Variable Extraction for Model Recovery in Scientific Literature

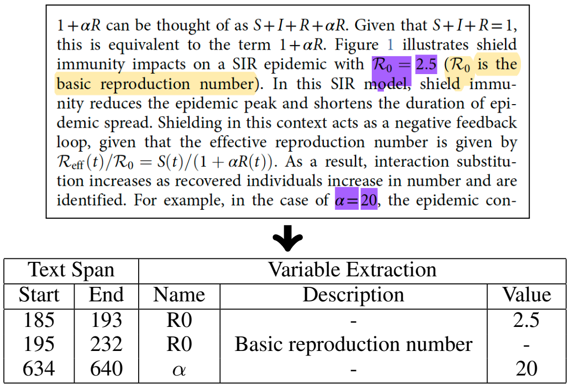
Chunwei Liu[1], **Enrique Noriega**[2],
Adarsh Pyarelal[2], **Clayton T. Morrison**[2], **Michael Cafarella**[1]
[1]MIT CSAIL [2]The University of Arizona
{chunwei, michjc}@csail.mit.edu
{enoriega, adarsh, claytonm}@arizona.edu

Abstract

Due to the increasing productivity in the scientific community, it is difficult to keep up with the literature without the assistance of AI methods. This paper evaluates various methods for extracting mathematical model variables from epidemiological studies, such as “infection rate (α),” “recovery rate (γ),” and “mortality rate (μ).” Variable extraction appears to be a basic task, but plays a pivotal role in recovering models from scientific literature. Once extracted, we can use these variables for automatic mathematical modeling, simulation, and replication of published results. We also introduce a benchmark dataset comprising manually-annotated variable descriptions and variable values extracted from scientific papers. Our analysis shows that LLM-based solutions perform the best. Despite the incremental benefits of combining rule-based extraction outputs with LLMs, the leap in performance attributed to the transfer-learning and instruction-tuning capabilities of LLMs themselves is far more significant. This investigation demonstrates the potential of LLMs to enhance automatic comprehension of scientific artifacts and for automatic model recovery and simulation.

1 Introduction

The surge in scientific publications, now exceeding five million articles annually [1], represents a challenge for any individual or group seeking to comprehensively review the state of the art of any given discipline. The sheer size of the information warrants the use of automated information extraction technologies to sort through and navigate vast scientific corpora. In this work, we study the scientific literature that concerns mathematical modeling, in order to aid model recovery (Pyarelal et al., 2020; Sharp et al., 2019; Schaffhauser et al., 2023): the creation of symbolic representations of mathematical models through information extraction methods applied to the scientific literature[2].



$1+\alpha R$ can be thought of as $S+I+R+\alpha R$. Given that $S+I+R=1$, this is equivalent to the term $1+\alpha R$. Figure 1 illustrates shield immunity impacts on a SIR epidemic with $\mathcal{R}_0 = 2.5$ ($\mathcal{R}_0$ is the basic reproduction number). In this SIR model, shield immunity reduces the epidemic peak and shortens the duration of epidemic spread. Shielding in this context acts as a negative feedback loop, given that the effective reproduction number is given by $\mathcal{R}_{\text{eff}}(t)/\mathcal{R}_0 = S(t)/(1+\alpha R(t))$. As a result, interaction substitution increases as recovered individuals increase in number and are identified. For example, in the case of $\alpha = 20$, the epidemic con-

Text Span		Variable Extraction		
Start	End	Name	Description	Value
185	193	R0	-	2.5
195	232	R0	Basic reproduction number	-
634	640	α	-	20

Figure 1: Example of variable extraction from a scientific paper text, illustrating the process of identifying and extracting elements such as the variable name, description, and initial value into a structured format. The figure highlights different types of extraction: variable description pairs in light orange and variable value pairs in light purple.

We introduce the task of *variable extraction*: the identification and organization elements such as variable names, descriptions, and initial values into a structured format, as illustrated in Figure 1. Variable extraction is a crucial step toward model recovery as it unlocks the basic units of models presented in scientific papers. By doing so, it not only deepens the understanding of the research but also facilitates the further rebuilding and enhancement of these models.

The complexity of variable extraction arises from the diverse forms and locations in which variables can appear within a document. Variables may be embedded in text, figures, tables, or even scattered throughout the paper as single characters, multiple words, single values, or ranges. This variability, coupled with their interdependencies,

[1] https://wordsrated.com/number-of-academic-papers-published-per-year/

[2] https://www.darpa.mil/research/programs/automating-scientific-knowledge-extraction-modeling

Proceedings of the 1st Workshop on AI and Scientific Discovery: Directions and Opportunities, pages 1–12
May 3, 2025 ©2025 Association for Computational Linguistics

underscores the importance and challenge of this task. Effective variable extraction is essential for identifying errors in models and converting them into executable code, which improves the accuracy and practicality of scientific research.

Until recently, the extraction of model variables from texts commonly employed conventional machine learning methods such as named-entity recognition (Tjong Kim Sang and De Meulder, 2003) and relation extraction (Zhang et al., 2017). However, the emergence of Large Language Models (LLMs) (Touvron et al., 2023; Jiang et al., 2023; OpenAI, 2024) has marked a significant change. With their enhanced natural language processing (NLP) capabilities, LLMs provide new options to enhance the efficiency and effectiveness of scientific text analysis, particularly in the extraction of variables and the broader process of model recovery.

To investigate the potential of these methods, we annotated 22 scientific papers, creating a public information extraction benchmark. This benchmark is designed to facilitate the evaluation of various variable extraction techniques. Subsequently, We then conduct a comprehensive evaluation of several LLMs designed for the variable extraction task, alongside a rule-based method and an optimized AI pipeline framework to provide additional perspective.

Our evaluation indicated that although no existing solution excels in the variable extraction task, certain configurations could significantly improve the extraction quality. The best-performing baseline model achieved an F1 score of only 0.49 or 0.60, depending on the evaluation metric used. However, by integrating rule-based approaches with LLMs, we enhanced performance, achieving F1 scores of up to 0.53 and 0.64, respectively. This integration highlights the complementary strengths of different methodologies: rule-based approaches provide additional variable extraction options from a different perspective, thus improving the performance of LLM. Overall, LLM-based solutions outperformed conventional rule-based solutions, demonstrating their capability to enhance the automatic comprehension of scientific artifacts and establish a robust foundation for automatic model recovery and simulation. These insights contribute to the ongoing discourse on improving the understanding and utilization of scientific literature, paving the way for more efficient and accurate scientific research in the era of information overload.

2 Related Work

In this work we focus on the intake of scientific literature to identify and recover the elements of mathematical models: *variable descriptions* and *variable values*, as described in text and we rely on NLP methods to recover them.

The field of Information Extraction (IE) is one of the main applications of NLP. It consists on identifying and extracting structured information from human-written text. These structure data, consists of named-entity recognition (Tjong Kim Sang and De Meulder, 2003) and relation extraction (Zhang et al., 2017). The structured data is then leveraged by downstream applications such as building knowledge bases (Shimorina et al., 2022), slot-filling (Chen et al., 2019), visualization for interactions (Noriega-Atala et al., 2023), or performing downstream inference (Lao et al., 2011).

Large language models (Touvron et al., 2023; Jiang et al., 2023; OpenAI, 2024), with their increasing versatility have become a useful tool for information extraction (Xu et al., 2024). Building on traditional models and LLMs, numerous systems have been proposed to automatically optimize AI-powered analytics and information extraction according to user preferences (Zheng et al., 2023; Chen et al., 2023; Liu et al., 2024; Patel et al., 2024; Lin et al., 2024; Liu et al., 2025). This work takes inspiration from these methods to identify and extract variable information.

Due to its sheer size, scientific literature is frequently the subject of IE research. Some disciplines, such as health sciences and biomedical research, have received a lot of attention due to their high potential for impact. Because of this, there exists a solid record of research activity around that has produced multiple high-quality datasets (Mohan and Li, 2019; Kim et al., 2013; Ohta et al., 2013; Saier et al., 2024) and systems (Valenzuela-Escárcega et al., 2018; Neumann et al., 2019; Wang et al., 2018) focused on clinical and medical applications.

Prior work at extracting mathematical elements has used various classical NLP methods. A CRF model to align mathematical expressions with their definitions (Yoko et al., 2012); a pattern-based data mining method to build mathematical ontologies from LaTeXsources (Jeschke et al., 2007); a NER system for abstract mathematical concepts (Collard et al., 2022) extracting mathematical elements from scientific text. Our work builds upon the ideas from prior research and introduces a high-quality, manually curated dataset featuring annotations of variable descriptions and values extracted from a corpus of scientific literature about COVID-19 and earth sciences. Utilizing this annotated dataset, we have comprehensively evaluated the most popular LLMs, machine learning models, and their combinations, assessing their effectiveness in identifying and extracting this critical information.

3 Variable Extractions Dataset

The benchmark comprises excerpts extracted from 22 papers that focus on pandemic research, specifically available at the benchmark repository[3] These papers typically introduce at least one epidemiological model, providing detailed descriptions and evaluations of the models and their variables. Collectively, the research papers address the challenge of modeling and forecasting the spread of COVID-19 under various scenarios and interventions. They explore a range of modeling approaches, including standard models like SIR and SEIR, as well as more complex frameworks such as COVID-ABS and Covasim. These models are used to analyze the effects of government interventions and to predict the trajectory of the pandemic in different regions. In addition, studies emphasize the extraction and annotation of relevant variables and parameters from the literature, with the aim of enhancing the precision and applicability of these epidemiological models in real-world scenarios.

3.1 Human Annotation

In our study, we meticulously annotated a set of documents to facilitate the extraction and analysis of scientific variables and their contextual data. The annotation process was designed to capture three primary types of information: (1) variable names and their descriptions, (2) variable names paired with their corresponding values, and (3) additional metadata, including model card attributes and scenario card attributes. Detailed guidelines for these annotation task are described in Appendix A.

3.1.1 Annotation Process

Annotation requires expertise in mathematical modeling of epidemics, making our current annotations challenging to obtain. Human annotators were tasked with identifying and labeling specific elements in the text according to the following categories:

Variables with Values: Annotators highlighted instances where a variable was directly associated with a numerical value or a range of values. This includes cases where the variable might be implied rather than explicitly stated. For example, annotators would mark the phrase "the estimated reproduction rate in the United States was around 2.5" to capture the variable (reproduction rate) and its value (2.5).

Variable Descriptions: This task involved identifying and highlighting descriptions of variables that explain or define the variable within the context of the document. For instance, the phrase "lambda represents the infection coefficient" would be annotated to link the variable lambda with its description.

3.1.2 Annotation Standards and Tools

The annotation was performed using Adobe Acrobat, which allowed annotators to use different colors to distinguish between the types of annotations, as specified in the guidelines. The standards for annotation emphasized precision, instructing annotators to prioritize accuracy in identifying and marking text elements. Generous alignment standards were applied during the evaluation of the annotations, focusing on the relevance and completeness of the information captured rather than strict adherence to text boundaries.

3.1.3 Quality Control

To ensure the quality and consistency of the annotations, each document underwent a review process. Annotations that were missed or incorrectly marked in the initial round were identified and corrected. This iterative process helped refine the annotations and improve the overall accuracy of the data set.

3.1.4 Post-Processing with Structured Format

After the annotation and quality review process, each paper will have a unified color code mapping for different annotation categories. We utilize pdfannots[4] tool to extract the PDF annotations into JSON format, categorizing the entries and their text spans from the original text. Pdfannots is a program that extracts annotations (highlights, comments, etc.).

For each annotation, we obtain the highlighted text and its surrounding context into a text passage. We aggregated passages shared by multiple annotations to remove redundancy, for example, a

[3] https://github.com/mitdbg/scivar

[4] https://github.com/0xabu/pdfannots

```
{
  "all_text": "1 + αR can be thought of as S + I + R + αR. Given
  ↪  that S + I + R = 1, this is equivalent to the usual form 1 +
  ↪  αR. Figure 1 illustrates shield immunity impacts on a SIR
  ↪  epidemic with (R0 = 2.5) (R0 is the basic reproduction
  ↪  number). In this SIR model, shield immunity reduces the
  ↪  epidemic peak and shortens the duration of epidemic spread.
  ↪  Shielding in this context acts as a negative feedback loop,
  ↪  given that the effective reproduction number is given by
  ↪  Reff(t) / R0 = S(t) / (1 + αR(t)). As a result, interaction
  ↪  substitution increases as recovered individuals increase in
  ↪  number and are identified. For example, in the case of (α =
  ↪  20), the epidemic concludes with less than 20% infected in
  ↪  contrast to the final size of ~90% in the baseline scenario
  ↪  without shielding (Fig. 2).",
  "page": 2,
  "annotations": [
    [185, 193, "R0 = 2.5", "var val"],
   [195, 232, "(R0 is the basic reproduction number)", "var desc"],
    [634, 640, "α = 20", "var val"]
  ],
  "file": "epidemic_model_analysis"
}
```

Figure 2: Example of SciVar JSON output extracted and formatted from an annotated PDF text block in Figure 1.

paragraph containing several variable descriptions will appear only once in the dataset with all its associated annotations attached. From the 22 scientific papers, we have collected 556 text chunks containing 2083 variable-related annotations (1236 for variable descriptions and 847 for variable values). Each text block is configured with a set of annotations, which include character index and span, text extraction, and annotation type. An example of the structured JSON output can be seen in Figure 2.

This post-processing step ensures that the annotations are not only accurately and automated captured but also structured in a way that facilitates further analysis and application in information extraction systems and other research tools.

4 Variable Extraction Approaches

In our evaluation, we utilized diverse approaches, including traditional rule-based extraction models, popular LLMs with varying degrees of enhancement, and an optimized AI pipeline framework.

4.1 Rule-based Information Extraction

We developed a rule-based information extraction system[5] using the Odin language (Valenzuela-Escárcega et al., 2016) to identify and extract variables mentioned in text alongside their associated definitions or descriptions and values associated with them. The rule-based system operates by matching patterns over the syntax of a sentence or phrase. Figure 3 depicts an example rule. With the help of a linguist, we designed a set of rules to match different ways in which a concept or symbol (the variable) is defined (the description) in scientific papers. Similarly, another subset of rules to match numerical values and quantities associated to variables. Rule-based information extraction tools serve as complement to LLM and other deep-learning based approaches. They trade generalization and recall capabilities for higher precision and interpretability.

```
1  - name: description_interpreted
2    label: Description
3    priority: ${priority}
4    type: dependency
5    example: "Beta can be interpreted
       ↪  as the effective contact rate."
6    pattern: |
7        trigger = [lemma="interpret"]
8        description:Phrase = nmod_as
9        variable:Identifier = nsubjpass
```

Figure 3: Example of a pattern-matching rule system designed to detect variable descriptions. The word `interpreted` will anchor the pattern (line 8). Outgoing syntactic dependencies of types `nmod_as` and `nsubjpass` to entities of types `Phrase` and `Identifier` link the rule's trigger to its description and variable arguments, respectively.

4.2 Vanilla LLM Extraction

LLMs have demonstrated exceptional performance on a variety of semantic information extraction tasks. In our study, we established LLM baselines using a vanilla pipeline, in which each LLM was provided with only snippets of text on paper and tasked with extracting variable names, descriptions, and values. To optimize the effectiveness of our approach, we conducted extensive prompt engineering, iterating through more than ten rounds of refinement. These prompts were developed by a team of four PhD or postdoctoral researchers in computer science major, and the most effective prompt was selected for use in our evaluations. Figure 4 illustrates the prompt template that was used in all LLM baselines. In this template, [] serves as a placeholder for the paper text, and the prompt specifies a structured format for the output, with default values provided for optional fields. Additionally, we incorporate a few-shot prompting setup that provides language models with several examples within the prompt to enhance their performance.

[5] https://github.com/ml4ai/skema/tree/main/skema/text_reading/scala

4.3 Tool Enhanced LLM Extraction

LLMs often share similar technical frameworks and have substantial overlap in their training datasets. This commonality can lead them to either overemphasize or overlook certain cases. To mitigate these biases and enhance extraction accuracy, it is beneficial to introduce additional perspectives. Therefore, beyond the standard evaluation using only the paper text, we have also incorporated outputs from a traditional model into our LLM evaluations. This approach is conceptually similar to the tool integration methods used in LangChain (Topsakal and Akinci, 2023); however, our objective is to generate a broader range of candidate options rather than to rely on the presumed high-quality outputs of these tools. As illustrated in Figure 4, these outputs are highlighted in blue font. The `[TOOL EXTRACTION]` provided by the traditional model offers supplementary variable options for consideration. However, in cases of discrepancy, the original text is always prioritized to ensure the fidelity of the information extracted.

Prompt: Please extract variable names and descriptions from the following paper text. You may refer to the provided tool extractions for your reference. Here is some paper text:
[TEXT]
This text may contain model related variables or parameters, their initial values and what they mean. If it does, list each of the variables on a separate line with the following attributes separated by "|":
name | description | numerical value.
If the variable's value uses other variables or there is no value for the variable, output "None" for that variable value; do not hallucinate a variable value or variable description that does not exist in the text.
[OPTIONAL_EXAMPLES]
Meanwhile, we get some variable extractions from another tool for your reference. These extractions may contain false positive or duplication cases. Please pay more attention to the true positive variables:
[TOOL_EXTRACTION]
Please try to extract variables on the original paper text first, then refer to the results from the tool extractions and see if you miss any variables. If you are not sure, please always check the original paper text.

Figure 4: Prompt templates for variable extraction using various setups. The black font indicates the prompt template for a standard LLM. The combination of black and brown fonts represents the template for few-shot prompting. The integration of black and blue fonts denotes the template enhanced by external tools.

```python
import palimpzest as pz

class Variable(pz.Schema):
    """ Represents a variable of a model in a scientific paper"""
    excerptid = pz.Field(desc="The unique identifier for the excerpt", required=True)
    name = pz.Field(desc="The label used for the scientific variable, like alpha or beta", required=True)
    description = pz.Field(desc="A description of the variable", required=False)
    value = pz.Field(desc="The value of the variable", required=False)

# define logical plan
excerpts = pz.Dataset("snippets", schema=pz.TextFile)
output = excerpts.convert(Variable, desc="A variable used or introduced in the paper snippet", cardinality="oneToMany")

# user specified policy and execute plan
policy = pz.MinimizeCostAtFixedQuality(min_quality=0.45)
results = pz.Execute(excerpts, policy=policy)
```

Figure 5: Palimpzest Code for Variable Extraction from Scientific Paper Snippets.

4.4 Optimized AI Pipeline Framework

We also incorporate a system featuring a simple and declarative user interface. Palimpzest is a system designed to streamline AI-powered analytics through declarative query processing (Liu et al., 2024). This system allows users to effortlessly specify analytical queries over unstructured data using a straightforward, Python-embedded declarative language. Users can define their desired data schema and attributes in natural language, enabling Palimpzest to automate complex optimization processes. This automation includes navigating various AI models, employing prompting techniques, and optimizing foundational models, thereby eliminating the need for the laborious tasks of manual pipeline tuning, model selection, and prompt engineering previously required when working with LLMs. By efficiently managing trade-offs between runtime, cost, and data quality, Palimpzest simplifies user interaction and significantly enhances the efficiency and cost-effectiveness of processing large-scale data. These capabilities position Palimpzest as a robust benchmark for evaluating the performance of AI-driven data processing systems in scientific and analytical contexts, ensuring substantial improvements in execution times and costs while maintaining or enhancing data quality.

The Palimpzest code snippet shown in Figure 5 demonstrates a declarative approach to extracting variables from scientific paper excerpts. It defines the 'Variable' class, which details a scientific variable found within the paper excerpt. This class includes fields for the variable's name, description, and value, with only the variable name being required. This setup efficiently captures the essential details needed for variable extraction, streamlining the process of transforming unstructured text into structured data suitable for further analysis.

The code then creates a dataset named "snippets" with the Palimpzest native ‘TextFile’ schema and processes it to convert each snippet into instances of the ‘Variable’ class, identifying variables mentioned in the text. This conversion cardinality ‘oneToMany’ allows for multiple variables per snippet, reflecting the typical structure of scientific excerpts.

Finally, a user-specified policy (‘MinimizeCostAtFixedQuality’) is set to optimize the extraction process by minimizing operational costs while maintaining the quality of the extracted data above a predetermined threshold. The ‘Execute’ function applies this policy to the dataset, demonstrating how Palimpzest simplifies complex data extraction tasks through its declarative programming model.

5 Evaluation

5.1 Experimental Setup

We evaluate a variety of models to assess their performance on the variable extraction dataset. The traditional rule-based model is denoted as `rules`, and an optimized AI pipeline framework is referred to as `Palimpzest`. Additionally, we examine several advanced models from OpenAI, including `GPT3.5 Turbo`, `GPT4 Turbo`, `GPT4o`, and `GPT4o-mini`. We also test two locally served LLMs, `Llama-3-8B-Instruct` and `Mistral-7B-Instruct-v0.2`, which are integrated via vLLM model serving APIs. Each LLM is evaluated using a standard API call, indicated by the prefix *pure_*, and an enhanced version that incorporates outputs from the traditional model, indicated by the prefix *tool_*. The LLM temperature parameter is set to zero to ensure reproducibility.

We executed all baseline models using the prompts or configurations outlined in the previous section. The results are then aligned with the human-annotated ground truth, as illustrated in Figure 2. This alignment is based on the input text chunk ID. Furthermore, we construct all possible candidate pairs by applying the Cartesian product to the sets of predicted extractions and ground truth, grouped by annotation type. This process resulted in a total of $330,558$ candidate pairs for evaluation. For each candidate pair, we employed a set of evaluation metrics to determine whether it qualified as a match.

To evaluate the F1 score in our study, we meticulously track the ground truth and prediction sets for each text chunk. During the evaluation process, when an evaluator confirms a match (though the criteria for a match may vary across different metrics), the index of the matched candidate pair is recorded in both the ground truth and prediction entries for that specific pair. After evaluating all candidate pairs associated with a given text chunk, we calculate the recall as the ratio of entries with at least one match in the ground truth set. Similarly, precision is calculated as the ratio of entries with at least one match in the prediction set. The F1 score is then computed using the harmonic mean of precision and recall, providing a balanced measure of the model’s accuracy in variable extraction tasks. In cases where the evaluation focuses on specific tasks, such as variable descriptions or variable values extraction only, we count only the corresponding entries and disregard the others.

5.2 GPT-4 as a Similarity Evaluator

We employed the GPT-4 turbo model to perform similarity evaluations, comparing its outputs with a ground-truth dataset to assess precision and accuracy across different tasks. Depending on whether the candidate pair being evaluated corresponds to "var_desc" or "var_val" (examples provided in Figure 2), we use specific prompts as illustrated in Figure 6. To ensure conciseness, we limit the output token length to one.

Prompt for Variable Description/Value:

You are a human evaluator. The following pair of text describes a variable and its description/value.
[VAR_DESC_A]/[VAR_VAL_A]
[VAR_DESC_B]/[VAR_VAL_B]
Please check if they mean the same. Answer y or n.

Figure 6: GPT4 Turbo prompt templates for evaluating the consistency of variable descriptions and values.

According to Table 1, no existing solution performs exceptionally well on the variable extraction task. However, the integration of rule-based approaches with LLMs has shown significant improvements. The best-performing baseline model achieved an F1 score of only 0.491, while the integration with LLMs, particularly the GPT-4 variants, enhanced performance, achieving F1 scores as high as 0.525. This represents a 20% improvement over the setups using only LLMs, except for `GPT3.5T` where the integration did not yield a performance boost. Such integration highlights the complemen-

Table 1: Average performance with GPT4 similarity evolution with ground-truth (bold font indicates the best over each setup).

Model	Overall Performance			Variable Descriptions			Variable Values		
	Recall	Precision	F1	Recall	Precision	F1	Recall	Precision	F1
pure_GPT3.5T	0.576	0.337	0.393	0.677	0.361	0.431	0.404	0.305	0.323
tool_GPT3.5T	0.568	0.307	0.369 ↓	0.647	0.318	0.396	0.429	0.281	0.306
3shot_GPT3.5T	0.543	0.412	0.437 ↑	0.558	0.433	0.457	0.521	0.378	0.400
pure_GPT4T	0.655	0.443	**0.491**	0.708	0.428	0.495	0.527	0.460	0.456
tool_GPT4T	0.662	0.451	0.500 ↑	0.711	0.440	0.506	0.559	0.478	0.476
3shot_GPT4T	0.645	0.502	**0.535** ↑	0.650	0.471	0.514	0.629	0.557	0.553
pure_GPT4o	0.647	0.424	0.480	0.708	0.438	0.504	0.513	0.382	0.408
tool_GPT4o	0.689	0.460	0.520 ↑	0.727	0.453	0.526	0.589	0.468	0.483
3shot_GPT4o	0.499	0.360	0.389 ↓	0.486	0.376	0.395	0.525	0.369	0.397
pure_GPT4o-mini	0.619	0.376	0.437	0.693	0.410	0.479	0.499	0.325	0.360
tool_GPT4o-mini	0.694	0.465	**0.525** ↑	0.729	0.446	0.520	0.619	0.481	0.504
3shot_GPT4o-mini	0.545	0.322	0.378 ↓	0.578	0.370	0.426	0.475	0.231	0.282
pure_llama	0.600	0.402	0.446	0.671	0.422	0.483	0.456	0.373	0.372
tool_llama	0.629	0.396	0.451 ↑	0.706	0.411	0.482	0.479	0.354	0.369
3shot_llama	0.488	0.181	0.244 ↓	0.550	0.204	0.279	0.402	0.139	0.180
pure_mistral	0.572	0.234	0.301	0.661	0.220	0.302	0.404	0.285	0.310
tool_mistral	0.564	0.277	0.335 ↑	0.650	0.265	0.343	0.412	0.307	0.317
3shot_mistral	0.493	0.190	0.248 ↓	0.588	0.191	0.262	0.352	0.194	0.217
rules	0.392	0.317	0.320	0.447	0.352	0.358	0.299	0.244	0.245
Palimpzest	0.574	0.451	0.473	0.566	0.435	0.460	0.555	0.453	0.465
structured_GPT4o	0.64	0.443	0.492	0.682	0.435	0.498	0.535	0.446	0.449
structured_GPT4o-mini	0.658	0.424	0.484	0.689	0.406	0.476	0.593	0.442	0.473

tary strengths of diverse methodologies: rule-based approaches provide additional variable extraction options from different perspectives, thereby enhancing the performance of LLMs.

Among the models tested, the tool-enhanced versions generally outperformed their pure counterparts, with `tool_GPT4o-mini` achieving the highest F1 score of 0.525. This indicates that the additional suggestions provided by tool extractions can effectively guide LLMs to achieve better performance. In contrast, the rule-based approach alone (`rules`) demonstrated lower effectiveness, with an F1 score of 0.320, emphasizing the overall superior capability of LLM-based solutions in managing complex extraction tasks.

However, few-shot prompting does not consistently yield improved extraction results, as indicated by Table 1. Only GPT3.5T and GPT4T models showed improvement with the few-shot setting, while others experienced diminished performance. This variability could be attributed to the inherent complexity of the variable extraction task, where the diverse scenarios may not benefit significantly from a few additional examples. Moreover, the inclusion of more tokens in the prompt might dilute the attention mechanism, thereby worsening the results.

The Palimpzest system, utilizing `GPT-4o` as its conversion model, yielded results comparable to `pure_GPT4o`, achieving an F1 score of 0.473. By enforcing a strict format constraint, Palimpzest trades some recall for higher precision, offering a more reliable output without the need for extensive model selection and prompt engineering. This approach not only simplifies the extraction process but also enhances the usability and applicability of the system in practical scenarios, establishing a robust foundation for automatic model recovery and simulation.

Additionally, we conducted a distinct quality assessment for both variable descriptions and variable values, with detailed results presented in Table 1. The observations mentioned above remain consistent across these evaluations. However, almost all baselines demonstrated better F1 scores on the variable descriptions task compared to their overall performance, with the exception of Palimpzest, which excelled in both cases in general but performed slightly better in the variable value extraction task.

5.3 Token-based Evaluation

In addition to the GPT-based evaluation, we examined the token-level precision, recall and F1 scores used for QA and other span prediction NLP tasks (Rajpurkar et al., 2016). Token-level scores account for the correct number of tokens predicted by each method, giving credit based on the proportion of tokens predicted correctly and penalizing for tokens predicted incorrectly. Table 2 shows the token level performance on the variable extrac-

Table 2: Average token-level scores for variable descriptions and variable values.

Model	Overall Performance			Variable Descriptions			Variable Values		
	Recall	Precision	F1	Recall	Precision	F1	Recall	Precision	F1
pure_GPT3.5T	0.622	0.578	0.552	0.730	0.663	0.645	0.458	0.449	0.410
tool_GPT3.5T	0.551	0.527	0.505 ↓	0.636	0.625	0.599	0.421	0.379	0.362
3shot_GPT3.5T	0.514	0.492	0.467 ↓	0.560	0.486	0.483	0.444	0.499	0.443
pure_GPT4T	0.661	0.587	0.571	0.770	0.670	0.666	0.496	0.460	0.428
tool_GPT4T	0.667	0.638	0.610 ↑	0.771	0.712	0.695	0.508	0.527	0.482
3shot_GPT4T	0.638	0.585	0.562 ↓	0.733	0.623	0.623	0.493	0.527	0.471
pure_GPT4o	0.664	0.621	0.595	0.759	0.688	0.673	0.521	0.520	0.477
tool_GPT4o	0.667	0.620	0.599 ↑	0.768	0.686	0.681	0.513	0.521	0.475
3shot_GPT4o	0.541	0.512	0.487 ↓	0.557	0.488	0.482	0.517	0.548	0.495
pure_GPT4o-mini	0.645	0.641	**0.600**	0.766	0.701	0.686	0.463	0.549	0.470
tool_GPT4o-mini	0.659	0.691	**0.640** ↑	0.771	0.773	0.738	0.489	0.567	0.490
3shot_GPT4o-mini	0.644	0.589	**0.564** ↓	0.703	0.579	0.586	0.556	0.604	0.532
pure_llama	0.614	0.599	0.557	0.712	0.718	0.672	0.465	0.417	0.383
tool_llama	0.621	0.630	0.585 ↑	0.723	0.780	0.716	0.466	0.401	0.385
3shot_llama	0.554	0.553	0.511 ↓	0.614	0.607	0.573	0.461	0.470	0.417
pure_mistral	0.609	0.486	0.488	0.718	0.583	0.591	0.444	0.340	0.332
tool_mistral	0.508	0.450	0.435 ↓	0.606	0.551	0.532	0.359	0.295	0.288
3shot_mistral	0.564	0.489	0.482 ↓	0.693	0.592	0.592	0.369	0.332	0.316
rules	0.429	0.498	0.437	0.494	0.583	0.505	0.329	0.369	0.335
Palimpzest	0.569	0.513	0.488	0.648	0.527	0.526	0.448	0.490	0.431

tion dataset. The results don't diverge significantly from the GPT-based evaluation and consistently highlight the strength of LLM-based methods. Crucially, token-level scores rely solely on manual annotations, therefore any conclusions drawn from them are based only on the ground truth and not subject to any potential inaccuracies from a model-based evaluation.

Table 3: Overall Performance with Full Paper Text on Selected Models

Model	Recall	Precision	F1
rules	0.701	0.101	0.172
pure_GPT3.5T_C	0.750	0.250	0.340
pure_GPT4T	0.564	0.488	0.490
tool_GPT4T	0.678	0.467	**0.506**

5.4 Full Paper Context Extraction Evaluation

We conducted variable extraction evaluations using the full text of each of the 22 articles. This approach limits the number of language models that can be used due to the token limits imposed by many LLMs. We present the results of a rule-based model, GPT-3.5T (with chunking the long text into chunks within the model limit) , and GPT-4T. The overall performance is shown in Table 3.

When dealing with the extensive context of a scientific paper, LLMs can struggle to maintain focus, often resulting in lower recall. In contrast, the rules and pure_GPT3.5T_C with chunking options manage to maintain relatively high recall. Overall, even in the context of lengthy texts, integrating tool outputs helps LLMs concentrate on the extraction task, leading to improved results.

6 Conclusion

We have introduced a dataset for extracting variable descriptions and values from scientific literature, a crucial building block for the automated recovery of mathematical models from the literature. We conducted a battery of evaluations using different commercial and open-source LLMs, a rule-based information extraction system, and a declarative AI pipeline framework. In our experiments, we found that LLM-based methods tend to be the most effective methods to identify and extract variable descriptions and values; however, testing ensembles of rule-based and LLM-based information extractions working in tandem, boost the performance yield the best results most of the time. Considering that all the methods tested in this work did not use any form of supervised learning, there is ample room for improvement. In future work, multiple interesting avenues for research can be explored: Using semi-supervised and data-augmentation methods to augment the size of the dataset, and the use of supervised fine-tuning of encoder-based language models for generation and token prediction can improve the accuracy of the results.

7 Acknowledgments

We would like to express our gratitude for the support provided by the DARPA ASKEM Award HR00112220042. Additionally, we extend our appreciation to Patty Gahan and Robyn Kozierok from MITRE for their diligent annotation efforts.

8 Limitations

We recognize that our work has certain limitations. As is common in research involving human annotations, budget and labor constraints have resulted in a relatively small dataset compared to those constructed using automatic or semi-automatic methods. Moreover, the occurrence of mathematical variable descriptions and values within natural language text is inherently sparse due to the nature of the articles we analyzed. Additionally, our study focuses exclusively on English literature, which may limit its generalizability to other languages.

Despite the small size of our dataset, it was curated by multiple domain experts following a well-defined annotation protocol, ensuring high quality. We hope that by releasing this dataset, we can inspire future efforts to curate larger datasets and foster new research in this area.

9 Ethical Considerations

All of the articles annotated in our dataset are published with an open access license. We identify the papers in Appendix B.

References

Qian Chen, Zhu Zhuo, and Wen Wang. 2019. Bert for joint intent classification and slot filling. *ArXiv*, abs/1902.10909.

Zui Chen, Lei Cao, Sam Madden, Tim Kraska, Zeyuan Shang, Ju Fan, Nan Tang, Zihui Gu, Chunwei Liu, and Michael Cafarella. 2023. Seed: Domain-specific data curation with large language models. *arXiv e-prints*, pages arXiv–2310.

Jacob Collard, Valeria de Paiva, Brendan Fong, and Eswaran Subrahmanian. 2022. Extracting mathematical concepts from text. In *Proceedings of the Eighth Workshop on Noisy User-generated Text (W-NUT 2022)*, pages 15–23, Gyeongju, Republic of Korea. Association for Computational Linguistics.

Sabina Jeschke, Marc Wilke, Marie Blanke, Nicole M. Natho, and Olivier F. Pfeiffer. 2007. Information extraction from mathematical texts by means of natural language processing techniques. In *Proceedings of the International Workshop on Educational Multimedia and Multimedia Education*, Emme '07, page 109–114, New York, NY, USA. Association for Computing Machinery.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Jin-Dong Kim, Yue Wang, and Yamamoto Yasunori. 2013. The Genia event extraction shared task, 2013 edition - overview. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 8–15, Sofia, Bulgaria. Association for Computational Linguistics.

Ni Lao, Tom Mitchell, and William W. Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 529–539, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Yiming Lin, Madelon Hulsebos, Ruiying Ma, Shreya Shankar, Sepanta Zeigham, Aditya G Parameswaran, and Eugene Wu. 2024. Towards accurate and efficient document analytics with large language models. *arXiv preprint arXiv:2405.04674*.

Chunwei Liu, Matthew Russo, Michael Cafarella, Lei Cao, Peter Baile Chen, Zui Chen, Michael Franklin, Tim Kraska, Samuel Madden, Rana Shahout, et al. 2025. Palimpzest: Optimizing ai-powered analytics with declarative query processing. In *CIDR 2025*.

Chunwei Liu, Matthew Russo, Michael Cafarella, Lei Cao, Peter Baille Chen, Zui Chen, Michael Franklin, Tim Kraska, Samuel Madden, and Gerardo Vitagliano. 2024. A declarative system for optimizing ai workloads. *Preprint*, arXiv:2405.14696.

Sunil Mohan and Donghui Li. 2019. Medmentions: A large biomedical corpus annotated with UMLS concepts. *CoRR*, abs/1902.09476.

Mark Neumann, Daniel King, Iz Beltagy, and Waleed Ammar. 2019. ScispaCy: Fast and Robust Models for Biomedical Natural Language Processing. In *Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 319–327, Florence, Italy. Association for Computational Linguistics.

Enrique Noriega-Atala, Md. Rahat-Uz-Zaman, Ruchika Bhat, Mladen Jergovic, Stephen G. Kobourov, and Janko Nikolich-Zugich. 2023. Visualizing interaction networks and evidence in biomedical corpora. In *2023 IEEE 16th Pacific Visualization Symposium (PacificVis)*, pages 41–50.

Tomoko Ohta, Sampo Pyysalo, Rafal Rak, Andrew Rowley, Hong-Woo Chun, Sung-Jae Jung, Sung-Pil Choi, Sophia Ananiadou, and Jun'ichi Tsujii.

2013. Overview of the pathway curation (PC) task of BioNLP shared task 2013. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 67–75, Sofia, Bulgaria. Association for Computational Linguistics.

OpenAI. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Liana Patel, Siddharth Jha, Carlos Guestrin, and Matei Zaharia. 2024. Lotus: Enabling semantic queries with llms over tables of unstructured and structured data. *arXiv preprint arXiv:2407.11418*.

Adarsh Pyarelal, Marco Antonio Valenzuela-Escárcega, Rebecca Sharp, Paul Douglas Hein, Jon Stephens, Pratik Bhandari, HeuiChan Lim, Saumya Debray, and Clayton T. Morrison. 2020. Automates: Automated model assembly from text, equations, and software. *CoRR*, abs/2001.07295.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Tarek Saier, Mayumi Ohta, Takuto Asakura, and Michael Färber. 2024. Hyperpie: Hyperparameter information extraction from scientific publications. In *European Conference on Information Retrieval*, pages 254–269. Springer.

Timo Schaffhauser, Daniel Garijo, Maximiliano Osorio, Daniel Bittner, Suzanne Pierce, Hernán Vargas, Markus Disse, and Yolanda Gil. 2023. A framework for the broad dissemination of hydrological models for non-expert users. *Environmental Modelling & Software*, 164:105695.

Rebecca Sharp, Adarsh Pyarelal, Benjamin Gyori, Keith Alcock, Egoitz Laparra, Marco A. Valenzuela-Escárcega, Ajay Nagesh, Vikas Yadav, John Bachman, Zheng Tang, Heather Lent, Fan Luo, Mithun Paul, Steven Bethard, Kobus Barnard, Clayton Morrison, and Mihai Surdeanu. 2019. Eidos, INDRA, & delphi: From free text to executable causal models. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 42–47, Minneapolis, Minnesota. Association for Computational Linguistics.

Anastasia Shimorina, Johannes Heinecke, and Frédéric Herledan. 2022. Knowledge extraction from texts based on Wikidata. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Track*, pages 297–304, Hybrid: Seattle, Washington + Online. Association for Computational Linguistics.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Oguzhan Topsakal and Tahir Cetin Akinci. 2023. Creating large language model applications utilizing langchain: A primer on developing llm apps fast. In *International Conference on Applied Engineering and Natural Sciences*, volume 1, pages 1050–1056.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Marco A Valenzuela-Escárcega, Özgün Babur, Gus Hahn-Powell, Dane Bell, Thomas Hicks, Enrique Noriega-Atala, Xia Wang, Mihai Surdeanu, Emek Demir, and Clayton T Morrison. 2018. Large-scale automated machine reading discovers new cancer-driving mechanisms. *Database*, 2018:bay098.

Marco A. Valenzuela-Escárcega, Gus Hahn-Powell, and Mihai Surdeanu. 2016. Odin's runes: A rule language for information extraction. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 322–329, Portorož, Slovenia. European Language Resources Association (ELRA).

Yanshan Wang, Liwei Wang, Majid Rastegar-Mojarad, Sungrim Moon, Feichen Shen, Naveed Afzal, Sijia Liu, Yuqun Zeng, Saeed Mehrabi, Sunghwan Sohn, and Hongfang Liu. 2018. Clinical information extraction applications: A literature review. *Journal of Biomedical Informatics*, 77:34–49.

Derong Xu, Wei Chen, Wenjun Peng, Chao Zhang, Tong Xu, Xiangyu Zhao, Xian Wu, Yefeng Zheng, Yang Wang, and Enhong Chen. 2024. Large language models for generative information extraction: A survey. *Preprint*, arXiv:2312.17617.

Kristianto Giovanni Yoko, Minh-Quoc Nghiem, Yuichiroh Matsubayashi, and Akiko AIZAWA. 2012. Extracting definitions of mathematical expressions in scientific papers. *The 26th Annual Conference of the Japanese Society for Artificial Intelligence*, JSAI2012:3P1IOS2a3–3P1IOS2a3.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, pages 35–45.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Jeff Huang, Chuyue Sun, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. 2023. Efficiently programming large language models using sglang. *arXiv preprint arXiv:2312.07104*.

A Annotation Guidelines

A.1 General Conventions

Annotators should prioritize precision over recall in their first round of annotation on each document. Annotations that are missed (i.e., elements that should be annotated but haven't been) can be corrected upon document review. The task is to identify all instances of such expressions in each text, including in the title, abstract, and figure and table captions. Figures, tables, keywords, floating equations, acknowledgment sections, and references, however, are not annotated.

A.2 Nested Annotations

"Nested annotations" can happen when annotators tag nested elements that occur within the boundaries of a longer annotation. This task prioritizes tagging the longest extent of an expression in cases of overlapping annotation. For example, in the "variable with value" expression below, "United States" is not tagged even though it is a "location context."

- `the [estimated reproduction rate in the United States as a whole stood at around 2.5].`

A.3 Events

The evaluation will use generous alignment standards that do not require exactly matching extents but it is preferable, though not mandatory, to exclude white space and punctuation when annotating.

A.4 Annotation Types

Annotators are asked to use assigned colors to highlight five different types of annotations: Variables with Values, Variable Descriptions, Locations and Temporal Contexts, and Model Card annotations. The guidelines below provide further instructions for each annotation type.

Variables with Values

This entity type captures variables with their numeric values. Values expressed as ranges should be annotated. To qualify as a Variable with Value, the expression must contain a number assigned to a simple expression.

This entity type is marked in blue. Some examples include:

- growth rate of 0.01
- r0 = 1.2
- Reproduction numbers of COVID-19 vary in different studies and regions of the world (in addition over time) but have generally been found to be between 1.5 and 6.
- the estimated reproduction rate in the United States as a whole stood at around 2.5.
- The number of unquarantined infected cases was 1200.
- Beta represents a value 1-3

Do annotate a value expression as a Variable with Value even when the variable is implied, and not explicit. Annotate and then add a pop-up note to indicate the implied variable. For example, "334" would be annotated as a Variable with Value and then noted as "Implied variable: unquarantined infected cases."

- The number of unquarantined infected cases was 1200. The number[6] had been 334.

Do not include confidence intervals in the extent of the variable with value expression:

- the mean control reproductive number is 6.47 (95% CI 5.71-7.3)

Do not tag equations as variables with values.

- $I(t) = Io_e^{xt}$

Variable Descriptions

This entity type captures descriptions of variables. In the case of complex phrases, highlight the whole span of text that contains the complete information. This entity type is highlighted in yellow. Some examples include:

- lambda represents the infection coefficient
- infected (asymptomatic or pauci-symptomatic infected, undetected)
- B is the number of such variables
- y is the recovery rate constant
- S is the total number of infected
- normalized infection i

[6] The number refers to the unquarantined infected cases. As such, this is a way to handle coreference with implied variables.

- I infections
- time $Td = ln2/\alpha$
- Hp is the Hubble constant
- Susceptible, Exposed, Infectious versus S Susceptible, E Exposed, I Infectious

Do not tag vacuous expressions as variable descriptions, such as:

- parameter v

B Dataset Articles

Table 4 contains the list of DOIs of the articles annotated to create the variable descriptions and values dataset.

DOI
10.1073/pnas.2112532119
10.1287/opre.2022.2306
10.1101/2020.04.09.20047498
10.1016/j.chaos.2020.110088
10.1371/journal.pcbi.1009149
10.1038/s41591-020-0883-7
10.1073/pnas.2006520117
10.1016/j.idm.2020.03.001
10.1016/j.chaos.2020.109846
10.1371/journal.pone.0236386
10.3390/ijerph18179027
10.1038/s41467-020-20544-y
10.1038/s41598-022-06159-x
10.1016/j.physa.2020.125498
10.1007/s40484-020-0199-0
10.1038/s41591-020-0895-3
10.1186/s13104-020-05192-1
10.1016/j.healthplace.2020.102404
10.3390/jcm9020462
10.1016/j.idm.2020.02.001
10.1175/JPO-D-20-0286.1
10.1002/jmv.25827

Table 4: Digital Object Identifiers (DOI) of the articles used to build the annotations of the dataset.