MetaMesh

Multi-agent Collaboration for Meta-Prompting

by Tawab Safi

Background

- From healthcare to supply chain, multiple sectors reply on SOP (Standard Operating Procedures) and complex policy documents.
- Subject Matter Experts (SMEs) are currently needed to interpret and act on these documents – costly & time-consuming.

Opportunity for Automation:

- Emerging LLMs show promise for understanding and acting on large instruction sets (e.g. return policies, insurance clauses).
- A robust LLM agent could drastically cut operational costs, speed up workflows, and reduce human error.

Background

Existing Techniques:

- Retrieval-Augmented Generation (RAG) or fine-tuning requires curated “gold” datasets - often difficult or impossible to create.
- Prompting alone (e.g., Chain-of-Thought) doesn’t suffice, and manually creating specialized agents for each policy or procedure document isn’t scalable.

New Approaches:

- Research on MetaGPT and AutoAgent demonstrates static & dynamic multi-agent collaboration – primarily for code generation.
- Recent work on Meta-Prompting show that task-agnostic scaffolding can be used to guide LLMs in processing complex instructions – doesn’t consider multi-agent collaboration yet.

Context

CUAD Dataset

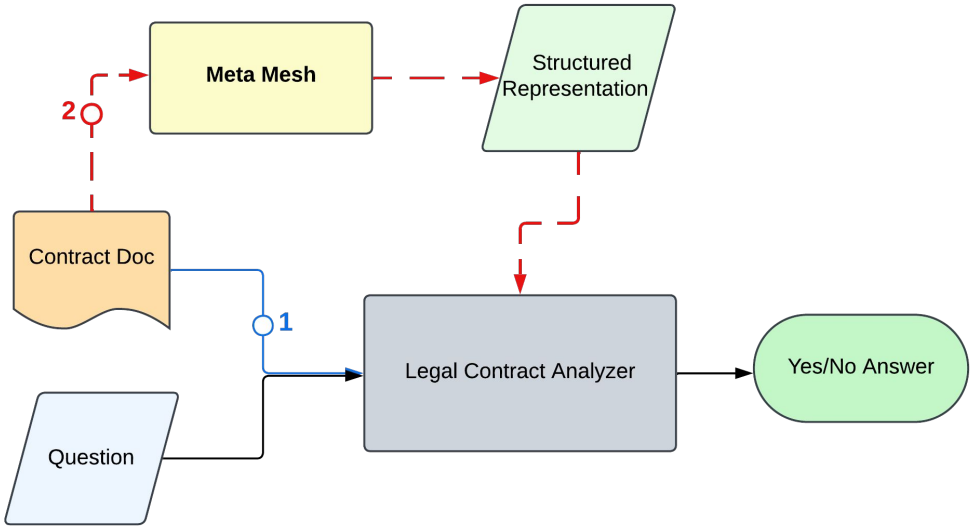
- ~510 real-world contracts, annotated by legal experts
- 41 clauses types identified, I focus on 31 yes/no questions (e.g. non-compete, exclusivity).
- Tests model understanding and interpretation of legal contracts.

Hypothesis:

1. Better structured intermediate representation → improved question-answering performance on legal documents.
2. Specialized Agent Personas focused on different aspects of a contract yield richer understanding of the underlying legal document.

Context

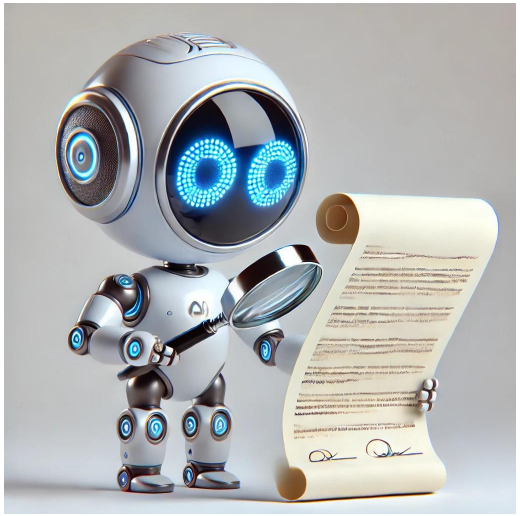
1. Can an agent better answer questions about a contract if provided in structured format?
2. Can we have multiple agents collaborate to create a complete and rich intermediate structured representation for a contract?



Main Components:

1. Legal Contract Analyzer
2. MetaMesh Framework

Legal Contract Analyzer



An agent that takes either a **contract** or **structured intermediate representation** of the contract as input, and can yes/no questions.

Inputs: (Contract **OR** Intermediate Rep) **AND** Question

Output: <str: yes or no, followed by brief description>

Models: gpt-4o **OR** gpt-4o-mini **OR** any local model through Ollama

Analytics Feature: Measures *processing time* and *token count* per contract and per question.

MetaMesh Framework

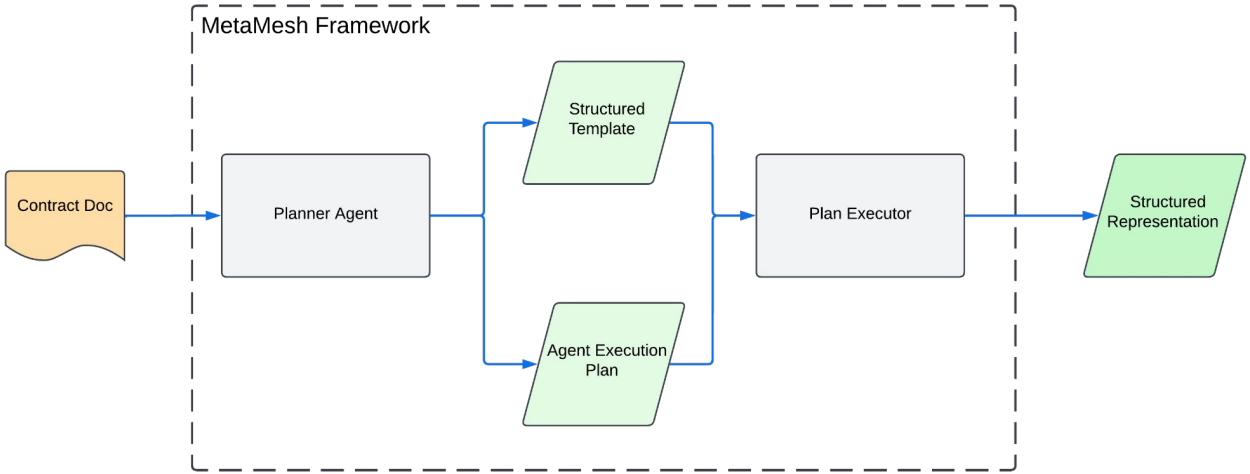
Key Components:

1. *Planner Agent*

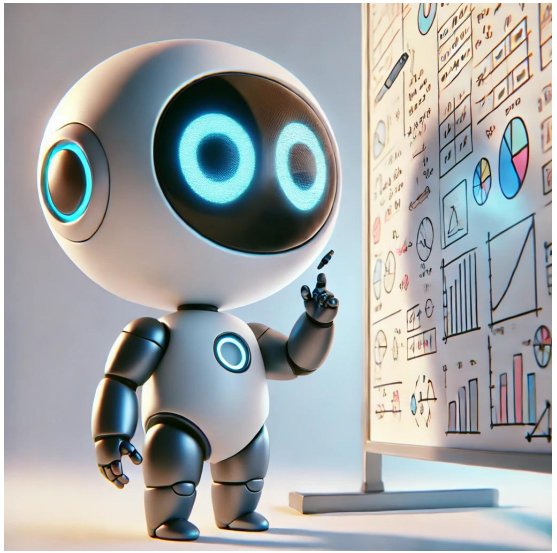
Creates the template and plans the agents to fill it in.

2. *Plan Executor*

Dynamic Multi-Agent environment that creates and runs the agents.



Planner Agent



Prompt Summary: Create a template that can be used to best capture all important information about the given contract and then design a specialized agent for each filling in each section of the template.

Models: o1-preview **OR** gpt-4o

Input: Contract Document

Output:

1. *Structured Template:*
    a. Sections: Category of data points
    b. Datapoints: individual unit of information
2. *Agent Execution Plan:* For each section in the template a specialized agent is created with a prompt defining its role and task.

Analytics Feature: Measures *processing time* and *token count* per contract.

Template and Plan Structure:

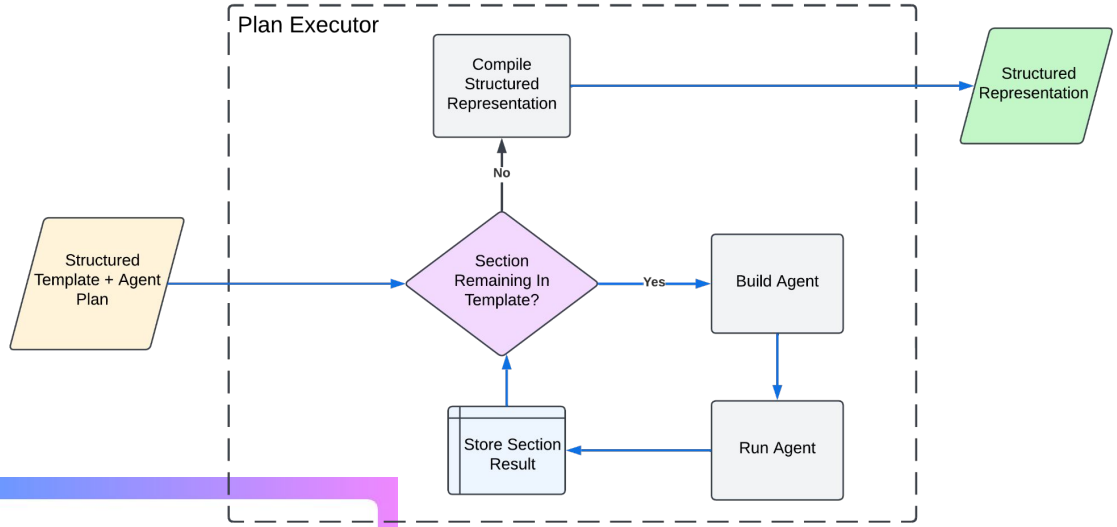
```python
class Plan(BaseModel):
    introduction: str
    sections: List[TemplateSection]

class TemplateSection(BaseModel):
    section_name: str
    section_description: str
    data_points: List[DataPoint]
    section_agent: AgentInfo

class AgentInfo(BaseModel):
    agent_name: str
    agent_instructions: str

class DataPoint(BaseModel):
    data_point_name: str
    data_point_description: str
    data_point_questions: List[str]
    data_point_instructions: str
```

Plan Executor

The executor loops through each section in the defined template, while creating an expert agent for each of them who is assigned with the task of extracting all the data points for the section.

Each expert agent files out a section in the final JSON Structured Representation created for the contract.

Analytics Feature: Measures *processing time* and *token count* for each agent.




```python
class StructuredRepresentation(BaseModel):
    introduction: str
    sections: List[Section]

class Section(BaseModel):
    section_name: str
    section_description: str
    data_points: List[DataPoint]

class DataPoint(BaseModel):
    data_point_name: str
    data_point_description: str
    data_point_questions: List[str]
    data_point_overview: str
    data_point_answers: str
    data_point_verbatim: str
```

Evaluation Setup

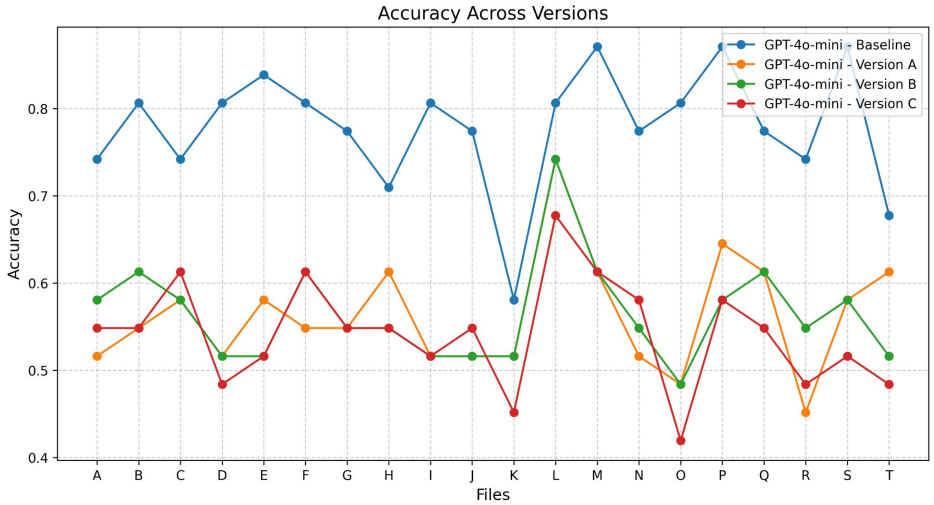
Complete Evaluation Dataset: ~510 contract files and 31 questions per file

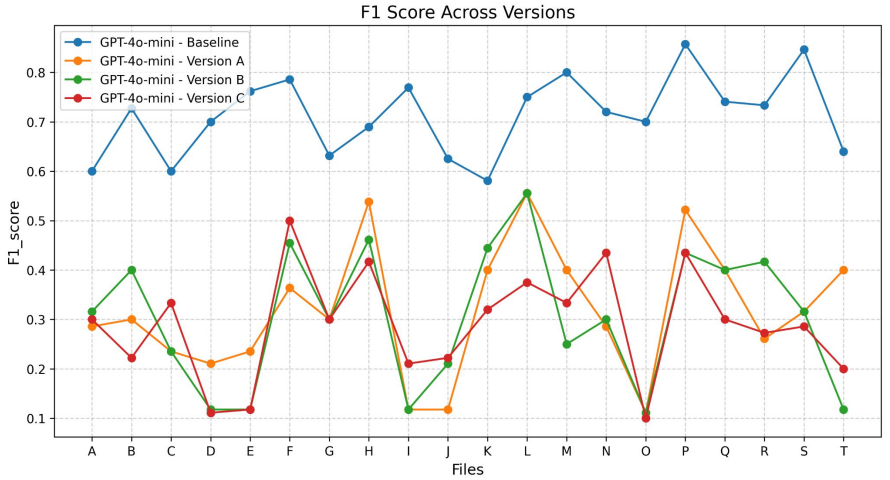
Constraint: Limited OpenAI API Budget + Compute Resources

Sampled a subset of 20 contracts for evaluation using the following steps:

- Used Legal Contract Analyzer agent with llama 2:13B model to evaluate the complete dataset for all questions. → **find contracts that relatively small agents don't perform well on.**
- Calculate f1, precision, recall, and accuracy per contract based on the predicted answers and use its weighted average to sort the files.
- Constraints for file selection:
  - Per file Yes distribution >= 35% and No distribution <= 65%
  - Select 5 worse performance contracts (lowest weighted perf score that meets the dist criteria)
  - Select 15 average performance contracts (closer to 50% mark that meets the dist criteria)

Evaluation Result



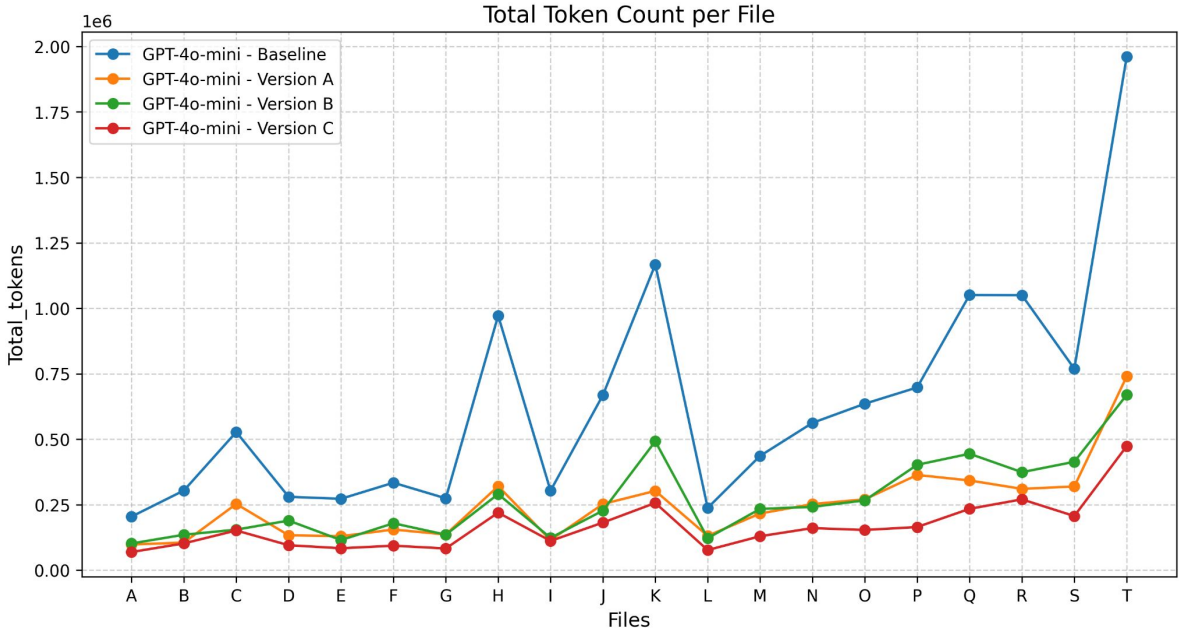


We used gpt-4o-mini based Contract Analyzer Agent, where each version represents its input:

- **Baseline:** Contract Doc
- **Version A:** Intermediate Rep created by: Planner → o1-preview & PlanExecutor → gpt-4o
- **Version B:** Intermediate Rep created by: Planner → o1-preview & PlanExecutor → gpt-4o-mini
- **Version C:** Intermediate Rep created by: Planner → gpt-4o & Plan Executor → gpt-4o

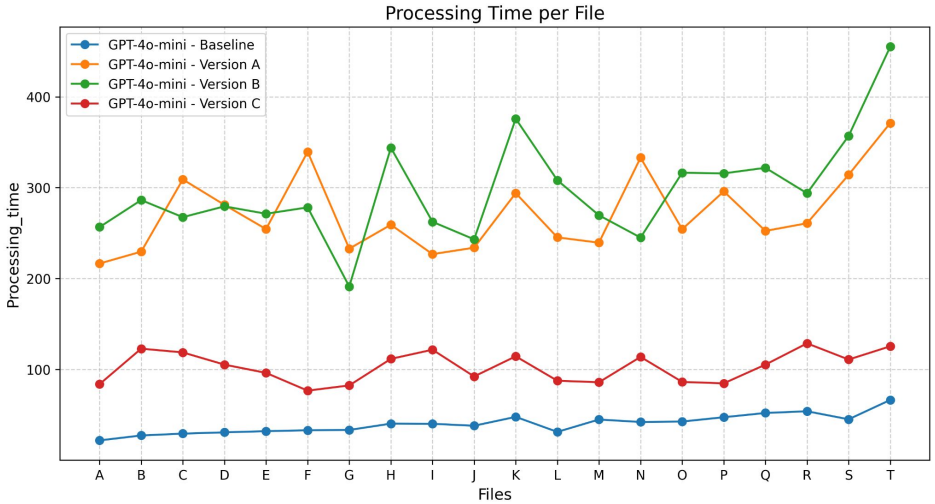
Evaluation Result

MetaMesh framework shows reduced overall token count usage. This will only get better as the number of downstream tasks against the document increases.

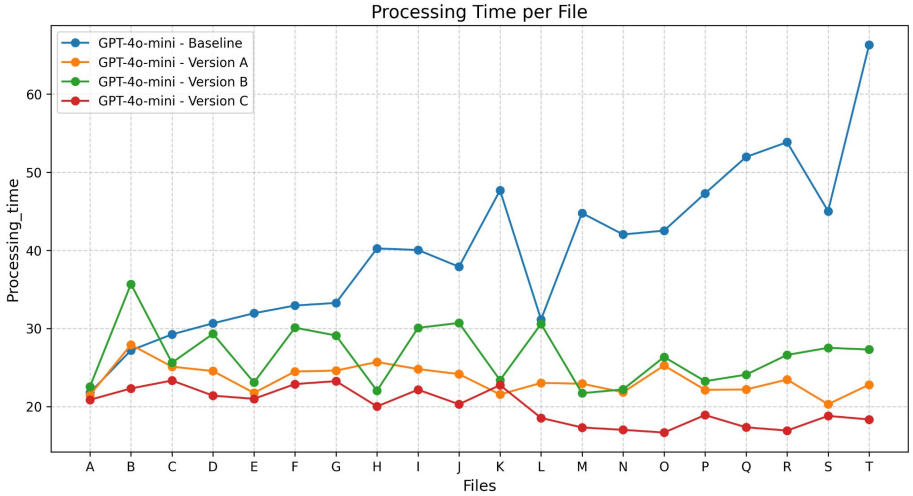


Evaluation Result

MetaMesh Framework shows a significant boost in latency (llm processing time) when documents are preprocessed and stored for downstream tasks.



No Document Pre-Processing



With Document Pre-Processing

Future Work

1. **Self-Evaluation and Collaborative Evaluation in Planner and Plan Executor**
    a. “LLMs are better evaluators than generators”
    b. Planner might design more representative templates and agent plans if provided with meaningful feedback in the process.
    c. Each agent in the Plan Executor can use self evaluation mechanisms to reflect and iteratively refine its output.
2. **Update the multi-agent environment to allow group of agents to work on a task**
    a. Currently MetaMesh run the agents linearly, each focusing on an isolated section.
    b. Collaboration among agents on and across tasks can lead to a boost in performance.
3. Update the Planner to select different representation structures **(knowledge graph, dataframe, JSON, etc)** based on what best captures the document information.
4. Setup evaluation for **local models** and see how they perform with and without intermediate representations.

Contributions & Conclusion

- MetaMesh provides a framework to dynamically create intermediate representations for contracts that can be used for downstream tasks.

- The framework can be easily scaled to create intermediate representations for any complex and long-form policy or procedure document.

- The framework shows significant reduction in token count and latency, but currently lacks on performance metrics like accuracy or F1-Score.

- If the representations fully captures all important information, then smaller models might be able to perform better or around similar as larger models that are using the complete doc.

References

Chen, G., et al. (2024). AutoAgents: A Framework for Automatic Agent Generation. arXiv preprint arXiv:2309.17288.

Guha, N., et al. (2023). LegalBench: A Collaboratively Built Benchmark for Measuring Legal Reasoning in Large Language Models. Digital Commons @ Osgoode Hall Law School.

Rodrigues, J., & Branco, A. (2024). Meta-prompting Optimized Retrieval-augmented Generation. Retrieved from [https://www.semanticscholar.org/reader/20587151ec740dcc0b0910783b868938a60cd7d0].

Suzgun, M., & Kalai, A. T. (2024). Meta-Prompting: Enhancing Language Models with Task-Agnostic Scaffolding. Retrieved from [https://www.semanticscholar.org/reader/59eeb8a259ef2a9c981868470480f53b67854060].

Wynter A., et al (2023). On Meta Prompting. arXiv preprint arXiv:2312.06562.

Wu, Q., et al. (2023). AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation Framework. arXiv preprint arXiv:2308.08155.

Zhang, Y., et al. (2024). Meta Prompting for AI Systems. arXiv preprint arXiv:2311.11482.