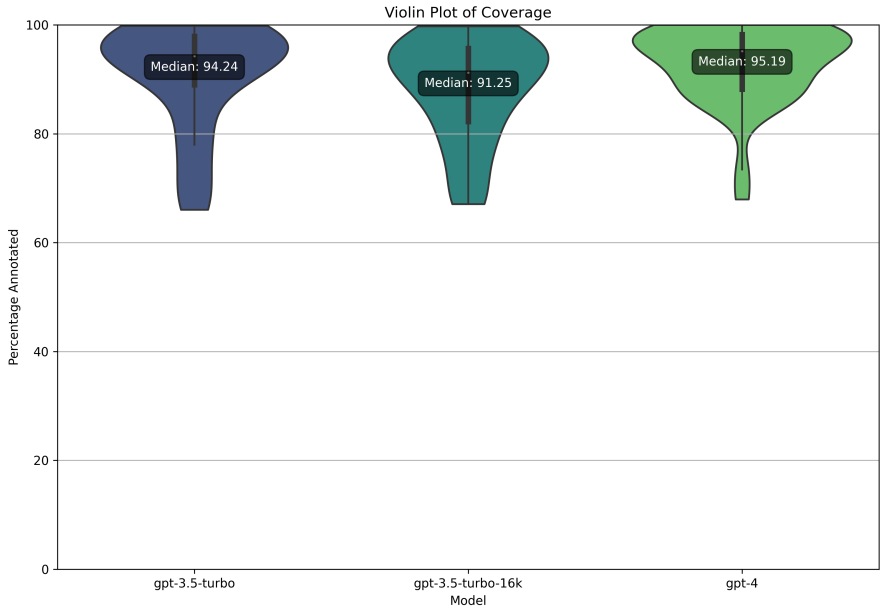


Figure 5.3.: Violin plot depicting the coverage of annotations for each GPT model. GPT-4 represents the highest proportion of the paper annotated successfully and consistently, with GPT-3.5 scoring less.

5.1.4. Semantic Accuracy

Semantic accuracy provides a measure of the correctness of the annotations. We weighted it against the coverage to calculate a per paper-based total score. Because of the extensive difficulty of manually reviewing semantic accuracy, we evaluated six carefully picked papers representing various low/high CoNLL scores and lengths. As shown

in Figure 5.4, GPT-4 consistently outperformed all the other models by a significant margin. There was one paper where GPT-4 achieved a 100% semantic accuracy, but weighing it with coverage brings it down to 98%. GPT-4's worst performance is almost as good as the best performances of other models, making it superior. This comes as no surprise since GPT-4 is one of the largest (and most expensive) LLMs as of writing this thesis (late 2023).

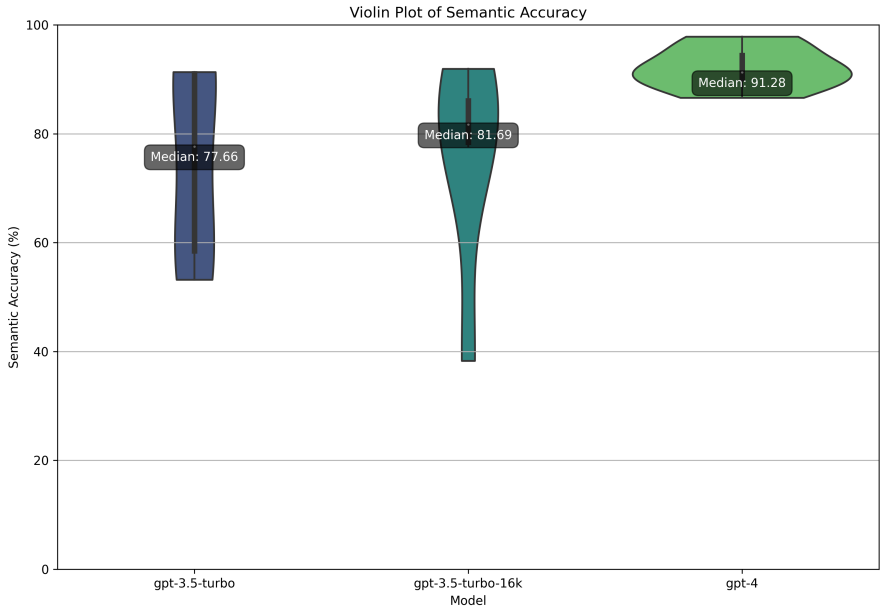


Figure 5.4.: Violin plot showing the weighted semantic accuracy of all GPT Models. GPT-4 significantly outperforms others with even its worst performance matching GPT-3.5 and GPT-3.5-16K's best.

5.1.5. Variance of the Results Data

To account for these models' stochastic nature and measure our experiment's reproducibility and stability, we repeated the same experiment on one reference paper[1] (Singleton, 2021). We ran the experiment four times, evaluating the variance in the CoNLL scores. GPT-3.5 and GPT-4 proved monumentally stable, whereas GPT-3.5-16k, a newer model, still exhibited volatility issues. These outcomes show that GPT-4 consistently generates annotations with a high CoNLL score compared to GPT-3.5 and is displayed in Figure 5.5.

[1] `https://arxiv.org/pdf/2107.10832.pdf`

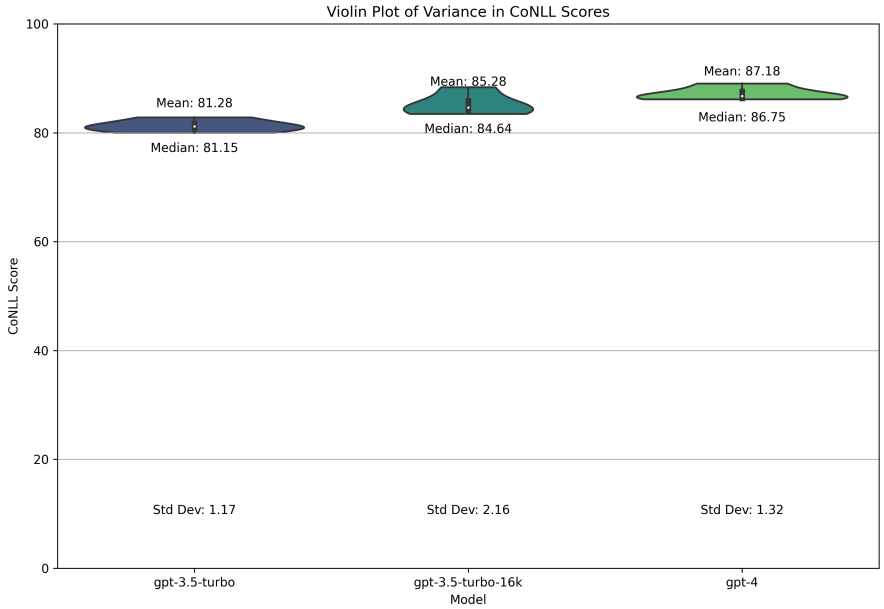


Figure 5.5.: Violin plot capturing the variance in CoNLL scores for the same paper annotated four times highlighting the stability of GPT-3.5 and GPT-4 and the relative volatility of GPT-3.5-16K.

5.1.6. Running Time and Costs

The financial aspects of utilising GPT models are quantified through token usage. A comprehensive visualisation of the average costs and time durations for our experiments is provided in Figure 5.6. It is crucial to note that the length of the paper influences both the cost and the time needed. To offer a more standardised comparison, we present the costs normalised per 1,000 annotations in Figure 5.7.

Another pivotal dimension is the trade-off between cost and time efficiency. While the ideal scenario would be to minimise both, practical constraints often make this challenging. This relationship is further explored in Figure 5.8. GPT-3.5 emerged as the most cost-effective and time-efficient option among the GPT models evaluated. This efficiency is attributable to OpenAI's competitive token pricing and additional optimisations (for ChatGPT). Conversely, GPT-4 incurred the highest expenses due to its elevated token costs. GPT-3.5 and GPT-4 demonstrated remarkable stability, contributing to their lower time expenditures. On the other hand, GPT-3.5-16k exhibited instability, leading to increased running time.

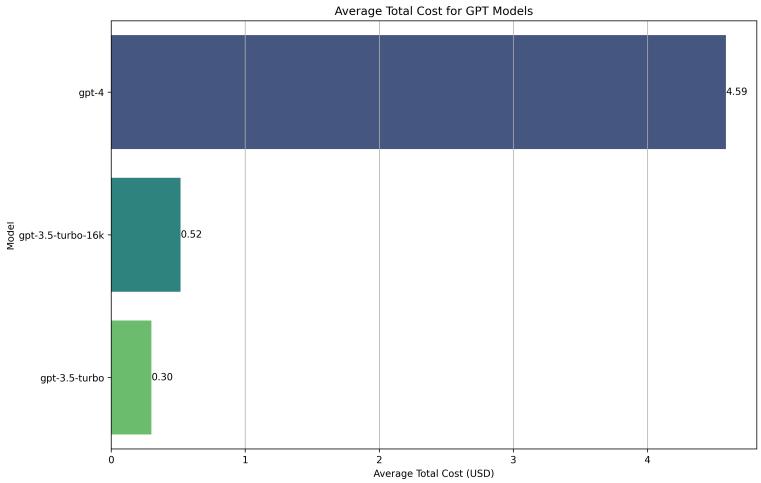
5.2. Open Souce LLMs

Next, we also thoroughly examined the annotations generated by the Open Source Models. This analysis used a smaller subset with 7 of the 40 papers selected by Asakura et al. (2022) as the ground truth. Dictionaries and annotations were generated for these papers. The Open Source models varied drastically in terms of their performance. Among the two models we evaluated, Vicuna-33b lagged noticeably behind the GPT models in performance. This outcome is expected because Vicuna-33b operates on a comparably small 33-billion parameter architecture. However, it is noteworthy that a small-scale model still demonstrates a formula grounding capability.

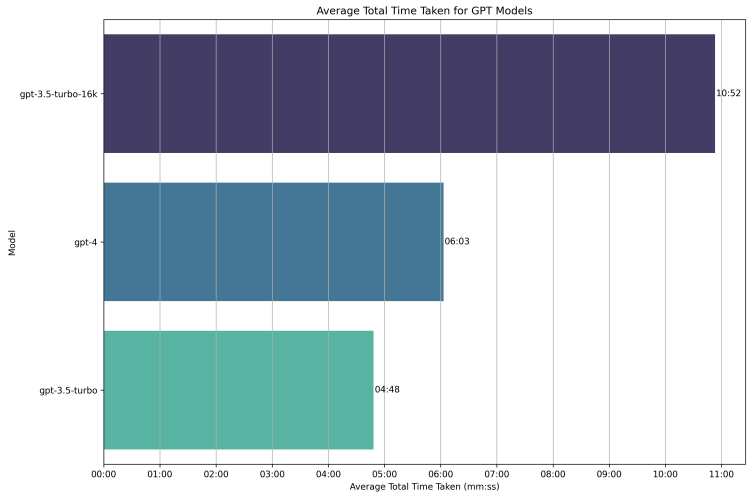
Conversely, StableBeluga2 exhibited outstanding performance, nearly matching that of GPT-4. This is particularly impressive, considering that StableBeluga2 operates on only a 70-billion parameter framework, while GPT-4 is rumoured to have a staggering 1.8 trillion parameters. Moreover, StableBeluga2 consistently outperformed GPT-3.5 across multiple metrics. This superior performance is likely attributable to the specialised nature of StableBeluga2, designed as an "instruct" model, in contrast to GPT models that are general-purpose chat models not explicitly optimised for formula grounding.

5.2.1. CoNLL Score

Vicuna-33b struggled in several cases, even scoring zero in one instance, hinting at its inability to generate any meaningful dictionary in some papers. StableBeluga2, on the other hand, aptly managed to deliver performances that stood almost on par with the GPT models as illustrated in Figure 5.9. Despite this, GPT models maintained a

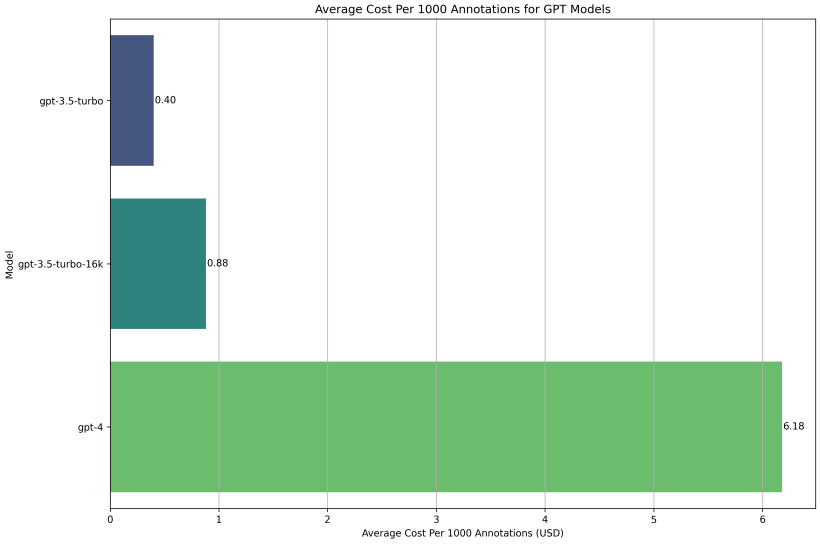


(a) Average Cost of Automation

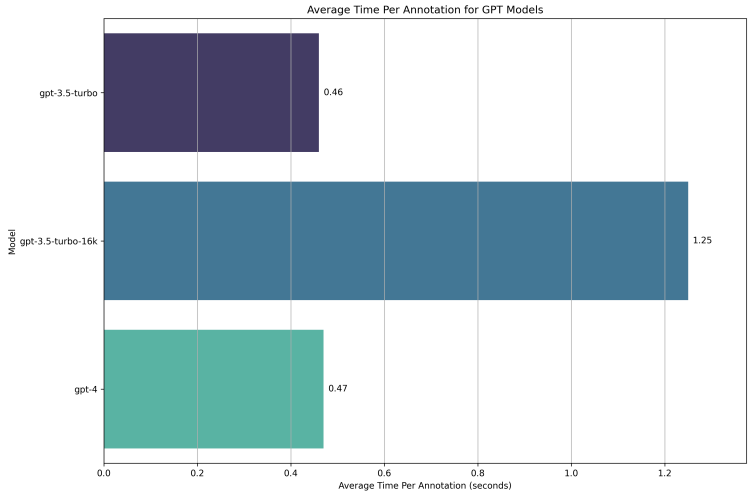


(b) Average Duration of Automation

Figure 5.6.: Visualisation of the average costs and durations for the GPT models annotation underlining how the paper's length influences both the cost and the time needed, offering a crucial glimpse into economical dynamics of the models.



(a) Average Cost per 1000 Annotations



(b) Average Duration per Annotation

Figure 5.7.: Visualisation of the standardised comparison, presenting the costs and durations normalised per 1,000 annotations showcasing the trade-off between cost and efficiency of models.

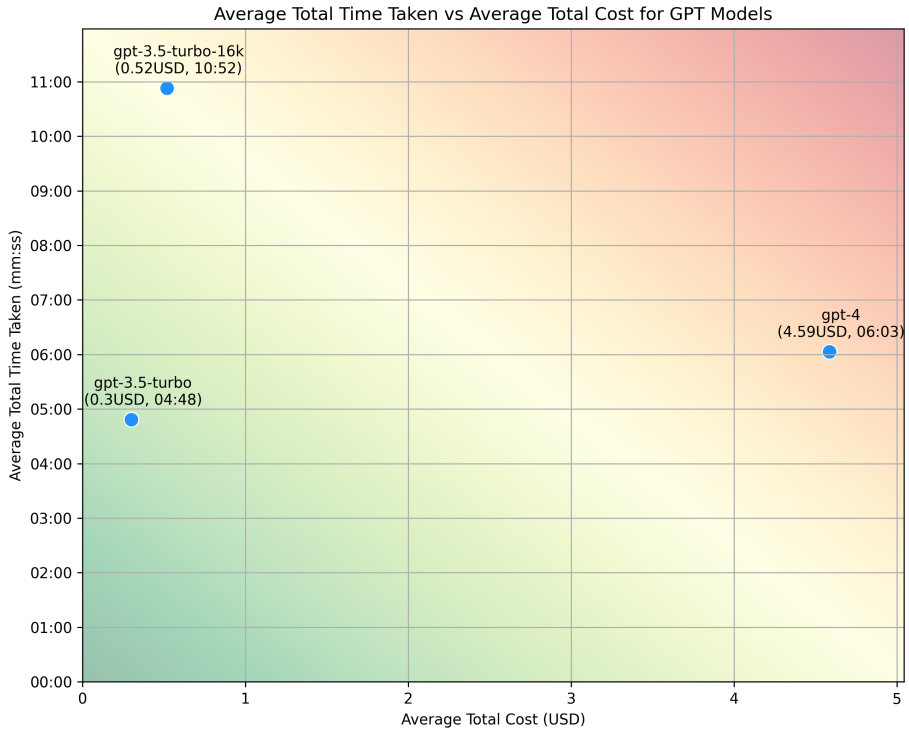


Figure 5.8.: Scatter plot illustrating the relationship between average cost and time taken indicating that while GPT-3.5 is the most cost-effective and time-efficient, GPT-4 incurs the highest expenses due to elevated token costs and GPT-3.5-16K exhibits instability, leading to increased running time.

discernible edge in disambiguation capabilities over their open-source counterparts. This advantage is likely attributable to the extensive training that GPT models undergo.

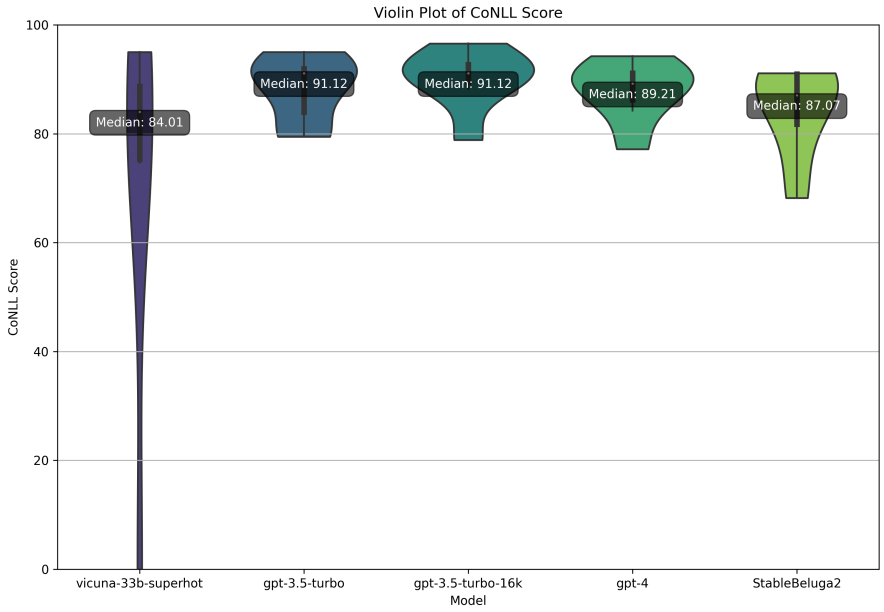


Figure 5.9.: Violin plot of the CoNLL scores for GPT models and open-source models. Vicuna-33b struggles in several instances compared to StableBeluga2, which demonstrates the performance almost comparable to GPT models.

5.2.2. Coverage of Annotation

Once again, vicuna-33b faced challenges in providing complete coverage for one paper, resulting in one score of zero. StableBeluga2, on the other hand, achieved performances comparable to the GPT Models, as can be seen in Figure 5.10. Even disregarding the isolated zero score for Vicuna-33b, its performance generally remains subpar compared to its more advanced counterparts. GPT-4, meanwhile, consistently achieves high coverage for all the papers.

5.2.3. Semantic Accuracy

Semantic accuracy is the cornerstone of our comparative study. Due to the labour-intensive nature of manual evaluation, we selected a subset of six papers for this purpose. StableBeluga2, an Open Source LLM, beats GPT-3.5 entirely but could not surpass GPT-4, which deserves mention due to the unmatched complexity and

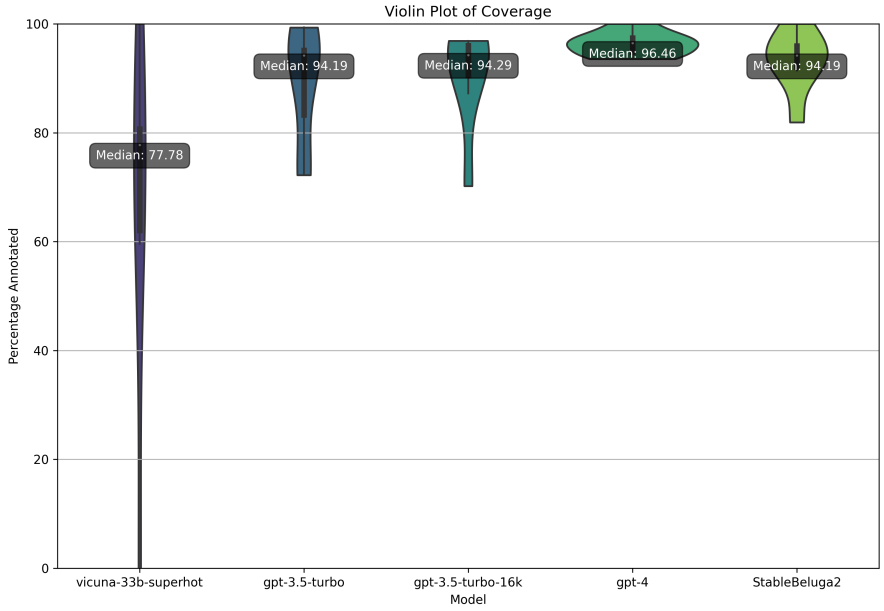


Figure 5.10.: Violin plot of the coverage of seven selected papers annotated by open-source and GPT models. Vicuna-33b has noticeably lower coverage than others, while StableBeluga2 performed almost on par with GPT models.

sophistication of GPT-4's architecture. As shown in Figure 5.11, the comparable performance of StableBeluga2 and the GPT models reinforces this open-source model's potential to accurately understand and reflect the context of scientific papers. Vicuna-33b's performance was notably lacklustre, a limitation likely attributable to its smaller model size. GPT-4 and StableBeluga2 consistently generate annotations of high semantic accuracy for all sorts of papers we tested it against.

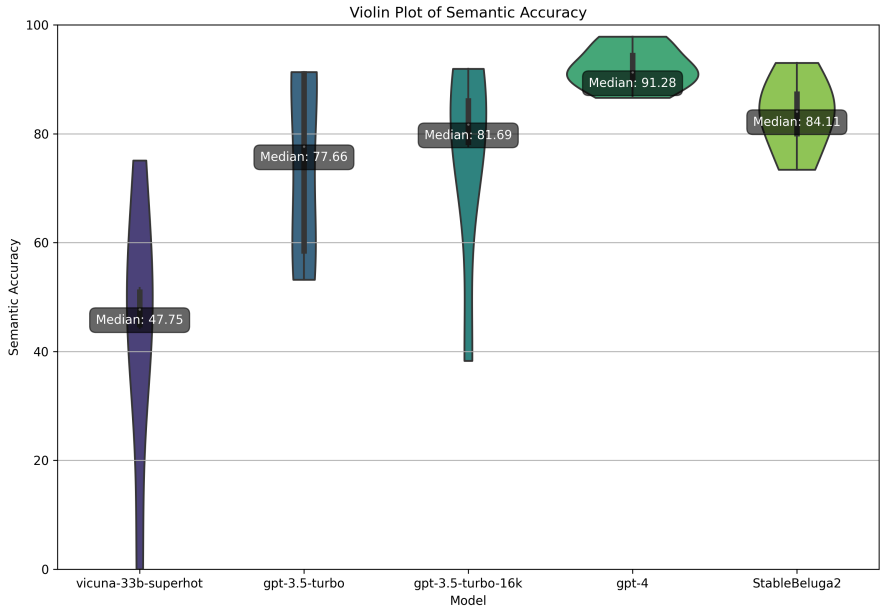


Figure 5.11.: Violin plot of Semantic Accuracy Scores for open-source and GPT models on selected six papers. While StableBeluga2 outperformed GPT-3.5, it fell short of surpassing GPT-4. Vicuna-33b lagged noticeably.

5.2.4. Running Time and Costs

The computation of time and cost differ between open-source and GPT models. Unlike GPT models, the cost for open-source models revolves around GPU run-time cost (in our case, on the servers of runpod.io) and not token usage. The running time is visualised in Figure 5.12. However, since the time taken here depends on the length of the paper, it is essential to compare the costs per annotation. This can be visualised in 5.13. Because of cheaper hardware, the average run time of open-source LLMs was prolonged. On average, they were 5-10x slower. This is especially noticeable for StableBeluga2 as it is a reasonably large LLM. The cost for the experiments can differ

from person to person, based on their machinery. We had to pay for the GPU usage, but if a person owns GPUs, it can be "free of cost". Regardless, we have visualised the cost we paid for annotations in Figure 5.14. It can also very well happen that the cost to use these GPUs is higher than for GPT tokens. GPT-4 is significantly more expensive than all other models. vicuna-33b has a similar operating cost to GPT-3.5 but performs way worse. StableBeluga2, which performs better than GPT-3.5, costs 3x more than GPT-3.5 and 3x less than GPT-4.

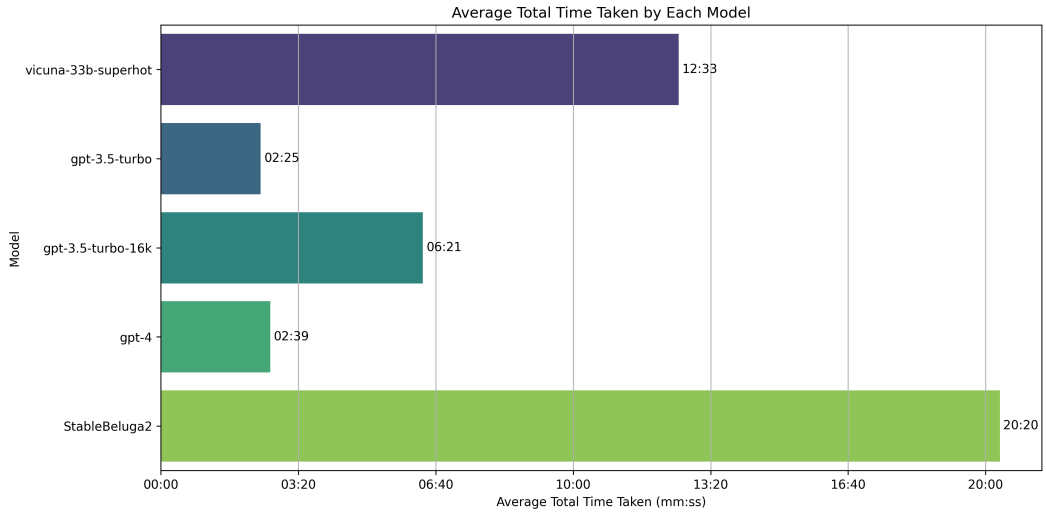


Figure 5.12.: Bar plot of average duration of the annotation process for open-source and GPT models. Due to slow hardware, the average run time of open-source LLMs was worse, especially noticeable for StableBeluga2.

5.3. Evaluating Potential Correlations

Upon completion of our results assessment, it was fundamental to investigate potential correlations inherent in the results. We probed two areas of interest: a potential correlation between the CoNLL Score and Semantic Accuracy and a possible linkage between the CoNLL Score and the paper's publication date. These examinations were crucial to determine whether any part of the original paper was contained within OpenAI's training data.

5.3.1. Connection Between CoNLL Score and Semantic Accuracy

Given that these two metrics reflect distinct facets of the paper, we anticipated that little to no correlation would be discernible. Nevertheless, several statistical measures were employed to thoroughly evaluate this potential relationship: Pearson's Correlation

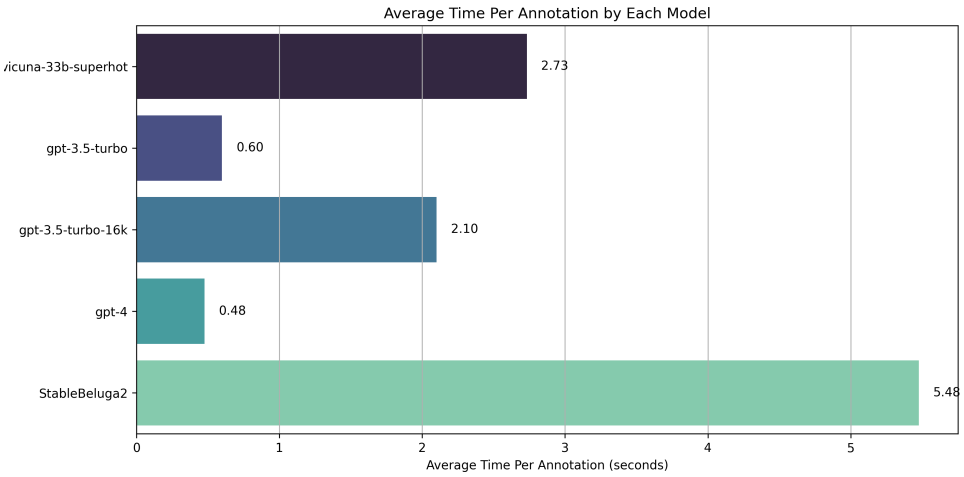


Figure 5.13.: Bar plot of average time cost per annotation for open-source and GPT models. The cost of open-source models depends on availability of GPUs. As per our experiments, costs for GPT-4 were higher due to elevated token costs.

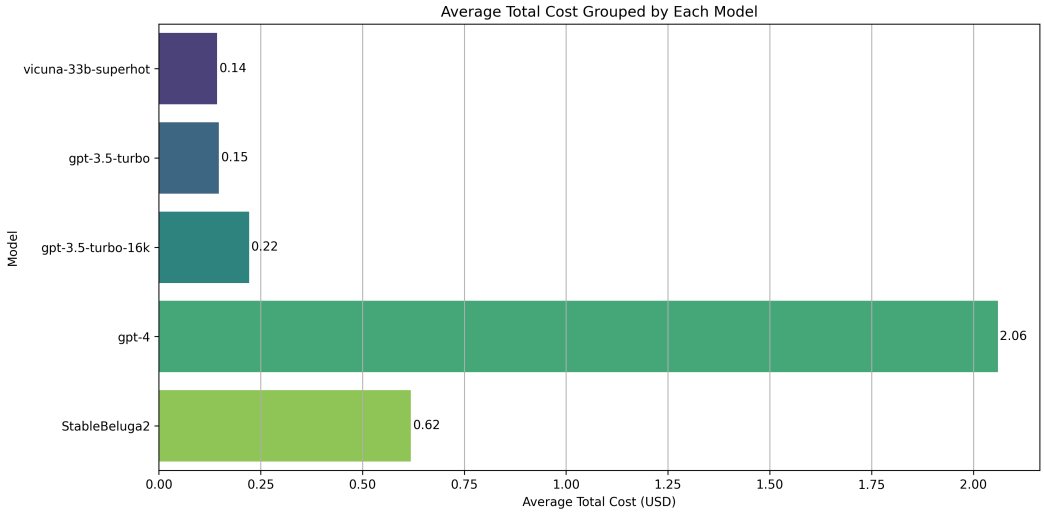


Figure 5.14.: Bar plot of average costs for annotation by all five models. GPT-4 represented the highest costs while remaining models, including open-source ones, cost less on an average.

Coefficient, Spearman's Rank Correlation, and Kendall's Tau. As seen in Table 5.2, the correlation results indicate a trivial association between these metrics, as the correlation constant is virtually zero, supported further by a notable high p-value.

Table 5.2.: Correlation Coefficients and P-values

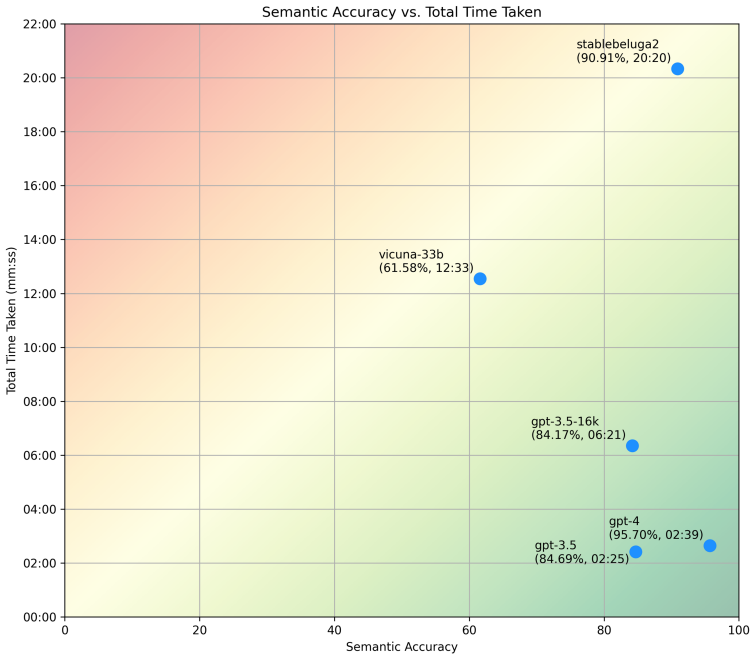
Method	Correlation	P-value
Pearson's Correlation Coefficient	-0.1570	0.5472
Spearman's Rank Correlation	-0.0006	0.9981
Kendall's Tau	-0.0075	0.9670

5.3.2. Influence of Publish Date on CoNLL Score

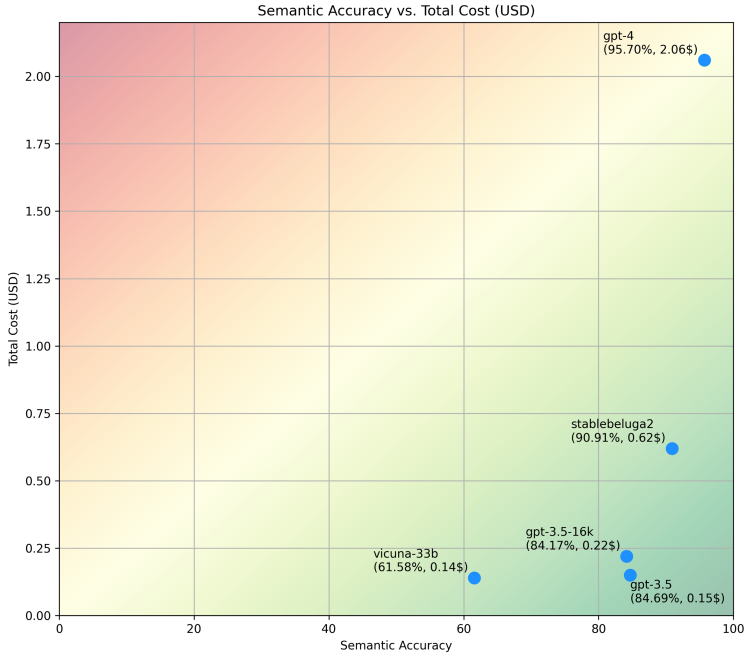
A query was raised regarding the possibility of papers released before September 2021—presumed to be the cut-off date for building OpenAI's GPT Models training data—yielding higher CoNLL Scores than those published afterwards. The difference in average CoNLL scores between papers released before was higher by 1.63 compared to those released after the cut-off date. This might be attributable to the variable nature of LLMs rather than the inclusion in the training set. For post-cut-off papers, the semantic accuracy was also marginally lower—by an average of 5.87%. However, this difference did not significantly suggest being due to training data influence. Furthermore, the use of GPT to generate novel annotations makes it improbably likely for these exact or similar outputs to exist in the training dataset and affect the results. An observation worthy of mention is that the post-2021 papers contained significantly more concepts, which may have influenced the score. Henceforth, it is reasonable to conclude that a paper being part of the training dataset or not does substantially impact the score.

5.4. Overall

In comprehensive terms, GPT-4 emerged as the most impressive model due to its superior performance, albeit at a higher cost. GPT-3.5 was the most cost-effective and fastest model to operate. All GPT models provided commendable annotations for automation purposes, with the Open Source LLM StableBeluga2 marking a significant breakthrough with its "zero-cost" operation and performance that is almost at par with the GPT models. This is particularly noteworthy, considering StableBeluga2 is a 70 billion parameter model, and GPT-4 is rumoured to be a 1.8 trillion parameter model. The "instruct" nature of StableBeluga2, as opposed to the general-purpose chat model design of GPT, likely contributed to its performance in formula grounding. Figure 5.15 visualises the performance-to-cost ratio for all five models, which helps to choose the models for different purposes.



(a) Total Time Taken vs Semantic Accuracy for all five LLMs



(b) Total Cost (USD) vs Semantic Accuracy for all five LLMs

Figure 5.15.: Scatter plots of the Total Time Taken and 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.

6. Conclusion

This thesis explored the potential of automating the annotation of mathematical identifiers (grounding of formulae) in scientific papers using Large Language Models (LLMs) such as GPT-3.5 and GPT-4 from OpenAI, as well as some open-source alternatives. The goal was to streamline the process of coreference resolution and formula grounding, traditionally a laborious and expensive task demanding extensive manual effort.

Our research utilised MioGatto, a Math Identifier-oriented Grounding Annotation Tool, as the base platform for annotation. We developed a method to generate a dictionary of mathematical identifiers and their possible descriptions using LLMs and then associate each instance of an identifier with its appropriate definition based on the given context.

We evaluated the results using two primary metrics: the CoNLL score, which quantifies the quality of coreference clusters, and semantic accuracy, which measures the correctness of the annotations. The performance of the LLMs was also assessed in terms of the coverage of annotation, the time and cost of annotation, and the variance in scores due to the stochastic nature of LLMs.

The findings of this study were remarkable. GPT-4 emerged as the most effective model, delivering superior performance regarding the CoNLL score and semantic accuracy. However, it was also the most expensive to operate. GPT-3.5, while not as impressive in its performance, was the most cost-effective and the fastest of the models.

Interestingly, the open-source model StableBeluga2 demonstrated significant potential. Despite operating on a smaller parameter framework than the GPT models, it delivered performance almost on par with them. This is particularly noteworthy given that StableBeluga2 is designed as an "instruct" model, whereas GPT models are general-purpose chat models not explicitly optimised for formula grounding.

The other open-source model we evaluated, Vicuna-33b, lagged noticeably behind the GPT models in performance. However, the fact that it could generate any meaningful dictionary and annotations indicates the potential of open-source LLMs in this domain.

Our research also revealed some intriguing insights. For instance, we found no significant correlation between the CoNLL score and semantic accuracy, indicating that these two metrics reflect distinct facets of the paper. We also found that whether a paper was part of the training dataset did not substantially impact the CoNLL score.

The findings of this study have significant implications. They demonstrate the potential of proprietary and open-source LLMs in automating the annotation of mathematical identifiers in scientific papers. This could significantly streamline the process of corefer-

ence resolution and formula grounding, making it faster, more cost-effective, and more accessible.

However, our research also highlights the challenges in this domain. The stochastic nature of LLMs, the complexity of mathematical identifiers and their context, and the lack of a standard measure of semantic accuracy all add to the task's complexity.

In conclusion, this thesis has significantly contributed to mathematical language processing. It has demonstrated the potential of LLMs in automating the annotation of mathematical identifiers and has laid the groundwork for future research in this domain. However, there is still much work to be done. Future research could explore the use of other LLMs, refine the methods used in this study, and develop more sophisticated measures of semantic accuracy. Despite the challenges, the potential benefits of automating this process are immense, and the progress made in this study is a promising step towards realising this potential.

We addressed the research questions proposed in Chapter 1 throughout this thesis. To provide a cohesive understanding, we will summarise the answers to these questions in the context of our findings:

1. **Efficacy of LLMs:** LLMs, specifically GPT-3.5, GPT-3.5-16k, GPT-4, and some open-source LLMs, have proven to be highly effective in generating accurate annotations for mathematical identifiers. GPT-4 emerged as the most effective model, delivering superior performance in both the CoNLL score and semantic accuracy, although it was also the most expensive to operate. GPT-3.5, while not as impressive in its performance, was the most cost-effective and the fastest of the models. The open-source model StableBeluga2 also demonstrated significant potential, delivering performance almost on par with the GPT models.

2. **Contextual Understanding:** LLMs have shown a substantial ability to disambiguate mathematical identifiers based on context, given the inherent polysemy of these identifiers. This was reflected in the high CoNLL Score, with GPT-4 achieving the highest of 78.08 on average of all 40 papers.

3. **Coverage of Annotation:** LLMs could annotate many papers, with GPT-4 achieving the highest coverage. However, the coverage was not 100% in most cases, indicating room for improvement.

4. **Accuracy concerning Ground Truth:** The annotations generated by LLMs were highly accurate compared to the ground truth provided by manual annotations. This was reflected in the high semantic accuracy scores achieved by the models, where GPT-4 achieved even 100% on one paper and an overall high weighted average of 88.88%. However, manual evaluation of semantic accuracy is a laborious process, and future research could develop more sophisticated measures of semantic accuracy.

5. **Efficiency:** The automation process using LLMs significantly reduced the time required for annotating scientific papers, and the cost savings were substantial, especially when using GPT-3.5. However, the cost and time efficiency varied between models, with GPT-4 being the most expensive.

6. **Limitations of Automation:** While LLMs have shown great promise in automating the annotation process, there are limitations. The stochastic nature of LLMs can lead to variability in the results, and the complexity of mathematical identifiers and their context can pose challenges. Furthermore, there is a lack of a standard measure of semantic accuracy, adding to the complexity of evaluating the results. Moreover, LLMs are not an oracle; therefore, their annotations can not be simply treated as the golden truth. The annotations generated by LLMs must be taken with a pinch of salt. Human annotations are not perfect by any means, either. Therefore, both LLMs and humans are not 100 per cent reliable techniques to generate annotations; however, LLMs can undoubtedly be used to assist humans for this purpose.

In conclusion, the research questions posed at the outset of this thesis have been comprehensively addressed, and the findings have demonstrated the immense potential of LLMs in automating the annotation of mathematical identifiers in scientific papers. However, there are still challenges to be overcome and areas for improvement, which provide exciting avenues for future research in this field.

7. Future Work

While this thesis has made significant strides in the grounding of formulae, it also opens up numerous avenues for future research and development. The following are some of the promising directions for further exploration:

1. **Improving Semantic Accuracy:** Although the semantic accuracy achieved in this study was high, there is still room for improvement. Future work could explore more sophisticated measures of semantic accuracy and develop methods to improve the correctness of the annotations.

2. **Expanding Model Selection:** This study primarily focused on GPT models and a select few open-source LLMs. However, there are numerous other LLMs available that could be harnessed for this task. Future research could explore the use of other models and compare their performance. Another possibility is to use GPT-4 to generate high-quality dictionaries but use GPT-3.5 for the association of the definitions to the identifiers. This way, cost can be reduced by up to 80%, and a high performance can still be guaranteed.

3. **Improving Coverage:** While the coverage of annotation achieved in this study was high, it was incomplete. Future work could improve the coverage, aiming to achieve 100% annotations of the papers.

4. **Reducing Costs:** Although the automation process significantly reduced the time and cost of annotation compared to manual methods, the cost of operating some models, particularly GPT-4, was still high. Future research could explore reducing these costs, making the automation process even more cost-effective.

5. **Incorporating Feedback Mechanisms:** One potential avenue for future exploration is incorporating feedback mechanisms into the annotation process. This could allow for continuous improvement of the annotations over time.

6. **Semantic Accuracy Metric:** Calculating the semantic accuracy manually is tedious. Having a method to calculate it automatically would save further hours in evaluating the new methods for grounding formulae.

A significant obstacle in the current research is the financial cost associated with annotation, a burden that could be better. To address this issue, our future work aims to leverage open-source LLMs for dictionary generation, which would be "cost-free,"

assuming the availability of the necessary hardware. The next step would be to employ a custom-trained machine-learning model to handle the association of each occurrence of mathematical identifiers.

By integrating these components—dictionary generation via open-source LLMs and automated association through a machine learning model—we aim to create a cost-effective, locally executable solution for formula grounding. This approach would alleviate the financial constraints and democratise access to this valuable research tool.

In conclusion, while this thesis has made significant contributions to the field of mathematical language processing, there is still much work to be done. The potential benefits of automating the annotation of mathematical identifiers are immense, and the progress made in this study is a promising step towards realising this potential. The avenues for future research outlined here provide exciting opportunities for further advancements in this field.

A. Appendix

This chapter contains all the information that we deemed to be not crucial for the main content.

A.1. Definitions

Definition 1 (Token) *A token is an instance of a sequence of characters in some particular document that are grouped together as a useful semantic unit for processing.* [1]

Definition 2 (Prompt) *A prompt is a question, statement, or request that is input into an AI system to elicit a specific response or output.*[2]

A.2. XML Encoding Example

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

A.2.1. 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>
```

[1] https://nlp.stanford.edu/IR-book/html/htmledition/tokenization-1.html

[2] https://dev.to/avinashvagh/understanding-the-concept-of-natural-language-processing-nlp-and-prompt-engineering-35hg

```
            <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.2.2. 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>μ</mi>
    <mn>0</mn>
  </msub>
  <mrow>
    <mo fence="true" form="prefix">(</mo>
    <msub>
      <mi>I</mi>
      <mtext>enc</mtext>
    </msub>
    <mo>+</mo>
    <msub>
      <mi> ϵ </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>
```

Abbreviations

GPU Graphical Processing Units

LLMs Large Language Models

NLP Natural Language Processing

POS Part-of-Speech

STEM Science, Technology, Engineering and Mathematics

VRAM Video RAM

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 (2023). *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*.