Medical Document Classification with Ollama

This tutorial demonstrates how to use DocETL with Ollama models to classify medical documents into predefined categories. We'll use a simple map operation to process a set of medical records, ensuring that sensitive information remains private by using a locally-run model.

Setup

Prerequisites

Before we begin, make sure you have Ollama installed and running on your local machine.

You'll need to set the OLLAMA_API_BASE environment variable:

```
export OLLAMA_API_BASE=http://localhost:11434/
```

API Details

For more information on the Ollama REST API, refer to the Ollama documentation.

Pipeline Configuration

Let's create a pipeline that classifies medical documents into categories such as "Cardiology", "Neurology", "Oncology", etc.

Initial Pipeline Configuration

```yaml
datasets:
  medical_records:
    type: file
    path: "medical_records.json"

default_model: ollama/llama3

system_prompt:
  dataset_description: a collection of medical records
  persona: a medical practitioner analyzing patient symptoms and reactions to
medications

operations:
  - name: classify_medical_record
    type: map
    output:
      schema:
        categories: "list[str]"
    prompt: |
      Classify the following medical record into one or more of these
categories: Cardiology, Neurology, Oncology, Pediatrics, Orthopedics.

      Medical Record:
      {{ input.text }}

      Return your answer as a JSON list of strings, e.g., ["Cardiology",
"Neurology"].

pipeline:
  steps:
    - name: medical_classification
      input: medical_records
      operations:
        - classify_medical_record

output:
  type: file
  path: "classified_records.json"
```

Running the Pipeline with a Sample

To test our pipeline and estimate the required timeout, we'll first run it on a sample of documents.

Modify the `classify_medical_record` operation in your configuration to include a `sample` parameter:

```yaml
operations:
  - name: classify_medical_record
    type: map
    sample: 5
```

```
    output:
      schema:
        categories: "list[str]"
    prompt: |
      Classify the following medical record into one or more of these
categories: Cardiology, Neurology, Oncology, Pediatrics, Orthopedics.

      Medical Record:
      {{ input.text }}

      Return your answer as a JSON list of strings, e.g., ["Cardiology",
"Neurology"].
```

Now, run the pipeline with this sample configuration:

```
docetl run pipeline.yaml
```

Adjusting the Timeout

After running the sample, note the time it took to process 5 documents.

> **Timeout Calculation**
>
> Let's say it took 100 seconds to process 5 documents. You can use this to estimate the time needed for your full dataset. For example, if you have 1000 documents in total, you might want to set the timeout to:
>
> (100 seconds / 5 documents) * 1000 documents = 20,000 seconds

Now, adjust your pipeline configuration to include this timeout and remove the sample parameter:

```
operations:
  - name: classify_medical_record
    type: map
    timeout: 20000
    output:
      schema:
        categories: "list[str]"
    prompt: |
      Classify the following medical record into one or more of these
categories: Cardiology, Neurology, Oncology, Pediatrics, Orthopedics.
      Medical Record:
      {{ input.text }}
      Return your answer as a JSON list of strings, e.g., ["Cardiology",
"Neurology"].
```

Caching

DocETL caches results (even between runs), so if the same document is processed again, the answer will be returned from the cache rather than processed again (significantly speeding up processing).

Running the Full Pipeline

Now you can run the full pipeline with the adjusted timeout:

```
docetl run pipeline.yaml
```

This will process all your medical records, classifying them into the predefined categories.

Conclusion

Key Takeaways

- This pipeline demonstrates how to use Ollama with DocETL for local processing of sensitive data.
- Ollama integrates into multi-operation pipelines, maintaining data privacy.
- Ollama is a local model, so it is much slower than leveraging an LLM API like OpenAI. Adjust the timeout accordingly.
- DocETL's sample and timeout parameters help optimize the pipeline for efficient use of Ollama's capabilities.

For more information, e.g., for specific models, visit https://ollama.com/.