LAW: Legal Agentic Workflows for Custody and Fund Services Contracts

William Watson*, **Nicole Cho***,
Nishan Srishankar*, **Zhen Zeng**, **Lucas Cecchi**, **Daniel Scott**,
Suchetha Siddagangappa, **Rachneet Kaur**, **Tucker Balch**, **Manuela Veloso**
J.P. Morgan AI Research
New York, New York, USA
nicole.cho@jpmorgan.com

Abstract

Legal contracts in the custody and fund services domain govern critical aspects such as key provider responsibilities, fee schedules, and indemnification rights. However, it is challenging for an off-the-shelf Large Language Model (LLM) to ingest these contracts due to the lengthy unstructured streams of text, limited LLM context windows, and complex legal jargon. To address these challenges, we introduce LAW (Legal Agentic Workflows for Custody and Fund Services Contracts). LAW features a modular design that responds to user queries by orchestrating a suite of domain-specific tools and text agents. Our experiments demonstrate that LAW, by integrating multiple specialized agents and tools, significantly outperforms the baseline. LAW excels particularly in complex tasks such as calculating a contract's termination date, surpassing the baseline by 92.9% points. Furthermore, LAW offers a cost-effective alternative to traditional fine-tuned legal LLMs by leveraging reusable, domain-specific tools.

1 Introduction

While the advancement of Large Language Models (LLMs) demonstrates great potential for a myriad of use-cases in Document AI and Natural Language Processing (NLP) (Minaee et al., 2024), the domain of legal contracts poses unique challenges. The necessity for models to comprehend long, multi-document context windows and dense legal jargon engenders the intellectual pursuit to construct a legal domain-specific LLM. Certain studies have empirically investigated this motivation such as comparing the zero-shot performance of general-purpose LLMs on legal texts (Jayakumar et al., 2023) or fine-tuning LLMs under the Federated-Learning setting (Yue et al., 2024). Similarly, Colombo et al. (2024) trained SaulLM-7B on an English legal corpus, leveraging the Mistral-7B architecture (Jiang et al., 2023). While these developments are promising, legal contracts are highly varied not only in terms of semantics but also accessibility. Therefore, compared to the computational cost, the usage of a fine-tuned legal LLM can be very limited in practice (Figure 1). Thus, we propose LAW, a legal agentic workflow framework, that uses a code generation agent to orchestrate reusable tools, that can be leveraged for a variety of different contracts. Moreover, our framework can be generalized across different types of queries. Instead of relying solely on a fine-tuned LLM to solve highly complex tasks, LAW leverages a suite of specialized legal domain-specific tools, and a robust orchestration framework built on top of the FlowMind framework proposed by Zeng et al. (2023). LAW's reusable tools are designed to tackle distinct tasks such as contract retrieval. Our tools are rigorously guardrailed through unit tests that map their failure modes, ensuring a comprehensive understanding of their operational limits. By utilizing this method, LAW focuses on selectively applying the appropriate tools and text agents for a task, thereby optimizing the problem-solving process and delivering accurate and reliable responses.

Contributions We empirically prove the optimized performance of LAW for complex legal tasks. Overall, our contributions are three-fold:

- We propose LAW, a novel approach to interacting with financial-legal contracts, utilizing reusable legal domain-specific tools and text agents that addresses practical constraints - specifically legal dataset accessibility, scalability, and cost. Our system that can allow both lay-people, and domain experts to query information from complicated legal documents.
- LAW significantly outperforms the baseline, achieving up to 92.9% accuracy gains across a range of queries, from direct retrieval to multi-

*Equal Contribution

Proceedings of the 31st International Conference on Computational Linguistics: Industry Track, pages 583–594
January 19–24, 2025. ©2025 Association for Computational Linguistics

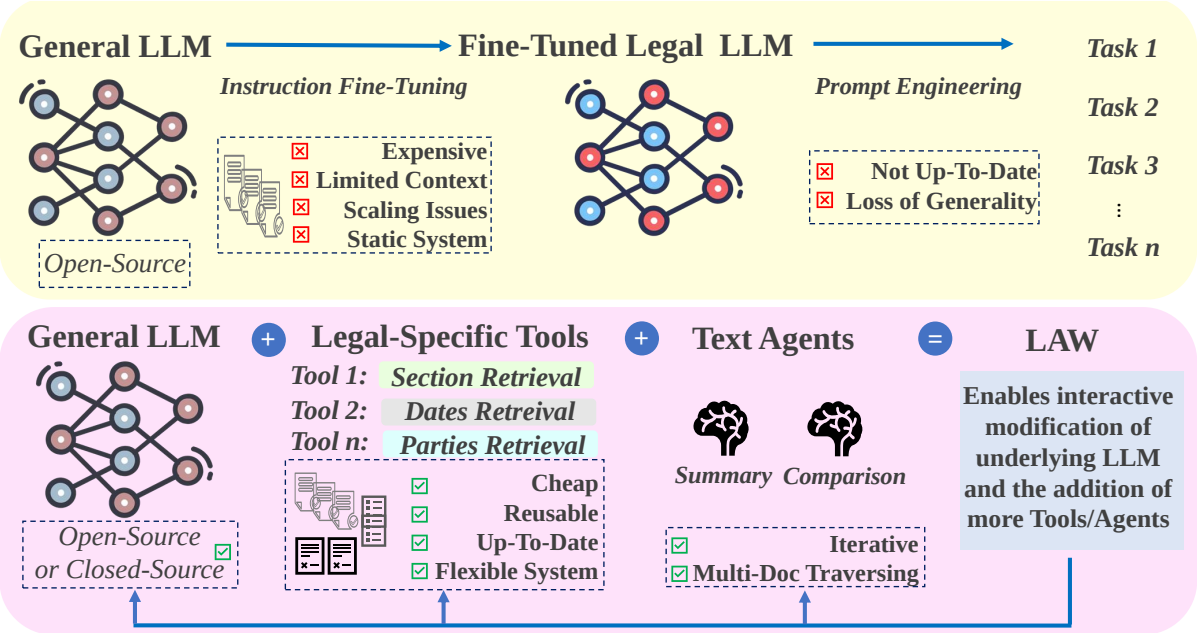

Figure 1: Comparing the traditional method of fine-tuning a legal LLM vs. LAW (Legal Agentic Workflows). Fine-tuning involves labeling contracts for a highly customized pipeline supported by open-source LLMs, which results in limited context, scale, or flexibility where coaxing additional information out of the model would require further tuning. The ensuing prompt engineering on the fine-tuned legal LLM also exacerbates the model's loss of generality. In contrast, LAW can operate on both closed-source or open-source LLMs and is equipped with legal domain-specific tools. These tools are cheaper to construct, reusable, simpler to construct, and incorporates recent data. The general LLM's orchestration of these tools along with the text agents engenders LAW, a highly interactive agentic system that also enables the addition of more tools and agents.

hop reasoning.

- LAW is the first legal agentic workflow system encompassing 23 years of regulatory contracts for the entire scope of public funds pursuant to the Investment Company Act of 1940. LAW can perform retrieval and analytical tasks that require an understanding over multiple documents, and each document contains many pages.

2 Related Works

LLMs in NLP LLMs such as Llama 2 (Touvron et al., 2023), PaLM (Chowdhery et al., 2023), GPT-3 (Brown et al., 2020), GPT-4 (Achiam et al., 2023), and Vicuna-13B (Chiang et al., 2023), have revolutionized NLP in many aspects. Their capabilities provide a foundation for more specialized adaptations, such as InstructGPT (Ouyang et al., 2022), which demonstrates how fine-tuning GPT models with human feedback can significantly enhance their alignment with user intent. Despite their impressive capabilities, LLMs face challenges like hallucination, outdated knowledge, and untraceable reasoning processes. Retrieval-Augmented Generation (RAG) (Lewis et al., 2020; Siriwardhana et al., 2023; Lin et al., 2023; Gao et al., 2023b) has emerged as a promising solution by effectively merging LLMs' intrinsic knowledge with external databases, enhancing both accuracy and reliability of generated content for knowledge-intensive tasks.

Domain-Specific Tools FlowMind (Zeng et al., 2023) introduces a generic prompt recipe that employs reliable Application Programming Interfaces (APIs) to ground LLM reasoning through the usage of tools. Additionally, HiddenTables (Watson et al., 2023) constructed an agentic system designed to enhance interactions with tabular data. Chen et al. (2023) and Gao et al. (2023a) explored how models can generate not only coherent text but also executable code snippets based on user queries. ToolFormer (Schick et al., 2023) and REACT (Yao et al., 2023), which are designed to enhance the model's interaction with external databases and software, helped LLMs access a wider range of resources, improving their ability to answer queries that required specialized knowledge. Watson and Liu (2021) demonstrated an end-to-end pipeline for financial extraction and transcription of tabular content from images. Furthermore, adaptations in Text-to-SQL (Rajkumar et al., 2022) methods for transforming natural language queries into database-readable commands show promise in streamlining document analysis tasks. CodeAct (Wang et al., 2024) demonstrated that executable Python code

can unify LLM agents' actions in a single action space, allowing for dynamic adjustments and new policies based on multi-turn interactions. Furthermore, LLMs in financial intelligence has evolved from traditional knowledge-graph and database approaches to domain-specific LLMs, though these face challenges with costs and accuracy, motivating the development of more sophisticated architectures (Watson and Liu, 2021; Watson et al., 2024; Cho et al., 2024).

Legal LLMs `SaulLM-7B` (Colombo et al., 2024), based on `Mistral-7B`, is specifically designed for legal text comprehension and generation. Trained on an extensive English legal corpus, it shows state-of-the-art capabilities in processing legal documents using instruction fine-tuning. Jayakumar et al. (2023) explored the zero-shot capabilities of general-purpose LLMs such as ChatGPT-3.5, LLaMA2-70b, and Falcon-180B on contract provision classification, noting their lower F1 scores compared to smaller, legal-specific fine-tuned models. Yue et al. (2024) presents FedJudge, the inaugural Federated Legal LLM framework, optimizing performance with minimal parameter updates during federated learning. Additionally, Fei et al. (2024) introduces InternLM-Law, tailored for diverse legal inquiries related to Chinese laws. Trautmann et al. (2022) assesses zero-shot Legal Prompt Engineering (LPE) for processing complex legal documents in multiple languages, focusing on legal judgment prediction tasks. Finally, Roegiest et al. (2023) examines the potential of LLMs to generate structured answers to legal questions, specifically in multiple-choice formats.

Evaluation Frameworks in Legal Environments Chen et al. (2021) and Nye et al. (2021) provide insight into the performance of LLMs in executing complex tasks. Their methodologies for assessing the accuracy and transparency of model outputs could be vital for deploying LLMs in legal settings where precision and accountability are crucial. Moreover, Liang et al. (2023) offers frameworks for ensuring that LLM operations adhere to legal and ethical standards. While the existing literature underscores significant advancements in legal LLM applications, `LAW`'s modular design employs an orchestrator agent integrating reusable tools for legal domain-specific tasks, marking a significant evolution from previous models.

3 Data Sourcing & Ingestion

EDGAR For our dataset, we procure contracts from EDGAR (Electronic Data Gathering, Analysis, and Retrieval), the U.S. SEC's (Securities and Exchange Commission)[1] database of regulatory filings. 23 years of filings are available in the omnibus filing 485BPOS which houses 2.7 million exhibits. From these, we procure 17,831 legal contracts (Appendix A).

Form 485BPOS Form 485BPOS is a post-effective amendment filed by all investment companies governed by the US Investment Company Act of 1940. These investment companies, colloquially dubbed *'40Act funds*, are mandated to file Form N-1A or Form 485BPOS, pursuant to Securities Act Rule 485(b) (U.S. Securities and Exchange Commission, 1984). We choose the legal contracts housed in Form 485BPOS omnibus filing as they capture the entire universe of all '40Act funds and account for a non-trivial (14%) of EDGAR filings.

Ingesting Contracts Contracts are difficult to directly ingest due to inconsistent reporting in EDGAR. Moreover, the SEC only allows a maximum throughput of 10 reports/second - this limitation necessitates the need to bring our data on-premise as EDGAR is not accessible at scale. In summary, we ingest a total of 22 GB of data on-premise within our knowledge base through a myriad of techniques such as:

- **Scalable Procurement:** We ingest at a rate of 112 documents/second - 6.7 hours were spent in terms of sequential processing. These contracts are not individually searchable on EDGAR; our knowledge base enables individual search.
- **AI Metadata Tagging and Search:** Each section is made searchable via title recognition algorithms, alongside contextual and visual cues to intelligently chunk each contract for precise retrieval within our distributed hybrid search.

4 Tools

We develop legal domain-specific tools that each undertakes a specialized task. These tools enable re-usability across varying contracts in our dataset; moreover, additional tools can be added at any stage of `LAW`'s development.

[1] https://www.sec.gov/

4.1 Tools for Direct Extraction

Tool for Extracting Dates Contracts house different types of dates such as the contract's `Effective Date` (when the current contract is effective), `Master Date` (when the master/original contract was effective), and `Dated Date` (when the current contract was signed). Our tool distinguishes these three types. Detection and extraction of dates is achieved via RoBERTa span detection (Liu et al., 2019), HTML parsing with BeautifulSoup4[2], and regular expression heuristics. Then, our tool standardizes all extracted dates to the `DD/MM/YYYY` format.

Tool for Extracting Parties This tool's objective is to find and extract the associated parties involved in signing the contract. These include the trust of funds and the custodian bank. Filing 485BPOS in EDGAR contains metadata about a subset of the involved parties for certain contracts. This is because the filing metadata only pertains to the specific legal entity making the EDGAR submission, rather than encompassing the full breadth of an investment manager's fund offerings and related parties. We use this as a guide to train our system's understanding of the full scope of contracts. We implement fuzzy matching to search for these parties in the contracts. The custom fuzzy matching built on top of RapidFuzz[3] aims to mitigate issues that may arise from stylistic differences in names such as special character usage, capitalization, and differing naming conventions. Additionally, we mitigate issues with some names being substrings of others by searching sequentially in order of increasing name length and removing found parties.

4.2 Tools for Multi-Hop Reasoning

Tool to Calculate Contract Lifecycle Contracts typically have a lifecycle during which the provisions are enforced. This tool aims to calculate the termination date of the contract's lifecycle. It uses our existing tool for dates to extract the effective date of the contracts. Next, it searches for the contract's duration or the termination date. If the contract mentions the duration (e.g. `3 years`), the tool translates the text into a numerical value. Finally, this numerical value is added to the effective date to generate a termination date.

Tool to Retrieve Master Contract This tool's goal is to differentiate between a `master` and an `amendment` agreement. Master agreements refer to the original contract that outlines all aspects of the relationship between the fund and the custodian bank. An amendment refers to contracts that amend the master or any subsequent amendments - amendments are typically less detailed as they can amend a single word. Our tool classifies and retrieves the master contract by comparing the extracted *effective date* with the *master effective date* using the tool for dates; if equal, the contract is considered to be the master. If determined to be an amendment, the tool searches for the master by matching the dates and parties.

Tool to Label Section Titles Contracts are semantically structured and hierarchical in the construction of their clauses. Each clause holds detailed knowledge regarding terminology such as indemnification, force majure, and termination. Therefore, when queries about particular terms arise, directly retrieving the relevant clause or section is far more efficient than reviewing the entire contract indiscriminately. However, parsing contracts into distinct semantic sections for effective retrieval presents significant challenges. Many contracts lack explicit section declarations and the language across different sections can be highly similar. To address this challenge, we employed a fine-tuned `t5-large` (Raffel et al., 2020) model trained to classify paragraphs into one of 20 potential section labels. These labels cover a broad spectrum of typical clauses found in contracts (Appendix B). Our training dataset is comprised of 1,500 paragraphs per title, systematically collected from a variety of contracts to ensure diverse linguistic representations. As an alternative solution, we trained a `t5-base` model for title generation instead of classification. Both section title classification and generation models perform similarly. (Appendix F) outlines the performance and training parameters. Section titles are then used for section search and retrieval as described in §6.

5 Text Agents

Summary Agent The summary agent aims to provide a useful summary of legal clauses. Our prompts enable the agent to focus on identifying and preserving key terms such as entity names and dates. A key challenge in summarizing relates to sections that exceed an LLM's context window.

[2] https://www.crummy.com/software/BeautifulSoup/

[3] https://github.com/rapidfuzz/RapidFuzz

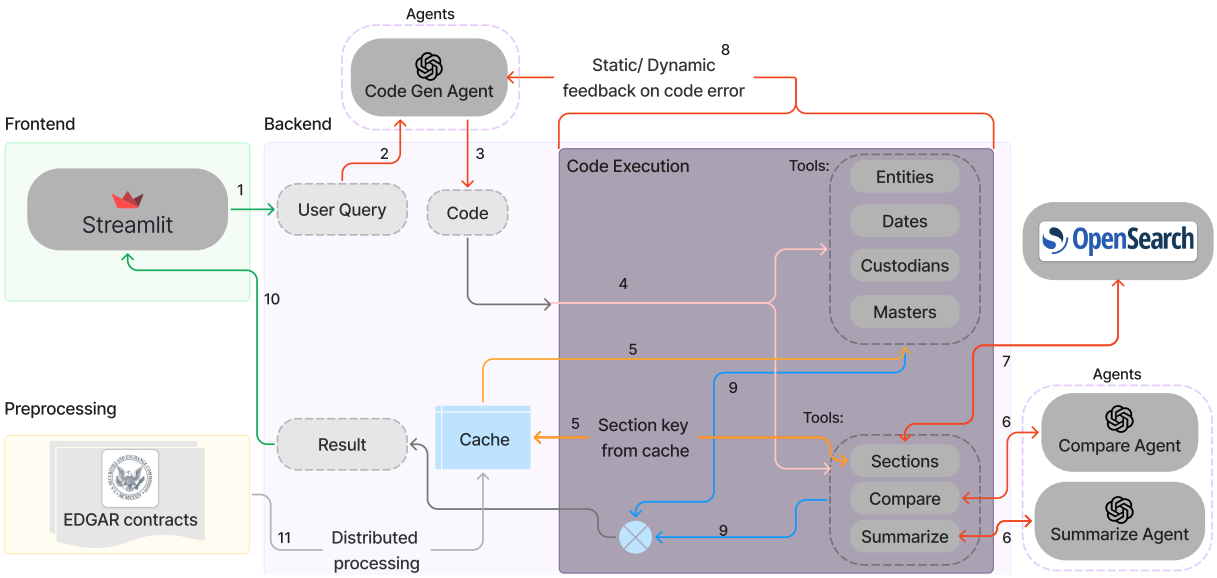

Figure 2: System Overview of **LAW**. **(1)** User query input on front-end (§6); **(2)** Query manipulation and custom modification added to prompt and sent to the code generation agent (§6); **(3)** Chat completion return from the code generation agent; **(4)** Execution of backend API tools (§4); **(5)** Tool retrieval of information from internal cache (§6); **(6)** Calls to text agents (§5); **(7)** Calls to multi-node OpenSearch cluster for text retrieval (§6); **(8)** Feedback on code runs back to the code generation agent in case of failure; **(9)** Concatenation of final output; **(10)** Final text output is rendered on the UI; **(11)** EDGAR contracts undergo continuous, offline, distributed processing to update our internal cache and OpenSearch systems (§3).

Legal contracts can be very long, averaging around 27K $\pm$ 51K tokens when encoded by `tiktoken`. Therefore, if the input text and prompt exceed the 16K token limit of `gpt-3.5-turbo`, the text is split into 8K token chunks. These chunks are processed in parallel by seperate sub-agents, with the output concatenated into a final summary.

Comparison Agent The comparison agent's purpose is to understand how particular clauses are different across time or entities. With a similar base prompt as the summary agent, it compares two bodies of text. The agent chronologically sorts the sections from different contracts and, in parallel, compares each pairwise set of sections in the list. For example, given a list of contracts' sections $\mathcal{L} = [s_0, s_1, ..., s_n]$, where s_i is an individual section, it performs $compare(s_i, s_{i+1}) \, \forall \, s_i \in \mathcal{L}, i \neq n$. To handle large bodies of text, the agent also performs repeated summaries on each section s_i to condense the body to a manageable size. The summarized sections are then passed to the comparison agent.

6 Engineering Infrastructure

The system overview for `LAW` is illustrated in Figure 2. In addition to the tools described previously, it also uses the following modules:

User Query We compose user queries with two parts : (1) the **entity** of interest; and (2) the **task** to be executed. The base entities are *Fund* ***X***, *Trust* ***X***, and *Custodian* ***X***. We also combine the base entities, e.g. *Fund* ***X*** and *Custodian* ***Y*** to find the contract for this particular relationship. The possible tasks to apply on the entities include: **(1)** Explore all contracts; **(2)** Find `{master agreements, master dates, termination dates, parties, clause X}`; **(3)** `{Summarize, Compare}` clause X. These tasks are motivated by legal use cases.

Caching To reduce runtime latency, our system batch pre-processes data extraction. This includes features related to the involved parties or dates. The extracted data is stored in a CSV file on the backend disk, acting as a cache. This cached data helps avoid latency especially for multi-step reasoning.

Section Search The large volume of legal text cannot be directly stored in our cache when retrieving contract sections. Instead, our contracts are segmented and indexed in an OpenSearch distributed datastore provided by AWS. For each contract, we retrieve the top 20 most relevant sections using the BM25 ranking algorithm. The ranking algorithm looks at the presence of the target clause in both the section texts as well as its indexed title (§4.2).

Code Generation Agent Our system prompts the agent to generate Python code that can resolve the user's query (§6). The prompt includes tool names, descriptions, and examples similar

User Query	LAW	Baseline
Retrieval	*Hit Rate*	
Explore all contracts	94.4	71.8
Find master agreements	100.0	65.4
Find master dates	93.3	36.2
Find termination dates	95.4	2.5
Find parties	100.0	16.3
Analytical	*BERTScore F1*	
Summarize clause X	89.5	68.1
Compare clause X	71.9	-

Table 1: A comparison of LAW with a simulated `gpt-3.5-turbo` baseline. For retrieval-type queries we measure the hit rate/recall calculating the percentage of correct retrievals compared to the ground truth. For analytical-type questions, we measure text similarity using BERTScore's (Zhang* et al., 2020) F1 metric. The contextual embeddings for BERTScore are obtained using the `bert-large-uncased` model.

to Chain-of-Thought (Wei et al., 2022) and Self-Refine (Madaan et al., 2023). The prompt specifies instruction preferences, such as outputs to display for particular tools, and execution preferences such as not printing outputs or leaving incomplete `todo` tags. LAW employs generated a three-tier system to generate and validate code:

- Syntax Validation: Performs pre-execution checks to verify code syntax, types, and security constraints.
- Hallucination Detection: Ensures generated code only calls tools that exist in LAW's toolset with valid parameter signatures.
- Runtime Validation: Implements specialized error handling that captures and categorizes execution failures for targeted remediation.

This verification framework enables LAW's orchestrator to maintain a feedback loop, providing specific correction suggestions to the code generation agent when errors occur.

7 Experiments

Dataset Curation We labeled a dataset of 720 user queries as described in §6. The tasks can be divided into two types: *retrieval* and *analytical*. *Retrieval* queries correspond to retrieving information about entities of interest from contracts. Queries that involve the exploration of all contracts, the extraction of dates, and parties fall into this category. *Analytical* queries require deeper insight, going beyond what can be extracted directly in the contracts. Queries that involve summarizing or comparing clauses across different contracts pertain to this category. We generated 20 queries for each combination of task and entity for *retrieval* queries and 10 for *analytical* queries. We randomly populated the entities of interest from the universe of '40Act funds. The ground truth answers are generated using hand-coded scripts that leverage the same tools and text agents that the proposed system has access to. This procedure makes the evaluation agnostic to the implementation of the tools and focuses exclusively on LAW's ability to generate code that correctly orchestrates the tools and the text agents.

Baseline setup Our baseline seeks to understand if `gpt-3.5-turbo`, as is, can be prompted to answer queries on contracts. For `Explore all contracts` and `Find master agreement` queries, we simulate a noisy RAG framework by providing a set of four correct contracts and four distractor contracts. We reformulate user queries into a set of sequential True/False scenarios where the goal of the baseline is to determine if the candidate contract is associated with the entity, or is a master agreement, respectively. This choice was implemented as most contracts exceed the context limit of `gpt-3.5-turbo`. For other queries, we choose four relevant contracts and prompt `gpt-3.5-turbo` to extract the desired pieces of information. In essence, we narrowed the search space and provided relevant context for the baseline, where the provided context is sufficient for answering the queries. The context limit adds significant constraint on being able to provide in-context examples as demonstrations.

Results Table 1 compares LAW against the baseline. LAW shows remarkable performance across *retrieval* to *analytical* queries. Among *retrieval* queries, for a true-false formulation of `Explore all contracts`, the baseline performs reasonably at 71.8% compared to 94.4% achieved by LAW. This similar performance is seen for `Find master agreements`. The baseline starts performing poorly at 36.2% when asked to lift the master date in contracts with a variety of dates. Moreover, the baseline quickly deteriorates for queries that require multi-hop reasoning, such as `Find termination dates`, where LAW surpasses the baseline by 92.9%. We observe that the baseline tends to hallucinate immensely, showing a near-compulsion to conjure fictional dates. Specifically, this operation depends on the LLM finding the term for the duration of the contract and adding it to the

master's effective date. Finally, when asked to extract parties, the baseline often uses the question as an entity hint, but fails at lifting the complete list of entities from a dense agreement. For *analytical* queries, the baseline underperforms on clause summarization because of an inability to understand which sections are relevant to a given clause. `Compare clause` requires understanding the trend of how a clause changed across multiple contracts. This capability of comparing clauses cannot be performed using the baseline setup, as an agentic workflow with a larger context length is required.

8 Conclusion

We present `LAW`, a novel legal agentic workflow, achieving the successful completion of complex legal tasks. In contrast to fine-tuning an open-source LLM, our agentic invention is applicable for both closed and open-source models, leverages legal domain-specific tools and text agents that are modifiable and reusable, and orchestrates comprehensive plans. `LAW` has achieved remarkable performance, demonstrating robustness across *retrieval* and *analytical* queries and out-performing the baseline. Thus, our framework successfully enables automated workflows for varying contracts that govern the critical custody business. Future work can focus on applying `LAW` to non-English contracts, and explore additional agents grounded in other specific domains.

Disclaimer

This paper was prepared for informational purposes by the Artificial Intelligence Research group of JPMorgan Chase & Co. and its affiliates ("JPMorgan") and is not a product of the Research Department of JPMorgan. JPMorgan makes no representation and warranty whatsoever and disclaims all liability, for the completeness, accuracy or reliability of the information contained herein. This document is not intended as investment research or investment advice, or a recommendation, offer or solicitation for the purchase or sale of any security, financial instrument, financial product or service, or to be used in any way for evaluating the merits of participating in any transaction, and shall not constitute a solicitation under any jurisdiction or to any person, if such solicitation under such jurisdiction or to such person would be unlawful.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. arXiv preprint arXiv:2303.08774.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. Advances in neural information processing systems, 33:1877–1901.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code. Preprint, arXiv:2107.03374.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. Preprint, arXiv:2211.12588.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Nicole Cho, Nishan Srishankar, Lucas Cecchi, and William Watson. 2024. Fishnet: Financial intelligence from sub-querying, harmonizing, neural-conditioning, expert swarms, and task planning. In Proceedings of the 5th ACM International Conference on AI in Finance, ICAIF '24, page 591–599, New York, NY, USA. Association for Computing Machinery.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. Journal of Machine Learning Research, 24(240):1–113.

Pierre Colombo, Telmo Pessoa Pires, Malik Boudiaf, Dominic Culver, Rui Melo, Caio Corro, Andre F. T. Martins, Fabrizio Esposito, Vera Lúcia Raposo, Sofia Morgado, and Michael Desa. 2024. Saullm-7b: A pioneering large language model for law. Preprint, arXiv:2403.03883.

Zhiwei Fei, Songyang Zhang, Xiaoyu Shen, Dawei Zhu, Xiao Wang, Maosong Cao, Fengzhe Zhou, Yining Li, Wenwei Zhang, Dahua Lin, Kai Chen, and Jidong Ge. 2024. Internlm-law: An open source chinese legal large language model. Preprint, arXiv:2406.14887.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023a. Pal: Program-aided language models. Preprint, arXiv:2211.10435.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023b. Retrieval-augmented generation for large language models: A survey. arXiv preprint arXiv:2312.10997.

Thanmay Jayakumar, Fauzan Farooqui, and Luqman Farooqui. 2023. Large language models are legal but they are not: Making the case for a powerful Legal-LLM. In Proceedings of the Natural Legal Language Processing Workshop 2023, pages 223–229, Singapore. Association for Computational Linguistics.

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.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in Neural Information Processing Systems, 33:9459–9474.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. 2023. Code as policies: Language model programs for embodied control. Preprint, arXiv:2209.07753.

Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Rich James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, et al. 2023. Ra-dit: Retrieval-augmented dual instruction tuning. arXiv preprint arXiv:2310.01352.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. Preprint, arXiv:1907.11692.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In Thirty-seventh Conference on Neural Information Processing Systems.

Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2024. Large language models: A survey. Preprint, arXiv:2402.06196.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. 2021. Show your work: Scratchpads for intermediate computation with language models. Preprint, arXiv:2112.00114.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. Preprint, arXiv:2203.02155.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research, 21(140):1–67.

Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. 2022. Evaluating the text-to-sql capabilities of large language models. Preprint, arXiv:2204.00498.

Adam Roegiest, Radha Chitta, Jonathan Donnelly, Maya Lash, Alexandra Vtyurina, and Francois Longtin. 2023. Questions about contracts: Prompt templates for structured answer generation. In Proceedings of the Natural Legal Language Processing Workshop 2023, pages 62–72, Singapore. Association for Computational Linguistics.

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. Preprint, arXiv:2302.04761.

Shamane Siriwardhana, Rivindu Weerasekera, Elliott Wen, Tharindu Kaluarachchi, Rajib Rana, and Suranga Nanayakkara. 2023. Improving the domain adaptation of retrieval augmented generation (rag) models for open domain question answering. Transactions of the Association for Computational Linguistics, 11:1–17.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. Preprint, arXiv:2307.09288.

Dietrich Trautmann, Alina Petrova, and Frank Schilder. 2022. Legal prompt engineering for multilingual legal judgement prediction. Preprint, arXiv:2212.02199.

U.S. Securities and Exchange Commission. 1984. Electronic Data Gathering, Analysis, and Retrieval (EDGAR) System. https://www.sec.gov/edgar.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. 2024. Executable code actions elicit better llm agents. Preprint, arXiv:2402.01030.

William Watson, Nicole Cho, Tucker Balch, and Manuela Veloso. 2023. HiddenTables and PyQTax: A cooperative game and dataset for TableQA to ensure scale and data privacy across a myriad of taxonomies. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pages 7144–7159, Singapore. Association for Computational Linguistics.

William Watson, Nicole Cho, and Nishan Srishankar. 2024. Is there no such thing as a bad question? h4r: Hallucibot for ratiocination, rewriting, ranking, and routing. Preprint, arXiv:2404.12535.

William Watson and Bo Liu. 2021. Financial table extraction in image documents. In Proceedings of the First ACM International Conference on AI in Finance, ICAIF '20, New York, NY, USA. Association for Computing Machinery.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. Advances in neural information processing systems, 35:24824–24837.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. Preprint, arXiv:2210.03629.

Linan Yue, Qi Liu, Yichao Du, Weibo Gao, Ye Liu, and Fangzhou Yao. 2024. Fedjudge: Federated legal large language model. Preprint, arXiv:2309.08173.

Zhen Zeng, William Watson, Nicole Cho, Saba Rahimi, Shayleen Reynolds, Tucker Balch, and Manuela Veloso. 2023. Flowmind: Automatic workflow generation with llms. In Proceedings of the Fourth ACM International Conference on AI in Finance, ICAIF '23, page 73–81, New York, NY, USA. Association for Computing Machinery.

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In International Conference on Learning Representations.

A Domain Details

A.1 Form 485BPOS Regulatory Context

Form 485BPOS is a document in the investment company industry, specifically used by mutual funds and other registered investment companies. It is filed with the U.S. Securities and Exchange Commission (SEC) under the Investment Company Act of 1940, often referred to as the '40 Act. It serves several key functions:

1. **Registration**: It is used to register new mutual funds or update existing registrations.
2. **Prospectus Updates**: '40Act Funds use Form 485BPOS to file updated prospectuses, which contain essential information for investors about the fund's investment objectives, risks, performance, and fees.
3. **Regulatory Compliance**: It ensures that funds comply with SEC disclosure requirements and regulations subject under the Investment Company Act of 1940.

Filing 485BPOS transcribes routine annual updates and other changes that become effective immediately upon filing. Currently, Form 485BPOS is 1.2% of the total available EDGAR filings[4].

A.2 Motivation and Impact

The contracts analyzed by LAW govern critical relationships between custodian banks and '40Act funds, which are investment vehicles for trillions of retail investors, especially for retirement income. Proper governance of key clauses is paramount for the health of these funds, custodian banks, and the broader financial ecosystem. Traditionally, an immeasurable amount of time has been spent on finding, analyzing, and comparing key clauses in these contracts.

LAW shows potential for application to other contract types and legal documents. It also holds relevance for non-U.S. funds (e.g., USCITs) operating under similar regulations, demonstrating the broad applicability of our approach.

A.3 Dataset Examples

485BPOS Contracts in EDGAR are highly variable in length, format, content, and type. These include master agreements, amendments, separate appendixes, or the list of funds or entities that are captured by an agreement. See below for several example filings used in our dataset.

- **Full Contract:** https://www.sec.gov/Archives/edgar/data/1831313/000182912624004293/tcwetftrust_exg2.htm
- **Single Amended:** https://www.sec.gov/Archives/edgar/data/1879238/000182912623004720/bondbloxxetf_exg4.htm
- **13th Amendment**: https://www.sec.gov/Archives/edgar/data/1592900/000182912623003816/easeriestrust_ex99g1xiv.htm
- **List of Funds:** https://www.sec.gov/Archives/edgar/data/837274/000119312507144235/dex99gx.htm

A.4 Domain Complexity

Analyzing lengthy legal contracts is difficult as there are no obvious headings, a plethora of legal concepts that are exceedingly difficult to digest, and no obvious categorization of paragraphs. These highly unstructured and dense legal documents encompass and describe in minute detail with different nuances in different formats the following contractual principles - such as `standard of care regimes`, `gross negligence`, `fiduciary responsibilities`, `breach of contract`, `liability for direct damages`, etc. These principles are presented in dense legal language, unstructured streams of text in different formats. Therefore, retrieving accurate numerical or textual values and analyzing/comparing them across tens of thousands of documents is a highly complex task. Within the legal domain, these retrieval and analytical tasks constitute as one of the most sophisticated and time-consuming tasks, especially in a highly unstructured database such as EDGAR, where no structured labels exist for the contracts.

B List of Key Clauses

A full list of key clauses is found in Table 2. The term *Indemnification* refers to protective clauses that govern when losses occur with a third party. In the complex legal and financial landscape, recovery and punitive measures when accidents or losses occur is extremely important - for example, if a third party vendor's software breaks, is it the client or the provider's responsibility to recuperate those losses. Clauses governing *Force majeure* events are often related to indemnification clauses, which refer to

[4] https://www.sec.gov/about/dera_edgarfilingcounts

Section Titles
account transactions-
authorized persons
definitions
duties and responsibilities
evidence of authority
fee schedule
fees and expenses
foreign custodian and subcustodian
governing law
indemnification
instructions
limitations and scope of use or liability
miscellaneous
nominees
proprietary information
recitals
standard of care liabilities
subcustodians and securities depositories
successor custodian
termination

Table 2: List of key clauses found in fund custody contracts.

an event that is outside of a party's control and prevents them from fulfilling their obligations. *Termination* entails the date and conditions at which the contract/legal responsibility will end.

C System Design & Implementation

Framework Our system is based on FlowMind's (Zeng et al., 2023) framework and code recipe, extending it to a robust agentic legal framework with Fund Custody Services specific APIs. The code generation agent is responsible for mimicking planning by generating a series of function calls, akin to thinking steps, that breaks the question into steps semantically linked to our tool calls.

Tools and Agents LAW incorporates tools for direct extraction, multi-hop reasoning, and text analysis. By allowing LAW the flexibility to reuse statements, pass previous information from a function call directly into another, and iterate over retrieved items, it can go beyond single-step reasoning. Text-based agents employ zero-shot prompting. The summarize and compare tools utilize specialized text agents to yield useful analytics, enhancing the system's capability to handle complex queries. Specifically, the summary agent's task is to succinctly summarize a particular clause, restricting output to facts present in the text, such as preserving key terms, entities, dates, etc.

Long-Context Contracts To handle long clauses and contracts, our system breaks the text into smaller chunks. These are then processed by separate "sub-agent" spawns to summarize, after which the results are concatenated into a complete summary. This approach ensures comprehensive analysis of extensive legal texts that would not fit into a standard context window.

Framework Extensibility While LAW currently demonstrates strong performance with its existing toolset, extending the framework to new tasks requires careful consideration. Adding new tools involves:

- Task Analysis: Identifying atomic operations that can be modularized into reusable components.
- Tool Development: Creating focused tools with clear inputs/outputs and comprehensive unit tests.
- Integration Testing: Verifying the tool's interaction with the code generation agent and other components.

The modular nature of LAW allows new tools to be added without modifying existing components. However, several challenges we faced included:

- Ensuring tool reliability across diverse contract formats.
- Maintaining clear boundaries between tool responsibilities.
- Balancing tool specificity with reusability.
- Managing increased complexity in the orchestration logic.

D Experimental Details

D.1 Dataset Creation and Validation

We collaborated with business end-users to identify useful pieces of information to extract and sections/titles that they often examine. For validation, we conducted a small pilot study with human users on shorter contracts, obtaining performance comparable to that reported in the paper.

D.2 Evaluation

Our evaluation methodology included comparing outputs against hard-coded scripts as a ground truth. We simulated a noisy RAG framework by obtaining a small subset of relevant and irrelevant documents, mimicking a scaled version of the 485BPOS

dataset. In this framework, we evaluate the system's ability to ignore irrelevant contracts while effectively extracting information from the relevant ones. The experiments employed a few-shot prompting strategy.

D.3 Error Analysis

The main source of error in obtaining termination dates was the hallucination of non-existent dates, which particularly affected baseline performance. We noted that generating baseline output for clause changes across contracts was not feasible due to context limitations when communicating pairs of clauses.

E Potential LAW Queries

- Find the termination dates of all contracts from custodian Goldman Sachs.
- Find the master dates of contracts between Trust Investor Counselor Series Fund Inc and Custodian State Street Bank and Trust.
- Compare the fees and expenses clause of the previous contracts.

F Title and Section Retrieval Metrics

Attributes	T5-Base	T5-Large
Method	Generation	Classification
Parameters	222,903,552	737,668,096
Samples / Title	400	1500
Accuracy	42%	46%
GPU	NVIDIA T4	NVIDIA A10G
# of GPUs	4	4
GPU Memory	64 GiB	96 GiB

Table 3: Experiment and environment details for section title generation and classification.

The following are the training parameters for T5-Base and T5-Large models.

- Training epochs: 3
- Per device batch size for train and eval: 2
- Learning rate: 0.01

G Sample Generated Code

Query: Compare the *authorized persons* clauses for Fund *BNY Mellon International Equity Income Fund*.

Customized suffix: Only compare subsequent clauses of five sampled non-empty contract sections. Ensure that there are also contracts for this entity choice.

Code generated:

```
# Step 1: Retrieve all agreements for the
    specified fund.
agreements = get_agreements_for(funds='BNY
    Mellon International Equity Income Fund')

# Step 2: Check if any agreements were found.
if agreements:
    # Step 3: Extract the clauses for the
        specified section from each agreement.
    clauses, urls, _ =
        get_section_v2(agg_list=[agreement[1]
        for agreement in agreements],
        section_name='authorized persons')

    # Step 4: Check if any clauses were found.
    if clauses:
        non_empty_clauses = [clause for clause
            in clauses if clause != '']
        num_clauses = len(non_empty_clauses)
        if num_clauses > 5:
            interval = num_clauses // 5
            selected_clauses =
                non_empty_clauses[::interval]
        else:
            selected_clauses = non_empty_clauses

        # Step 5: Compare the selected clauses.
        output = get_comparison_v1(
        list_agreement_tuples=[(agreement[0],
            agreement[1]) for agreement in
            agreements],
            text_list=selected_clauses)
    else:
        output = "No 'authorized persons'
            clauses found for Fund 'BNY Mellon
            International Equity Income Fund'"
else:
    output = "No agreements found for Fund 'BNY
        Mellon International Equity Income
        Fund'"

# Step 6: Return the output.
output
```