Tutorial: Analyzing Medical Transcripts with DocETL

This tutorial will guide you through the process of using DocETL to analyze medical transcripts and extract medication information. We'll create a pipeline that identifies medications, resolves similar names, and generates summaries of side effects and therapeutic uses.

Installation

First, let's install DocETL. Follow the instructions in the installation guide to set up DocETL on your system.

Setting up API Keys

DocETL uses LiteLLM under the hood, which supports various LLM providers. For this tutorial, we'll use OpenAI, as DocETL tests and existing pipelines are run with OpenAI.

Setting up your API Key

Set your OpenAI API key as an environment variable:

```
export OPENAI_API_KEY=your_api_key_here
```

Alternatively, you can create a `.env` file in your project directory and add the following line:

```
OPENAI_API_KEY=your_api_key_here
```

OpenAI Dependency

DocETL has been primarily tested with OpenAI's language models and relies heavily on their structured output capabilities. While we aim to support other providers in the future, using OpenAI is currently recommended for the best experience and most reliable results.

If you choose to use a different provider, be aware that you may encounter unexpected behavior or reduced functionality, especially with operations that depend on structured outputs. We use tool calling to extract structured outputs from the LLM's response, so make sure your provider supports tool calling.

If using a Gemini model, you can use the `gemini` prefix for the model name. For example, `gemini/gemini-2.0-flash`. (This has worked pretty well for us so far, and is so cheap!)

If using Ollama (e.g., llama 3.2), make sure your output schemas are not too complex, since these models are not as good as OpenAI for structured outputs! For example, use parallel map operations to reduce the number of output attributes per prompt.

Preparing the Data

Organize your medical transcript data in a JSON file as a list of objects. Each object should have a "src" key containing the transcript text. You can download the example dataset here.

Sample Data Structure

```
[
    {
        "src": "Doctor: Hello, Mrs. Johnson. How have you been feeling since
starting the new medication, Lisinopril?\nPatient: Well, doctor, I've noticed
my blood pressure has improved, but I've been experiencing some dry cough...",
    },
    {
        "src": "Doctor: Good morning, Mr. Smith. I see you're here for a
follow-up on your Metformin prescription.\nPatient: Yes, doctor. I've been
taking it regularly, but I'm concerned about some side effects I've been
experiencing...",
    }
]
```

Save this file as `medical_transcripts.json` in your project directory.

Creating the Pipeline

Now, let's create a DocETL pipeline to analyze this data. We'll use a series of operations to extract and process the medication information:

1. **Medication Extraction**: Analyze each transcript to identify and list all mentioned medications.
2. **Unnesting**: The extracted medication list is flattened, such that each medication (and associated data) is a separate document. This operator is akin to the pandas `explode` operation.
3. **Medication Resolution**: Similar medication names are resolved to standardize the entries. This step helps in consolidating different variations or brand names of the same medication. For example, step 1 might extract "Ibuprofen" and "Motrin 800mg" as separate medications, and step 3 might resolve them to a single "Ibuprofen" entry.
4. **Summary Generation**: For each unique medication, generate a summary of side effects and therapeutic uses based on information from all relevant transcripts.

Create a file named `pipeline.yaml` with the following structure:

Pipeline Structure

```yaml
datasets:
  transcripts:
    path: medical_transcripts.json
    type: file

default_model: gpt-4o-mini

system_prompt: # This is optional, but recommended for better performance. It
is applied to all operations in the pipeline.
  dataset_description: a collection of transcripts of doctor visits
  persona: a medical practitioner analyzing patient symptoms and reactions to
medications

operations:
  - name: extract_medications
    type: map
    output:
      schema:
        medication: list[str]
    prompt: |
      Analyze the following transcript of a conversation between a doctor and a
patient:
      {{ input.src }}
      Extract and list all medications mentioned in the transcript.
      If no medications are mentioned, return an empty list.

  - name: unnest_medications
    type: unnest
    unnest_key: medication

  - name: resolve_medications
    type: resolve
    blocking_keys:
      - medication
    blocking_threshold: 0.6162
    comparison_prompt: |
      Compare the following two medication entries:
      Entry 1: {{ input1.medication }}
      Entry 2: {{ input2.medication }}
      Are these medications likely to be the same or closely related?
    embedding_model: text-embedding-3-small
    output:
      schema:
        medication: str
    resolution_prompt: |
      Given the following matched medication entries:
      {% for entry in inputs %}
      Entry {{ loop.index }}: {{ entry.medication }}
      {% endfor %}
      Determine the best resolved medication name for this group of entries.
The resolved
      name should be a standardized, widely recognized medication name that
best represents
      all matched entries.

  - name: summarize_prescriptions
    type: reduce
```

```
    reduce_key:
      - medication
    output:
      schema:
        side_effects: str
        uses: str
    prompt: |
      Here are some transcripts of conversations between a doctor and a
patient:

      {% for value in inputs %}
      Transcript {{ loop.index }}:
      {{ value.src }}
      {% endfor %}

      For the medication {{ reduce_key }}, please provide the following
information based on all the transcripts above:

      1. Side Effects: Summarize all mentioned side effects of {{ reduce_key
}}.
      2. Therapeutic Uses: Explain the medical conditions or symptoms for which
{{ reduce_key }} was prescribed or recommended.

      Ensure your summary:
      - Is based solely on information from the provided transcripts
      - Focuses only on {{ reduce_key }}, not other medications
      - Includes relevant details from all transcripts
      - Is clear and concise
      - Includes quotes from the transcripts

pipeline:
  steps:
    - name: medical_info_extraction
      input: transcripts
      operations:
        - extract_medications
        - unnest_medications
        - resolve_medications
        - summarize_prescriptions
  output:
    type: file
    path: medication_summaries.json
    intermediate_dir: intermediate_results
```

Running the Pipeline

Pipeline Performance

When running this pipeline on a sample dataset, we observed the following performance metrics using `gpt-4o-mini` as defined in the pipeline:

- Total cost: $0.10
- Total execution time: 49.13 seconds

If you want to run it on a smaller sample, set the `sample` parameter for the map operation. For example, `sample: 10` will run the pipeline on a random sample of 10 transcripts:

```
operations:
  - name: extract_medications
    type: map
    sample: 10
    ...
```

To execute the pipeline, run the following command in your terminal:

```
docetl run pipeline.yaml
```

This will process the medical transcripts, extract medication information, resolve similar medication names, and generate summaries of side effects and therapeutic uses for each medication. The results will be saved in `medication_summaries.json`.

Further Questions

What if I want to focus on a specific type of medication or medical condition?

You can modify the prompts in the `extract_medications` and `summarize_prescriptions` operations to focus on specific types of medications or medical conditions. For example, you could update the `extract_medications` prompt to only list medications related to cardiovascular diseases.

How can I improve the accuracy of medication name resolution?

The `resolve_medications` operation uses a blocking threshold and comparison prompt to identify similar medication names. Learn more about how to configure this operation in the resolve operation documentation. To automatically find the optimal blocking threshold for your data, you can invoke the optimizer, as described in the optimization documentation.

Can I process other types of medical documents with this pipeline?

Yes, you can adapt this pipeline to process other types of medical documents by modifying the input data format and adjusting the prompts in each operation. For example, you could use it to analyze discharge summaries, clinical notes, or research papers by updating the extraction and summarization prompts accordingly.

How can I optimize the performance of this pipeline?

If you're unsure about the optimal configuration for your specific use case, you can use DocETL's optimizer, which can be invoked using `docetl build` instead of `docetl run`. Learn more about the optimizer in the optimization documentation.

How can I use the pandas integration?

DocETL provides a pandas integration for a few operators (map, filter, merge, agg). Learn more about the pandas integration in the pandas documentation.