Accelerating Literature Data Extraction using AI at the IPAC Archives

Nicholas Susemiehl Nima Chartab Meca Lynn Ricky Nilsson
IPAC, California Institute of Technology
1200 E. California Blvd., Pasadena, CA 91125, USA
{nsuse, nchartab, mlynn, ricky}@ipac.caltech.edu

Abstract

The repositories of the NASA Extragalactic Database (NED) and NASA Exoplanet Archive (NEA) provide the scientific community with regularly-updated records of published data on observed astronomical objects. Over their years of operation, both archives have seen increases in the rate of publications that are relevant to their holdings. We are developing an AI-powered data extraction tool to increase the rate at which NED and NEA can process published articles into their databases. This is accomplished through a multi-stage pipeline that combines large language models (LLMs) with programmatic techniques and human supervision to create formatted database load files. Pipeline evaluations on sampled human-made load files estimate a macro-level recall of 0.89 for NED and 0.83 for NEA with typical runtimes of 20 minutes per-paper. While this tool is still being developed (aided by human feedback), present results suggest that contemporary LLM-based approaches are useful for complex structured data extraction. Future work remains to improve the pipeline's precision and further adapt it to the wide variety of articles present in the literature.

1 Introduction

The NASA Extragalactic Database (NED)[1] and NASA Exoplanet Archive (NEA)[2] are two astronomical data repositories operated by IPAC at the California Institute of Technology. NED has collected over 1.1 billion distinct objects, including galaxies, quasars, and groups/clusters. NEA seeks to provide a complete list of confirmed exoplanets, which now number over 6,300, and their stellar hosts. Both archives ingest their holdings from the text and tables of published astronomical literature which present the discovery of new objects or new measurements of existing ones. Examples of data the archives extracts include the object's type, coordinates, and redshift for NED and a planet's mass, transit midpoint, and detection method for NEA. Data extracted from articles must be transformed to an archive-specific database load file. Both archives face challenges ingesting relevant articles at the rate they are published. Creating load files is a significant manual effort at NED, with hundreds of papers processed each month. Anticipated exoplanet yields from future missions have the potential to increase NEA's holdings by hundreds of thousands of objects (Perryman et al., 2014; Penny et al., 2019; Wilson et al., 2023; Matuszewski et al., 2023; Ge et al., 2024). Creating load files by hand can take minutes or hours depending on article complexity. Given these realities, it has become important for the data archives at IPAC to enhance the throughput of their data ingestion pipelines.

The field of natural language processing (NLP) has yielded tools that are increasingly capable of mining data from the text of scientific journal articles. Transformer-architecture large language models (LLMs), which use attention mechanisms to create dynamic contextual understanding of text (Vaswani et al., 2017), have been shown to be applicable to a wide range of NLP tasks even without expensive fine-tuning (Brown et al., 2020). Recent advances in LLMs have greatly expanded the possibilities of an AI-driven data extraction pipeline: enlarged context limits (now ranging from 250 thousand to over 1 million tokens) allow for the inclusion of multiple full articles and their tables in a single prompt (Gemini Team et al., 2024), increased instruction-following and thinking capabilities help the models reason over the numerous extraction rules and complex article structures (Wei et al., 2022; Guo et al., 2025), and tool usage enables dynamic context-driven unit conversions (Yao et al., 2022; Schick et al., 2023).

The goal of this work is to produce a tool which accelerates the processes of data extraction and

[1] https://ned.ipac.caltech.edu/
[2] https://exoplanetarchive.ipac.caltech.edu/

Proceedings of the 4th Workshop on Artificial Intelligence for Scientific Publications, pages 166–173
November 10, 2026 ©2026 Association for Computational Linguistics

load file creation for NED and NEA. There is no expectation that the load files created using these tools will be perfect, so human supervision and feedback is integral to this process.

2 Methods

The data ingestion pipelines for NED and NEA contain a variety of AI, programmatic, and human-driven components. At a high level, this pipeline is identical for both NED and NEA: newly published articles are discovered through listing services, relevant articles are identified, necessary data is extracted from relevant articles, and load files are created to ingest these data into the archives' holdings.

2.1 Data Retrieval

Each module of this work uses the cleaned body text of a journal article and/or tables published either in the article itself or externally. Articles are retrieved via the Science Explorer[3] (SciX) API according to a format priority order as accessible: first the publisher version of the article's HTML, then the publisher PDF, then the pre-print HTML, and finally the pre-print PDF. HTML files are converted to strings using Beautiful Soup (Richardson, 2007) and PDF files with Docling (Livathinos et al., 2025). Tables are converted to Pandas dataframes (Wes McKinney, 2010), images (Artifex Software, Inc., 2026), and markdown key-value format[4]. Preprocessing normalizes characters and strips extraneous formatting. External data tables hosted on publisher websites are accessed when possible. The impacts of text and table parsing errors are discussed in the Appendix (7.4).

2.2 Relevance Classification

Both NED and NEA use trained machine learning classifiers to predict the probability that an article is relevant to their holdings. The relevance classifier used by NED (Chen et al., 2022) is based on the Stanford Classifier (Finkel et al., 2005), while the NEA tool uses a finetuned INDUS (Bhattacharjee et al., 2024) model. Both of these tools successfully accelerate the task of scientists to identify relevant papers.

[3]https://scixplorer.org/

[4]https://www.improvingagents.com/blog/best-input-data-format-for-llms/

2.3 Data Extraction

Once a relevant article is identified, the data it presents in texts and tables must be extracted to structured outputs. Data ingestion follows a sequential pipeline involving four main LLM-backed stages. This begins with the pre-processing of article text/tables and ending with a draft database load file. An article's text and tables follow parallel paths: LLM prompts only include the body text and smaller tables while all tables go through a programmatic extraction code. Pre-processing and extraction currently use the LLM Qwen 3.8 27B (BF16 quantization; Qwen Team, 2026) while validation and normalization are performed with Gemma 4 31B (Q4_K_M quantization; Gemma Team et al., 2026). These be swapped with any tool-capable decoder LLM. These models are distributed over 6 x NVIDIA L40S (46 GB VRAM) for production and development, with one article running on a single card in the cluster. The prompting of these models employs in-context learning by providing examples of extracted data structures to accomplish this task without expensive finetuning. A diagram of the pipeline is available in the Appendix (Figure 1).

2.3.1 Pre-processing

The article text and tables go through initial preprocessing steps to prepare for extraction. The two main components identify the locations of object names and parameter labels (table column/row headings) before full extraction. Pre-processing includes a first pass at object name detection by searching the article text for matches to the NED/NEA accepted regular expression name patterns. This is inexpensive but not robust to variations from expected object name spellings (e.g. abbreviations), so name detection is performed more robustly in Stage 1 (Section 2.3.2). Parameter labels are identified two ways: similarity matching to a dictionary with keys as the preferred labels and values with lists of previously-seen labels and an LLM call prompted to find labels. Found names and labels are tagged within the text.

2.3.2 Stage 1: Name Extraction

The first data extraction step identifies the names of astronomical objects mentioned in the article and classifies them according to their relevance. The LLM is prompted to return verified object names as well as their basic type (extragalactic/galactic/solar system/exclude for NED and exoplanet/star/solar

system/exclude for NEA). This prompt includes names tagged via regular expressions in the previous step as likely object names, but not necessarily the only ones. Objects classified as extragalactic for NED and exoplanet/star for NEA (along with both primary and secondary objects) are then passed to the next stage for data extraction.

2.3.3 Stage 2: Parameter Extraction

The second primary extraction stage is an LLM call with the processed article text, small tables, and relevant names identified in Stage 1 included in the prompt. Tables with fewer than 200 rows (selected to preserve context usage and written in markdown key-value format) are included in the prompt in the markdown key-value format. The model is instructed to extract all NED and NEA parameters for the provided objects simultaneously. Descriptions are provided for each parameter along with examples. The text and previously-extracted data from the most similar article selected via retrieval augmented generation (RAG; Lewis et al., 2020) are included in the Stage 2 prompt as a task example. The RAG process uses the Qwen 3 embedding model (Zhang et al., 2025) to vectorize the first 32,000 tokens of an article and selects the most similar article to the present case by computing cosine similarities against a repository of previously-embedded articles. Stage 2 extracts the parameter data exactly how they appear in the text and returns them in a structured JSON. The Stage 2 outputs also include source texts for each extracted value, snippets of text from the article the LLM used as evidence to support the extraction of a particular data point. The source texts improve the extraction quality and are useful later for validation, normalization, and provenance tracing.

2.3.4 Tabular Data Extraction

A separate path exists for the extraction of data from tables treated as text for two reasons: LLM's required input of strings of texts flattens the two dimensional structure of tables to one dimension, losing associative information, and large tables (some having thousands of rows) consume significant fractions of a model's total token limit. A programmatic approach leverages the structure of tables to deterministically and efficiently extract data. This relies on the identification of object names (from regular expressions and Stage 1) and parameter labels (from dictionary matching and LLM label pre-pass). Once object names and parameter labels are found, the value in the cell that lies at the row/column interception of the labels is extracted. This method is capable of handling common structures such as objects in rows and parameters across columns, the transpose of this, single-object tables without an object name, and object names as row headings with parameters underneath. More generally, any object name that aligns in a row/column with a parameter label can be extracted, making this technique robust to a greater variety of interpreted table formats. Data extracted from this programmatic table code are later merged with the LLM-extracted data, with values from the tabular path favored in conflicting cases.

2.3.5 Stage 3: Validation

Once the main data extraction phases are complete, the results are passed to a series of three LLM calls to validate the previous work using an independent model. Stage 3A validates the recall by double checking that all data from the article that should have been included in the pipeline outputs are present, and adds any missing data (along with source texts). Stage 3B validates the precision by removing or reassigning any data that should not be associated with an extracted object. Stage 3C provides a subjective estimate of the performance in terms of precision/recall/F1 as a 1-10 confidence score. This is estimated according to a set of fifteen counted metrics from the pipeline generation, including what data were added/removed in stages 3A and 3B, names from Stage 1 missing in the generated files, and values without sufficient article evidence. If the confidence score is 6 or lower, one additional loop of stage 3 is performed (this does not repeat further). The confidence scores are used to give operators an estimate of pipeline performance when a human-made truth file is not available for direct score calculation. Future work will develop a more empirical means of deriving a confidence estimate and increase reliability.

2.3.6 Stage 4: Normalization & Load File Creation

All preceding steps dealt with the extraction of data exactly as they appear in the article, but NED and NEA have certain formats and units they require data to be in for ingestion. Stage 4 of the pipeline rewrites the extracted data to the correct format and performs any necessary unit conversions. This includes narrative forms of labels to archive-specific

	N.	Precision		Recall		F1		Runtime (min.)	
		macro	micro	macro	micro	macro	micro	median	max
NED	100	0.6363	0.7793	**0.8930**	**0.8867**	0.6462	0.8295	7.4	108.1
NEA	100	0.6072	0.5100	**0.8265**	**0.8127**	0.6131	0.6267	33.4	90.0

Table 1: Extraction pipeline performance on random samples of NED and NEA articles. Batches of size 100 were chosen to limit maximum runtime. Total NED runtime - 29.1 h. Total NEA runtime - 56.2 h. Development has thus far focused on maximizing recall.

encoding (e.g. 'galaxy' to 'G') and the conversion of certain values in their published unit to their archive-expected unit. First, names extracted by the AI pipeline are matched to archive-canonical forms by querying each archive's web-accessible name resolution service (this is useful for de-duplication and later scoring). Next is a deterministic pre-pass which attempts to establish the unit a value was published in from the extracted source text and convert it to the correct unit programmatically. All extracted values are then added to the fourth stage's LLM prompt along with the unit they are inferred to be presented in and the final required unit. The LLM then uses tool calling to perform deterministic conversions as it deems necessary, based on the article's context. After this final stage, all extracted data are written to a database load file.

2.4 Load File Scoring

Both archives hold repositories of historical database load files which contribute to an evaluation set of 9,758 unique bibcodes for NED and 907 for NEA. These records are used as the ground truths for comparisons against load files generated by the AI-driven extraction pipeline. Precision, recall, and F1 scores are computed to estimate the performance for each individual parameter (including names) and load files as a whole. Scores are calculated by matching object names in the test and truth load files via their queried canonical forms and comparing each AI-extracted value for strict equality with the respective human-extracted value. These are aggregated as the macro-average to assess file-level performance and micro-average to provide a score balanced by the amount of data in each file. Initial development has focused on maximizing the recall over the precision, owing to the relative ease of deleting unneeded data from a generated load file over searching for missed data.

2.5 Web User Interface

We developed a browser-accessible user interface available to NED and NEA curators to work with the data extraction tool. The functionalities of this site include bibcode/article upload for pipeline extraction, truth file upload for scoring, feedback submission, an interactive editor which shows the trace between extracted data and their locations within the original article text, and a chat interface to a Claude Sonnet model to allow for targeted actions (e.g. extract a specific table). The data extraction pipeline runs automatically on the accepted results from the paper relevance classifiers and these outputs are listed on the pipeline webpage daily for operator review.

3 Results

Development on this data extraction pipeline remains active and involves regular benchmarking on larger sets of sampled truth data. We present the results on sets of 100 articles drawn from the NED and NEA truth file repositories following distributions in number of objects, number of parameter types, and number of parameters per object (parameter density) proportional to the truth distributions. Table 1 shows the latest precision, recall, and F1 metrics as well as runtimes for both archives.

3.1 Discussion of Results

Overall performance varies noticeably between NED and NEA. NED articles achieve higher precision and recall, at both macro- and micro-aggregate levels. This is largely due to differences in the types and quantities of data included in load files for each archive. NEA load files often contain more parameters per object (both additional values and their accompanying uncertainties) than the typical NED load file. The latest NED benchmark set contains an average of 6 parameters per object while the NEA articles have an average of 27 parameters per object. Extracting the correct subset of data

while not including extraneous data and assigning the right data to the right objects therefore become more challenging in the NEA case. Differences in the macro- and micro-level aggregates can be attributed to the types of articles included in the samples: articles with more data extracted from them weigh more heavily in the macro-level score than articles with fewer amounts of object data, while the micro-level score aggregates extractions across articles.

3.1.1 File Creation Efficiency

The amount of time it takes a human curator to extract data and write a load file by hand varies according to factors such as the number of objects and parameters in the article (like the AI pipeline), and is typically within fifteen to forty-five minutes. The total runtime of the AI file generation process for a typical article (Table 1) is on a comparable scale to that of a human curator. The AI tool's runtime is also affected by stage 3C-triggered retries, which occurred in 6% of NED articles and 39% in NEA articles, primarily due to low precision confidence. The runtime of the NEA pipeline is further increased by tool calls for unit conversion, which the NED side does not use. While the AI tool writes load files at a similar speed as a human, it is more capable of efficiently processing large batches of articles. The reported runtimes are for processing on a single GPU, while multiple are available for NED and NEA production. In addition, these can run outside of working hours and be made available for a human curators to review later.

3.1.2 Operator Feedback

NED and NEA curators have been actively using the tool in their regular work and providing feedback that has been invaluable in revealing gaps in pipeline capabilities. This includes assistance towards aligning instructions given to the AI with the archives' conventions. While pipeline errors can be caused by parsing errors or convention misalignment, some data points scored in disagreement with the recorded truth may actually represent correct extractions. We observed that the AI tool may extract real data from an article that a human chose not to retrieve (lowering precision) and that the contents of the truth load file may not match the contents of the article because the author made some unreplicable transformation (lowering recall). Instances of this occurring have been noticed anecdotally through feedback but not quantified.

4 Conclusions

Pipeline performance depends heavily on the formatting and verbiage of an article. The articles that perform best are those with small numbers of objects or machine-readable tables. Atypical object name aliases, parameter labels, and table structures can present challenges. The heterogeneity of article format and necessity of careful scientific decisions for data inclusion suggest that AI-driven information retrieval will require a human to remain in the loop for these tasks. However, historical human-made load files and quantitative metrics make improving the pipeline tractable. While there are improvements yet to be made to the extraction performance and the tool's runtime can be significant, it is approaching a level of reliability that will allow for the ready adoption of load files generated autonomously.

4.1 Next Steps

Future development will focus on maximizing pipeline precision and recall while introducing more comprehensive evaluations. We will seek to improve the reliability of article text/table parsing and the stage 3C confidence scores. Larger article sets, performance comparisons between different LLMs, and a characterization the error rates of different components of the tool by running an ablation study will be included in future evaluations. Feedback will continue to be collected and incorporated.

5 AI Usage Disclosure

Almost the entirety of the code used in the implementation of the data extraction pipeline was written by an AI (Claude Opus/Fable) and reviewed by a human developer. None of the text in this article was written by an AI.

6 Acknowledgments

We thank Douglas McElroy, Denis Zaytsev, and Joseph Mazzarella for their important early contributions to this work.

7 Appendix

7.1 Pipeline Diagram

A visualization of the data extraction pipeline is presented in Figure 1 (Appendix).

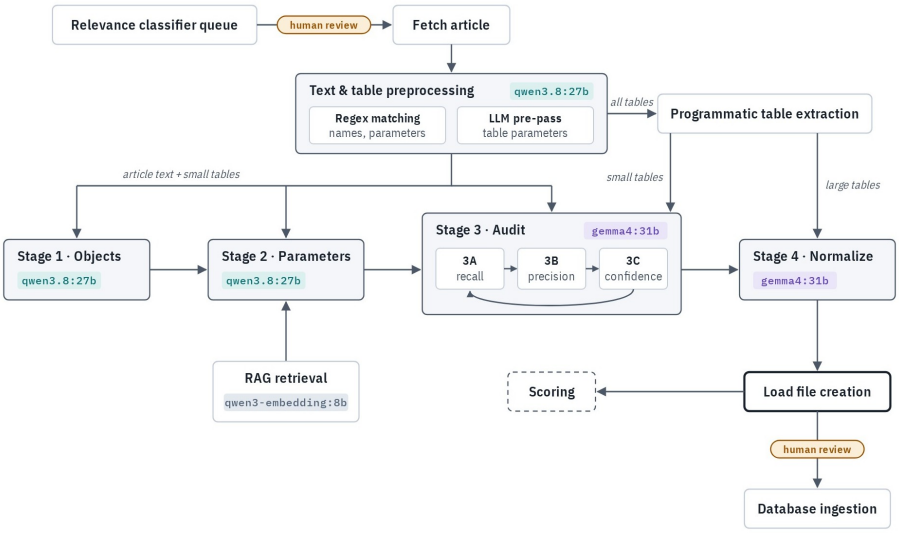


Figure 1: Visualization of pipeline stages. The model stages read the article text and only the small tables. Every table is also parsed programmatically: rows from small tables join the frame before Stage 3 audits it, and rows from large tables join after Stage 3, at Stage 4. Stages 1–3 keep values as printed. Only Stage 4 converts them to archive form. People review the queue before articles are fetched, and each load file before it is ingested.

7.2 User Interface

Figure 2 (Appendix) shows a screenshot of the AI pipeline's web user interface.

7.3 Error Source Analysis

We characterize the sources of errors for each archive according to several categories by counting the false positives and false negatives and examining the differences from truth for each article in the NED and NEA benchmarks (Table 1). These are presented in Table 2 (Appendix). For both archives, precision errors are caused by extra objects and parameters included in the generated file compared to the truth. NED false negatives came more from missing parameters while NEA false negatives were more from missing objects, suggesting object detection is stronger for NED.

	Category	NEA	NED
FP	Objects not in truth	53.5%	62.1%
	Extra parameters or values not in truth	36.1%	21.6%
	Wrong value	10.4%	16.3%
FN	Missed objects	37.2%	19.3%
	Parameter not emitted for a found object	31.0%	41.5%
	Wrong value	31.9%	39.3%

Table 2: Categorized errors for the 100-article NEA and NED benchmarks. Shares are of each archive's false positives (NEA 15,050; NED 3,962) or false negatives (NEA 4,589; NED 1,984).

7.4 Impact of Text/Table Processing on Scores

There is a possibility for errors to be introduced in the pipeline through the imperfect parsing of articles' texts and tables (either from the HTML or PDF source), even though these are performed using reliable packages (See Section 2.1). Conversion text artifacts that have been observed to cause errors include inline math rendered as images and dropped footnotes. Zero NED articles in the latest benchmark had any extraction mistakes due to text parsing errors, but 20% of NEA articles had at least one parsing error. These errors were mostly from periods and Latex formatting dropped from PDF conversions. Examples of table formats that often lead to parsing errors are multi-level headers and object/parameter labels split across multiple rows. In the latest benchmark set, 1% of NED failed to obtain an external table and zero had parsing errors. For NEA, 14% of articles had inaccessible external data and 1% of articles had a PDF parsing error. Parsing errors may be specific to the particular formats used by some publishers, which appear at different rates in NED and NEA holdings. Most NED articles were processed using their HTML while most NEA articles used the PDF due varying publisher access.

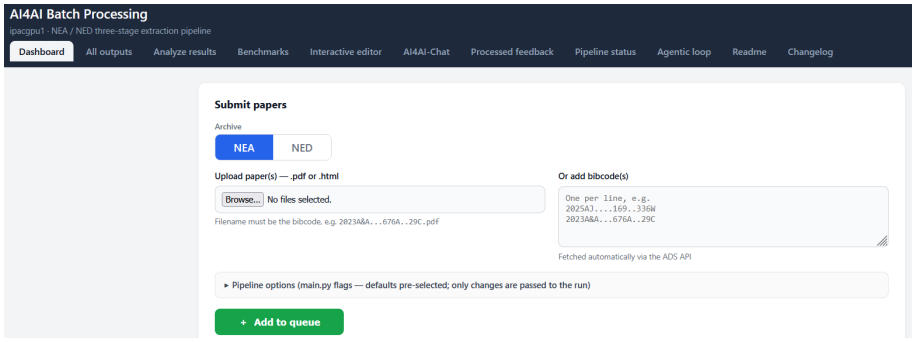


Figure 2: Screenshot the web UI's main page for uploading articles to process. Additional functionalities are in other tabs include viewing results from automated queues, uploading new truth files for scoring, interacting with an AI chat and visual editor interfaces, viewing latest benchmarks, and logging feedback.

References

Artifex Software, Inc. 2026. Pymupdf. Python package for high-performance document processing.

Bishwaranjan Bhattacharjee, Aashka Trivedi, Masayasu Muraoka, Muthukumaran Ramasubramanian, Takuma Udagawa, Iksha Gurung, Nishan Pantha, Rong Zhang, Bharath Dandala, Rahul Ramachandran, Manil Maskey, Kaylin Bugbee, Mike Little, Elizabeth Fancher, Irina Gerasimov, Armin Mehrabian, Lauren Sanders, Sylvain Costes, Sergi Blanco-Cuaresma, and 17 others. 2024. INDUS: Effective and Efficient Language Models for Scientific Applications. *arXiv e-prints*, arXiv:2405.10725.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language Models are Few-Shot Learners. *arXiv e-prints*, arXiv:2005.14165.

Tracy X. Chen, Rick Ebert, Joseph M. Mazzarella, Cren Frayer, Scott Terek, Ben H. P. Chan, David Cook, Tak Lo, Marion Schmitz, and Xiuqin Wu. 2022. Classification of Astrophysics Journal Articles with Machine Learning to Identify Data for NED. , 134(1031):014501.

Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, pages 363–370. Association for Computational Linguistics.

Jian Ge, Hui Zhang, Yongshuai Zhang, Yan Li, Dan Zhou, Haijiao Jiang, Pengjun Zhang, Xinyu Yao, Jiapeng Zhu, Yong Yu, Congcong Zhang, Zhenghong Tang, Jianqing Cai, Chaoyan Wang, Hongping Deng, Wen Chen, Kun Chen, Yingquan Yang, Xuliang Duan, and 50 others. 2024. Progress in the Earth 2.0 (ET) space mission. In *Space Telescopes and Instrumentation 2024: Optical, Infrared, and Millimeter Wave*, volume 13092, page 1309218. International Society for Optics and Photonics, SPIE.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, Soroosh Mariooryad, Yifan Ding, Xinyang Geng, Fred Alcober, Roy Frostig, Mark Omernick, Lexi Walker, Cosmin Paduraru, Christina Sorokin, and 181 others. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv e-prints*, arXiv:2403.05530.

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, Mayank Chaturvedi, Aditya Chawla, Victor Cotruta, Alice Coucke, Phil Culliton, Robert Dadashi, Lucas Dixon, Mohamed Elhawaty, Utku Evci, and 181 others. 2026. Gemma 4 Technical Report. *arXiv e-prints*, arXiv:2607.02770.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. , 645(8081):633–638.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *arXiv e-prints*, arXiv:2005.11401.

Nikolaos Livathinos, Christoph Auer, Maksym Lysak, Ahmed Nassar, Michele Dolfi, Panos Vagenas, Cesar Berrospi Ramis, Matteo Omenetti, Kasper Dinkla, Yusik Kim, Shubham Gupta, Rafael Teixeira de Lima, Valery Weber, Lucas Morin, Ingmar Meijer, Viktor Kuropiatnyk, and Peter W. J. Staar. 2025.

Docling: An Efficient Open-Source Toolkit for AI-driven Document Conversion. *arXiv e-prints*, arXiv:2501.17887.

F. Matuszewski, N. Nettelmann, J. Cabrera, A. Börner, and H. Rauer. 2023. Estimating the number of planets that PLATO can detect. , 677:A133.

Matthew T. Penny, B. Scott Gaudi, Eamonn Kerins, Nicholas J. Rattenbury, Shude Mao, Annie C. Robin, and Sebastiano Calchi Novati. 2019. Predictions of the WFIRST Microlensing Survey. I. Bound Planet Detection Rates. , 241(1):3.

Michael Perryman, Joel Hartman, Gáspár Á. Bakos, and Lennart Lindegren. 2014. Astrometric Exoplanet Detection with Gaia. , 797(1):14.

Qwen Team. 2026. Qwen3.8-Max: A new bar for coding and cowork.

Leonard Richardson. 2007. Beautiful Soup Documentation.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language Models Can Teach Themselves to Use Tools. *arXiv e-prints*, arXiv:2302.04761.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *arXiv e-prints*, arXiv:1706.03762.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *arXiv e-prints*, arXiv:2201.11903.

Wes McKinney. 2010. Data Structures for Statistical Computing in Python. In *Proceedings of the 9th Python in Science Conference*, pages 56 – 61.

Robert F. Wilson, Thomas Barclay, Brian P. Powell, Joshua Schlieder, Christina Hedges, Benjamin T. Montet, Elisa Quintana, Iain Mcdonald, Matthew T. Penny, Néstor Espinoza, and Eamonn Kerins. 2023. Transiting Exoplanet Yields for the Roman Galactic Bulge Time Domain Survey Predicted from Pixel-level Simulations. , 269(1):5.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. ReAct: Synergizing Reasoning and Acting in Language Models. *arXiv e-prints*, arXiv:2210.03629.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.