Playground

The DocETL Playground is an integrated development environment (IDE) for building and testing document processing pipelines. Built with Next.js and TypeScript, it provides a real-time interface to develop, test and refine your pipelines through a FastAPI backend.

Why a Playground? 🤔

This **interactive playground** streamlines development from prototype to production! **Our (in-progress) user studies show 100% of developers** found building pipelines significantly faster and easier with our playground vs traditional approaches.

Building complex LLM pipelines for your data often requires experimentation and iteration. The IDE lets you:

- 🚀 Test prompts and see results instantly
- ✨ Refine operations based on sample outputs
- 🔄 Build complex pipelines step-by-step

Public Playground

You can access our hosted playground at docetl.org/playground. You'll need to provide your own LLM API keys to use the service. The chatbot and prompt engineering assistants are powered by OpenAI models, so you'll need to provide an OpenAI API key.

Data Storage Notice

As this is a research project, we cache results and store data on our servers to improve the system. While we will never sell or release your data, if you have privacy concerns, we recommend running the playground locally using the installation instructions below.

Installation

There are two ways to run the playground:

1. Using Docker (Recommended for Quick Start)

The easiest way to get started is using Docker:

a) Create the required environment files:

Create `.env` in the root directory (for the FastAPI backend):

```
# Required: API key for your preferred LLM provider (OpenAI, Anthropic, etc)
# The key format will depend on your chosen provider (sk-..., anthro-...)
OPENAI_API_KEY=your_api_key_here
BACKEND_ALLOW_ORIGINS=http://localhost:3000,http://127.0.0.1:3000
BACKEND_HOST=localhost
BACKEND_PORT=8000
BACKEND_RELOAD=True
FRONTEND_HOST=localhost
FRONTEND_PORT=3000
```

Create `.env.local` in the `website` directory (for the frontend) **note that this must be in the `website` directory**:

```
# Optional: These are only needed if you want to use the AI assistant chatbot
# and prompt engineering tools. Must be OpenAI API keys specifically.
OPENAI_API_KEY=sk-xxx
OPENAI_API_BASE=https://api.openai.com/v1
MODEL_NAME=gpt-4o-mini

NEXT_PUBLIC_BACKEND_HOST=localhost
NEXT_PUBLIC_BACKEND_PORT=8000
```

b) Run Docker:

```
make docker
```

This will:

- Create a Docker volume for persistent data
- Build the DocETL image
- Run the container with the UI accessible at http://localhost:3000 and API at http://localhost:8000

To clean up Docker resources (note that this will delete the Docker volume):

```
make docker-clean
```

2. Running Locally (Development Setup)

For development or if you want to run the UI locally:

1. Clone the repository:

```
git clone https://github.com/ucbepic/docetl.git
cd docetl
```

2. Set up environment variables in `.env` in the root directory:

```
LLM_API_KEY=your_api_key_here
BACKEND_ALLOW_ORIGINS=http://localhost:3000,http://127.0.0.1:3000
BACKEND_HOST=localhost
BACKEND_PORT=8000
BACKEND_RELOAD=True
FRONTEND_HOST=0.0.0.0
FRONTEND_PORT=3000
```

Create `.env.local` in the `website` directory:

```
OPENAI_API_KEY=sk-xxx
OPENAI_API_BASE=https://api.openai.com/v1
MODEL_NAME=gpt-4o-mini

NEXT_PUBLIC_BACKEND_HOST=localhost
NEXT_PUBLIC_BACKEND_PORT=8000
```

Note

Note that the OpenAI API key, base, and model name in the `.env.local` file are only for the UI assistant functionality, not the DocETL pipeline execution engine.

1. Install dependencies:

```
make install      # Install Python package
make install-ui   # Install UI dependencies
```

2. Start the development server:

```
make run-ui-dev
```

3. Navigate to http://localhost:3000/playground to access the playground.

Setting up the AI Assistant

The UI offers an optional chat-based assistant that can help you iteratively develop your pipeline. It is currently very experimental. It can't write to your pipeline, but you can bounce ideas off of it and get it to help you iteratively develop your pipeline.

To use the assistant, you need to set your OpenAI API key in the `.env.local` file in the website directory. You can get an API key here. The API key should be in the following format: `sk-proj-...`. We only support the openai models for the assistant.

Self-hosting with UI API key management

If you want to host your own version of DocETL for your organization while allowing users to set their API keys through the UI, you'll need to set up encryption. Add the following to both `.env` and `website/.env.local`:

```
DOCETL_ENCRYPTION_KEY=your_secret_key_here
```

This shared encryption key allows API keys to be securely encrypted when sent to your server. Make sure to use the same value in both files.

Complex Tutorial

See this YouTube video for a more in depth tutorial on how to use the playground.

Simple Tutorial: Extracting Funny Quotes from Presidential Debates

In this tutorial, we'll walk through using the DocETL playground to extract funny or memorable quotes from presidential debate transcripts. We'll see how to:

1. Upload and explore data
2. Run a basic pipeline with sampling
3. Examine outputs
4. Iterate on prompts to improve results

Using LLMs to Help Write Pipelines

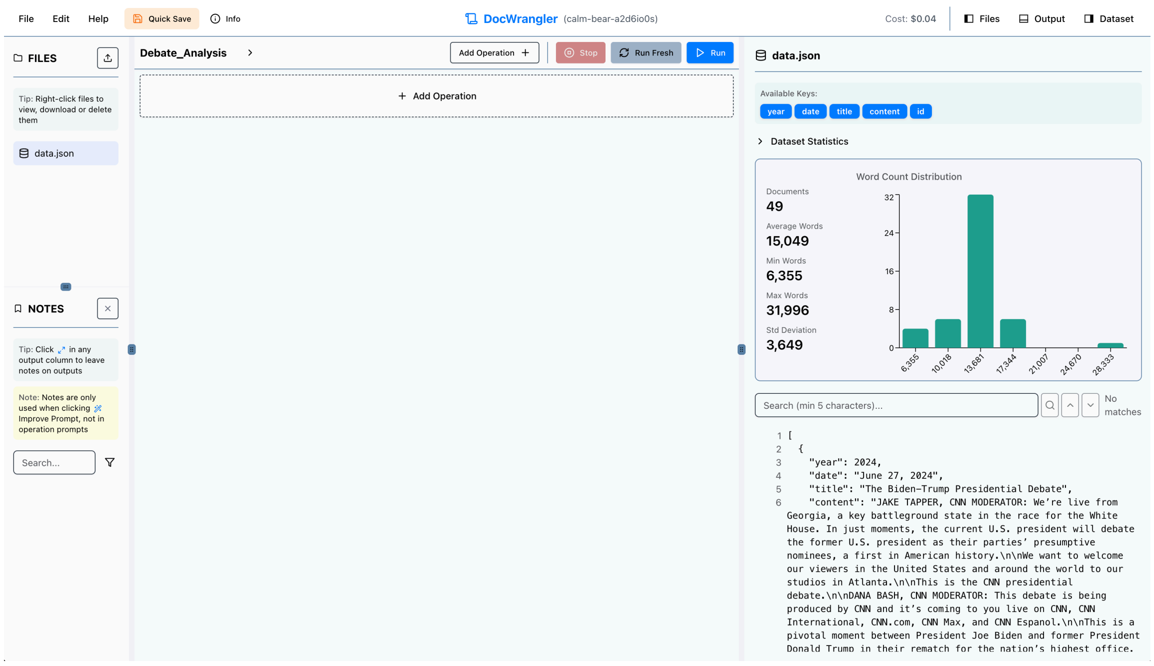
Want to use an LLM like ChatGPT or Claude to help you write your pipeline? See docetl.org/llms.txt for a big prompt you can copy paste into ChatGPT or Claude, before describing your task.

Step 1: Upload the Data

First, download the presidential debates dataset from here.

Once downloaded, use the left sidebar's upload button to load the data into the playground. The data contains transcripts from various presidential debates.

You can view the data by clicking the "toggle dataset view" button in the top right corner of the screen:



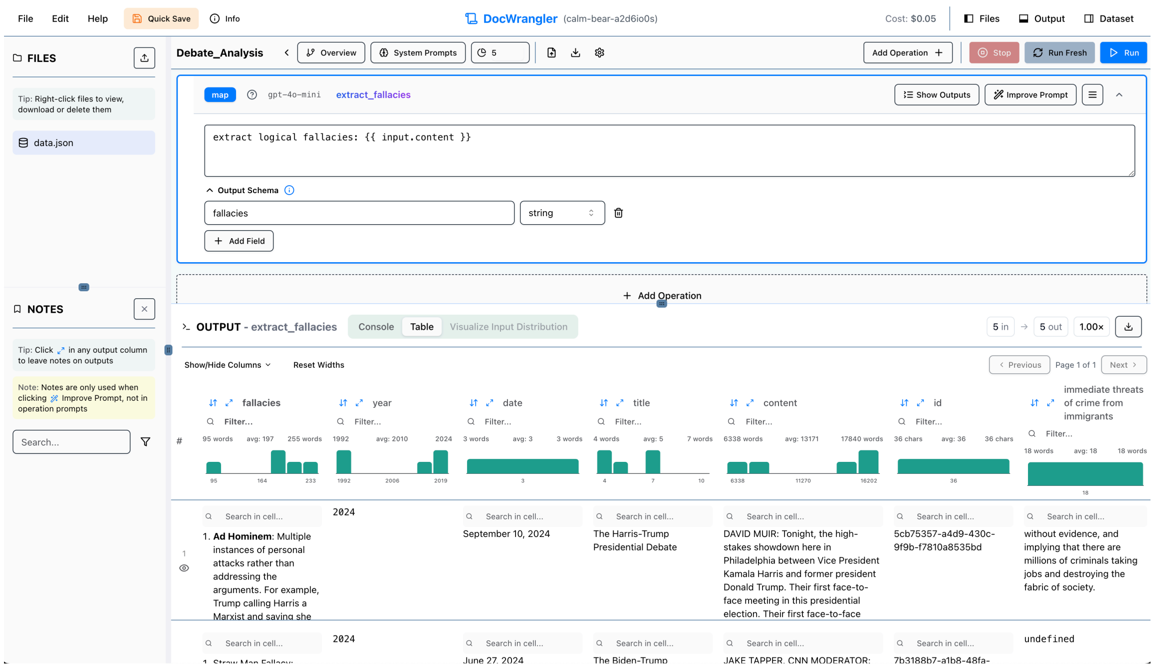
Documents	49
Average Words	15,049
Min Words	6,355
Max Words	31,996
Std Deviation	3,649

```
[
  {
    "year": 2024,
    "date": "June 27, 2024",
    "title": "The Biden–Trump Presidential Debate",
    "content": "JAKE TAPPER, CNN MODERATOR: We’re live from
Georgia, a key battleground state in the race for the White
House. In just moments, the current U.S. president will debate
the former U.S. president as their parties’ presumptive
nominees, a first in American history.\n\nWe want to welcome
our viewers in the United States and around the world to our
studios in Atlanta.\n\nThis is the CNN presidential
debate.\n\nDANA BASH, CNN MODERATOR: This debate is being
produced by CNN and it’s coming to you live on CNN, CNN
International, CNN.com, CNN Max, and CNN Espanol.\n\nThis is a
pivotal moment between President Joe Biden and former President
Donald Trump in their rematch for the nation’s highest office.
```

Step 2: Add a Map Operation

The pipeline is set to run on a sample of 5 documents, as indicated by the sample icon next to the pipeline name. This sampling helps us quickly test and iterate on our prompts without processing the entire dataset. This can be changed.

We'll add a map operation that processes each debate transcript to extract logical fallacies. Click the "Add Operation" button in the top right corner of the screen, and select "Map" under the LLM section. You can set the operation name to "extract_fallacies", and write a prompt and output schema.



```
extract logical fallacies: {{ input.content }}
```

Output Schema

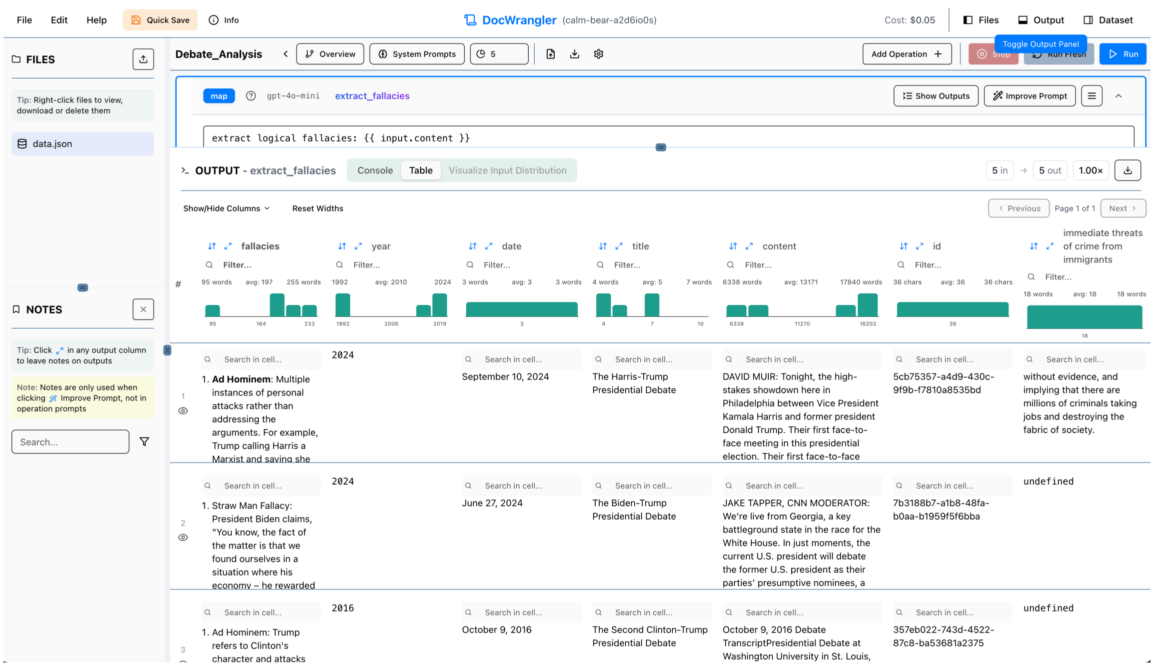
fallacies	string

#	fallacies	year	date	title	content	id	immediate threats of crime from immigrants
	95 words / avg: 197 / 255 words	1992 / avg: 2010 / 2024	3 words / avg: 3 / 3 words	4 words / avg: 5 / 7 words	6338 words / avg: 13171 / 17840 words	36 chars / avg: 36 / 36 chars	18 words / avg: 18 / 18 words
1	1. **Ad Hominem**: Multiple instances of personal attacks rather than addressing the arguments. For example, Trump calling Harris a Marxist and saying she	2024	September 10, 2024	The Harris-Trump Presidential Debate	DAVID MUIR: Tonight, the high-stakes showdown here in Philadelphia between Vice President Kamala Harris and former president Donald Trump. Their first face-to-face meeting in this presidential election. Their first face-to-face	5cb75357-a4d9-430c-9f9b-f7810a8535bd	without evidence, and implying that there are millions of criminals taking jobs and destroying the fabric of society.
	1. Straw Man Fallacy:	2024	June 27, 2024	The Biden-Trump	JAKE TAPPER, CNN MODERATOR:	7b3188b7-a1b8-48fa-	undefined

Step 3: Run the Pipeline and Check Outputs

Click the "Run" button to execute the pipeline. The outputs panel will show two important tabs:

- **Console**: Displays progress information and any potential errors
- **Table**: Shows the extracted funny quotes from each document in a table, as well as the other key/value pairs in the document. Here's what the table looks like after running the pipeline:



```
extract logical fallacies: {{ input.content }}
```

#	fallacies	year	date	title	content	id	immediate threats of crime from immigrants
1	1. **Ad Hominem**: Multiple instances of personal attacks rather than addressing the arguments. For example, Trump calling Harris a Marxist and saying she	2024	September 10, 2024	The Harris-Trump Presidential Debate	DAVID MUIR: Tonight, the high-stakes showdown here in Philadelphia between Vice President Kamala Harris and former president Donald Trump. Their first face-to-face meeting in this presidential election. Their first face-to-face	5cb75357-a4d9-430c-9f9b-f7810a8535bd	without evidence, and implying that there are millions of criminals taking jobs and destroying the fabric of society.
2	1. Straw Man Fallacy: President Biden claims, "You know, the fact of the matter is that we found ourselves in a situation where his economy – he rewarded	2024	June 27, 2024	The Biden-Trump Presidential Debate	JAKE TAPPER, CNN MODERATOR: We're live from Georgia, a key battleground state in the race for the White House. In just moments, the current U.S. president will debate the former U.S. president as their parties' presumptive nominees, a	7b3188b7-a1b8-48fa-b0aa-b1959f5f6bba	undefined
3	1. Ad Hominem: Trump refers to Clinton's character and attacks	2016	October 9, 2016	The Second Clinton-Trump Presidential Debate	October 9, 2016 Debate TranscriptPresidential Debate at Washington University in St. Louis,	357eb022-743d-4522-87c8-ba53681a2375	undefined

You can resize the rows and columns in the table by clicking and dragging the edges of the table cells, as in the image above. You can also rezise the outputs panel by clicking and dragging the top edge of the panel.

Step 4: Iterate on the Prompt

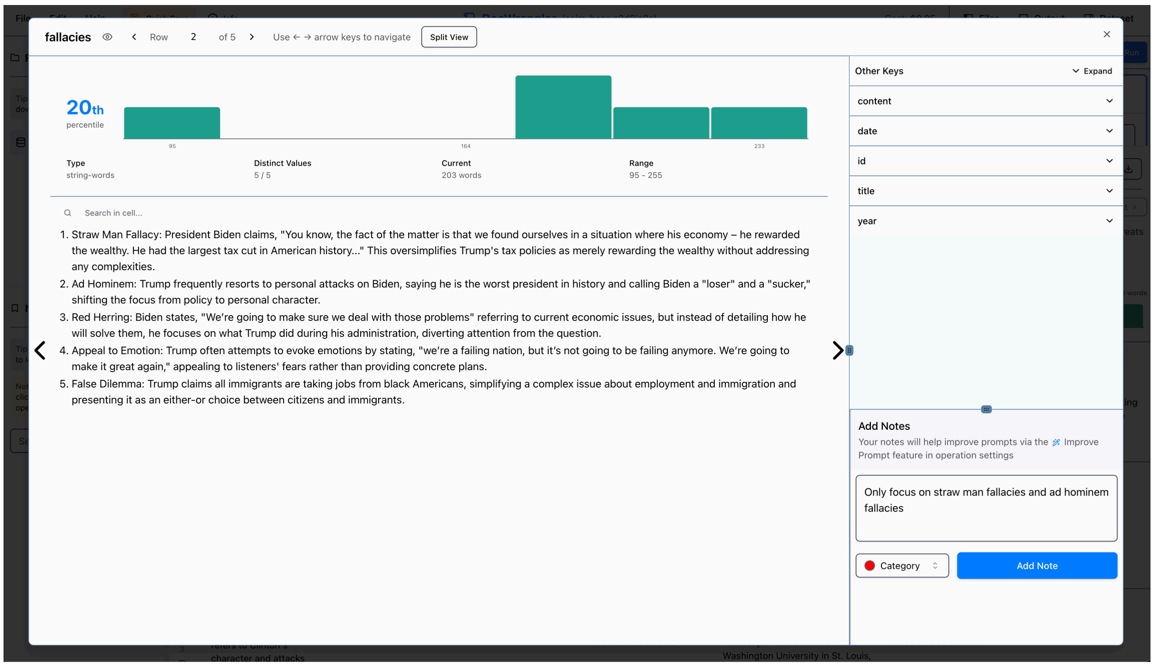
After reviewing the initial outputs, let's inspect them row by row to identify areas for improvement. Click the expand icon next to any column to see the full content.

We can modify the prompt to request additional context based on what we observe in the data. Here are some general patterns that could use improvement:

- Some quotes lack sufficient background context
- Speaker information could be more detailed
- The reasoning behind why quotes are memorable isn't always clear
- Time and setting details are often missing

- Impact and reactions to quotes aren't consistently captured

Here's an example of what giving notes might look like:



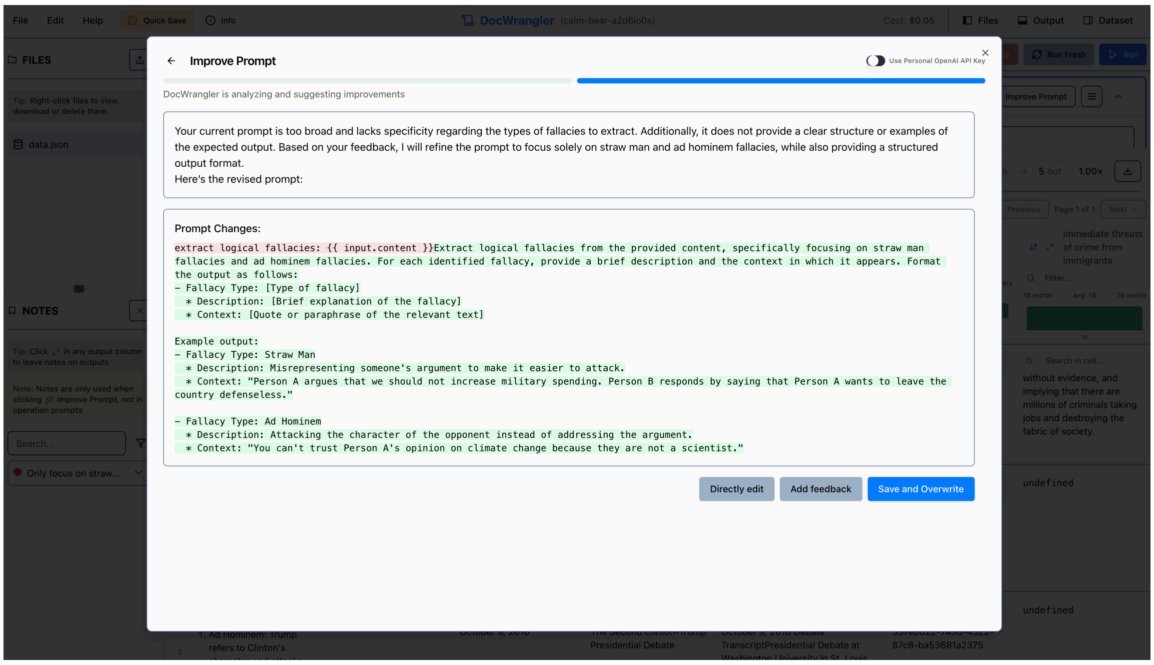
1. Straw Man Fallacy: President Biden claims, "You know, the fact of the matter is that we found ourselves in a situation where his economy – he rewarded the wealthy. He had the largest tax cut in American history..." This oversimplifies Trump's tax policies as merely rewarding the wealthy without addressing any complexities.
2. Ad Hominem: Trump frequently resorts to personal attacks on Biden, saying he is the worst president in history and calling Biden a "loser" and a "sucker," shifting the focus from policy to personal character.
3. Red Herring: Biden states, "We're going to make sure we deal with those problems" referring to current economic issues, but instead of detailing how he will solve them, he focuses on what Trump did during his administration, diverting attention from the question.
4. Appeal to Emotion: Trump often attempts to evoke emotions by stating, "we're a failing nation, but it's not going to be failing anymore. We're going to make it great again," appealing to listeners' fears rather than providing concrete plans.
5. False Dilemma: Trump claims all immigrants are taking jobs from black Americans, simplifying a complex issue about employment and immigration and presenting it as an either-or choice between citizens and immigrants.

Add Notes

Your notes will help improve prompts via the Improve Prompt feature in operation settings

Only focus on straw man fallacies and ad hominem fallacies

Once you've added enough notes (3-5 or more), you can click on the "Improve Prompt" button in the top right corner of the operation card. This will invoke the DocWrangler Prompt Improvement Assistant, which will suggest edits to your prompt:

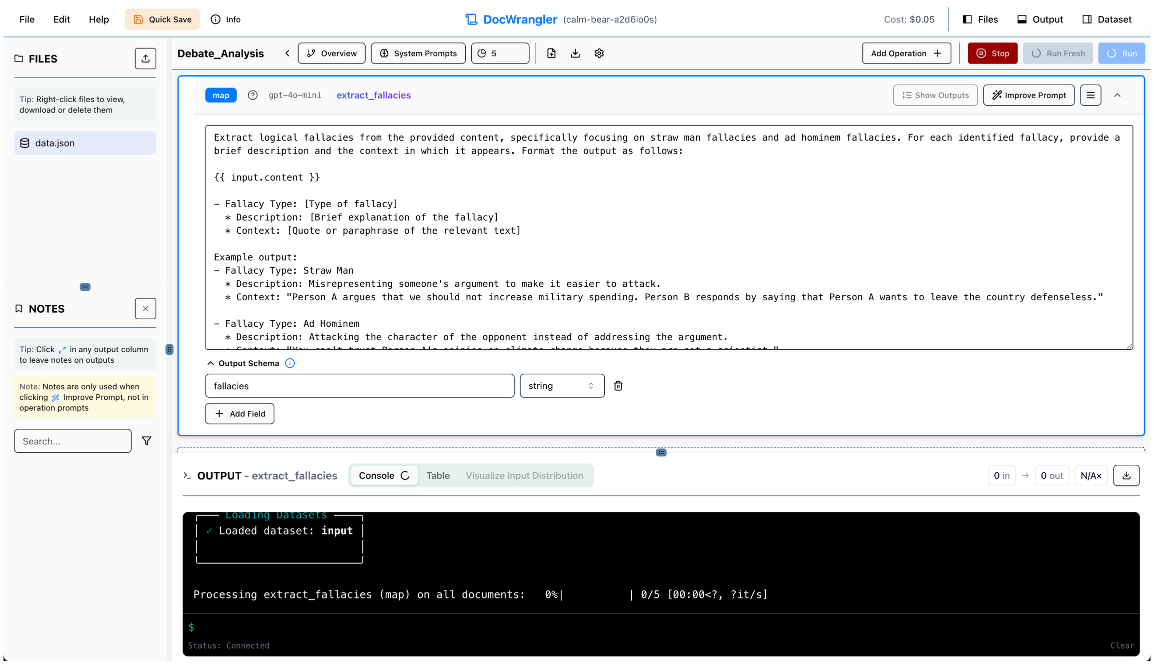

DocWrangler (calm-bear-a2d6io0s)
Improve Prompt
Use Personal OpenAI API Key
DocWrangler is analyzing and suggesting improvements
Directly edit
Add feedback
Save and Overwrite

Your current prompt is too broad and lacks specificity regarding the types of fallacies to extract. Additionally, it does not provide a clear structure or examples of the expected output. Based on your feedback, I will refine the prompt to focus solely on straw man and ad hominem fallacies, while also providing a structured output format.
Here's the revised prompt:

Prompt Changes:

```
extract logical fallacies: {{ input.content }}Extract logical fallacies from the provided content, specifically focusing on straw man
fallacies and ad hominem fallacies. For each identified fallacy, provide a brief description and the context in which it appears. Format
the output as follows:
- Fallacy Type: [Type of fallacy]
  * Description: [Brief explanation of the fallacy]
  * Context: [Quote or paraphrase of the relevant text]

Example output:
- Fallacy Type: Straw Man
  * Description: Misrepresenting someone's argument to make it easier to attack.
  * Context: "Person A argues that we should not increase military spending. Person B responds by saying that Person A wants to leave the
country defenseless."

- Fallacy Type: Ad Hominem
  * Description: Attacking the character of the opponent instead of addressing the argument.
  * Context: "You can't trust Person A's opinion on climate change because they are not a scientist."
```

Once you're satisfied with the new prompt, click "Save" to update the operation, and then you can rerun the pipeline to see the new outputs.


File
Edit
Help
Quick Save
Info
DocWrangler (calm-bear-a2d6io0s)
Cost: $0.05
Files
Output
Dataset
FILES
Tip: Right-click files to view, download or delete them
data.json
NOTES
Tip: Click in any output column to leave notes on outputs
Note: Notes are only used when clicking Improve Prompt, not in operation prompts
Search...
Debate_Analysis
Overview
System Prompts
5
Add Operation
Stop
Run Fresh
Run
map
gpt-4o-mini
extract_fallacies
Show Outputs
Improve Prompt
Output Schema
fallacies
string
Add Field
OUTPUT - extract_fallacies
Console
Table
Visualize Input Distribution
0 in
0 out
N/A×
Clear

```
Extract logical fallacies from the provided content, specifically focusing on straw man fallacies and ad hominem fallacies. For each identified fallacy, provide a
brief description and the context in which it appears. Format the output as follows:

{{ input.content }}

- Fallacy Type: [Type of fallacy]
  * Description: [Brief explanation of the fallacy]
  * Context: [Quote or paraphrase of the relevant text]

Example output:
- Fallacy Type: Straw Man
  * Description: Misrepresenting someone's argument to make it easier to attack.
  * Context: "Person A argues that we should not increase military spending. Person B responds by saying that Person A wants to leave the country defenseless."

- Fallacy Type: Ad Hominem
  * Description: Attacking the character of the opponent instead of addressing the argument.
```

```
┌── Loading Datasets ──┐
│ ✓ Loaded dataset: input │
│                      │
└──────────────────────┘

Processing extract_fallacies (map) on all documents:   0%|          | 0/5 [00:00<?, ?it/s]

$
Status: Connected
```

Caching Behavior

DocETL automatically caches the outputs of each operation. This means that if you run the pipeline multiple times without changing the operations, it will use the cached results instead of reprocessing the documents. This is especially helpful when:

- Iterating on downstream operations in a multi-step pipeline
- Running the pipeline on the full dataset after testing on samples
- Sharing results with teammates (cached outputs persist across sessions)

The cache is invalidated only when you modify an operation's configuration (e.g., changing the prompt or schema). This ensures you always see fresh results when making changes while avoiding unnecessary recomputation.

If you want to bypass the cache and force recomputation, you can click the "Clear and Run" button instead of the regular "Run" button.

Step 5: Run the Pipeline on the Entire Dataset

Once you're satisfied with your prompt, you can run the pipeline on the entire dataset. First, clear the sample size by clicking on the settings or gear icon next to the pipeline name.

Then, click the "Run" button again. This will process all documents and update the outputs panel with the results.

You can export the results to a CSV file by clicking the "download" button in the top right corner of the outputs panel, near where it says "Selectivity: 1.00x". The selectivity is the ratio of the number of documents in the output to the number of documents in the input for that operation. In this case, since we ran the pipeline on the full dataset, the selectivity is 1.0x.

Model Note

In this tutorial, we used `azure/gpt-4o` instead of `gpt-4o-mini` since content filters were triggered by `gpt-4o-mini` when processing political debate content. If you encounter similar content filter issues with `gpt-4o-mini`, consider using `azure/gpt-4o` or another model with less restrictive filters.

Community

Welcome to the DocETL community! We're excited to have you join us in exploring and improving document extraction and transformation workflows. We are committed to fostering an inclusive community for all people, regardless of technical background.

Code of Conduct

While we don't have a formal code of conduct page, we expect all community members to treat each other with respect and kindness. We do not tolerate harassment or discrimination of any kind. If you experience any issues, please reach out to the project maintainers immediately.

Contributions

We welcome contributions from everyone who is interested in improving DocETL. Here's how you can get involved:

1. **Report Issues**: If you encounter a bug or have a feature request, open an issue on our GitHub repository.
2. **Join Discussions**: Have a question or want to discuss ideas? Post on our Discord server.
3. **Contribute Code**: Look for issues tagged with "help wanted" or "good first issue" on GitHub. These are great starting points for new contributors.
4. **Join Working Groups**: We will create working groups in Discord focused on different project areas as discussed in our roadmap. Join the group(s) that interests you most!

To contribute code:

1. Fork the repository on GitHub.
2. Create a new branch for your changes.
3. Make your changes in your branch.
4. Submit a pull request with your changes.

Connect with Us

- **GitHub Repository**: Contribute to the project or report issues on our GitHub repo.
- **Discord Community**: Join our Discord server; we're looking to build a vibrant community of people interested in intelligent document processing.
- **Lab Webpages**: We are affiliated with the EPIC Lab at UC Berkeley. Visit our Lab Page for a description of our research. We are also affiliated with the Data Systems and Foundations group at UC Berkeley--visit our DSF Page for more information.

Request a Tutorial or Research Talk

Interested in having us give a tutorial or research talk on DocETL? We'd love to connect! Please email shreyashankar@berkeley.edu to set up a time. Let us know what your team is interested in learning about (e.g., tutorial or research) so we can tailor the presentation to your interests.

Frequently Encountered Issues

KeyError in Operations

If you're encountering a KeyError, it's often due to missing an unnest operation in your workflow. The unnest operation is crucial for flattening nested data structures.

Solution: Add an unnest operation to your pipeline before accessing nested keys. If you're still having trouble, don't hesitate to open an issue on GitHub or ask for help on our Discord server.

Browser freezing because of stale client storage

Run the following script:

Browser Storage Cleanup Script

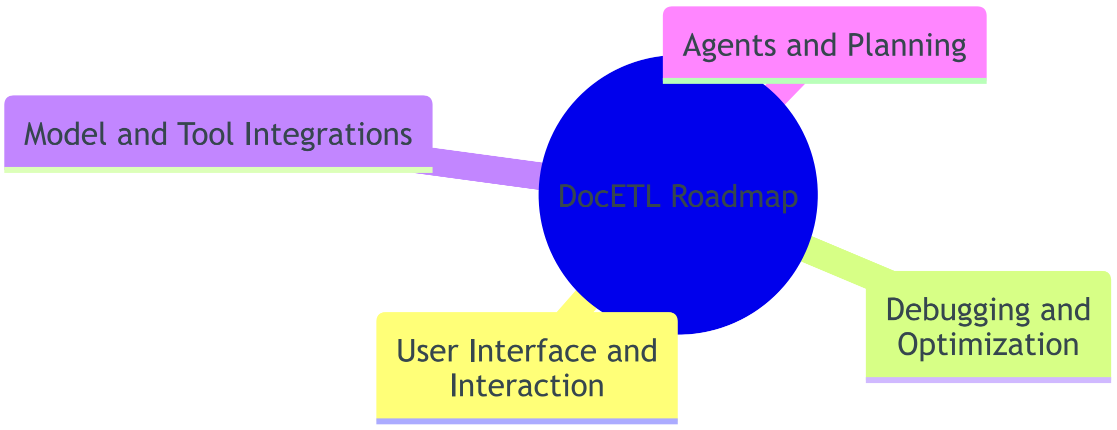
```
// Function to delete all localStorage items with prefix 'docetl_'
function cleanupDocETLStorage() {
    const prefix = 'docetl_';
    const itemsToDelete = [];

    // First, collect all matching keys
    for (let i = 0; i < localStorage.length; i++) {
        const key = localStorage.key(i);
        if (key && key.startsWith(prefix)) {
            itemsToDelete.push(key);
        }
    }

    // Then delete them and keep count
    const deletedCount = itemsToDelete.length;
    itemsToDelete.forEach(key => {
        localStorage.removeItem(key);
        console.log(`Deleted key: ${key}`);
    });

    console.log(`Cleanup complete. Deleted ${deletedCount} items with prefix
"${prefix}"`);
}

// Execute the cleanup
cleanupDocETLStorage();
```

Roadmap

Join Our Working Groups

Are you interested in contributing to any of these projects or have ideas for new areas of exploration? Join our Discord server to participate in our working groups and collaborate with the community!

We're constantly working to improve DocETL and explore new possibilities in document processing. Our current ideas span both research and engineering problems, and are organized into the following categories:


DocETL Roadmap
Agents and Planning
Model and Tool Integrations
Debugging and Optimization
User Interface and Interaction

User Interface and Interaction

- **Natural Language to DocETL Pipeline**: Building tools to generate DocETL pipelines from natural language descriptions.
- **Interactive Pipeline Creation**: Developing intuitive interfaces for creating and optimizing DocETL pipelines interactively.

Debugging and Optimization

- **DocETL Debugger**: Creating a debugger with provenance tracking, allowing users to visualize all intermediates that contributed to a specific output.
- **Plan Efficiency Optimization**: Implementing strategies (and devising new strategies) to reduce latency and cost for the most accurate plans. This includes batching LLM

calls, using model cascades, and fusing operators.

Model and Tool Integrations

- **Model Diversity**: Extending support beyond OpenAI to include a wider range of models, with a focus on local models.
- **OCR and PDF Extraction**: Improving integration with OCR technologies and PDF extraction tools for more robust document processing.
- **Multimodal Data Processing**: Enhancing DocETL to handle multimodal data, including text, images, audio, and video (as many of the LLMs support multimodal inputs, anyways).

Agents and Planning

- **Smarter Agent and Planning Architectures**: Optimizing plan exploration based on data characteristics. For instance, refining the optimizer to avoid unnecessary exploration of plans with the gather operator for tasks that don't require peripheral context when decomposing map operations for large documents.
- **Context-Aware Sampling for Validation**: Creating algorithms that can identify and extract the most representative samples from different parts of a document, including the beginning, middle, and end, to use in validaton prompts. This approach will help validation agents to verify that all sections of documents are adequately represented in the outputs, avoiding blind spots in the analysis due to truncation--as we currently naive truncate the middle of documents in validation prompts.
- **Benchmarks**: Developing a suite of benchmarks to evaluate the performance of different optimization strategies and agent architectures. These benchmarks will help us understand the trade-offs between accuracy, efficiency, and cost in different scenarios, guiding the development of more effective optimization techniques.