DocETL Optimizer

The DocETL optimizer finds a plan that improves the accuracy of your document processing pipelines. It works by analyzing and potentially rewriting operations marked for optimization, finding optimal plans for execution.

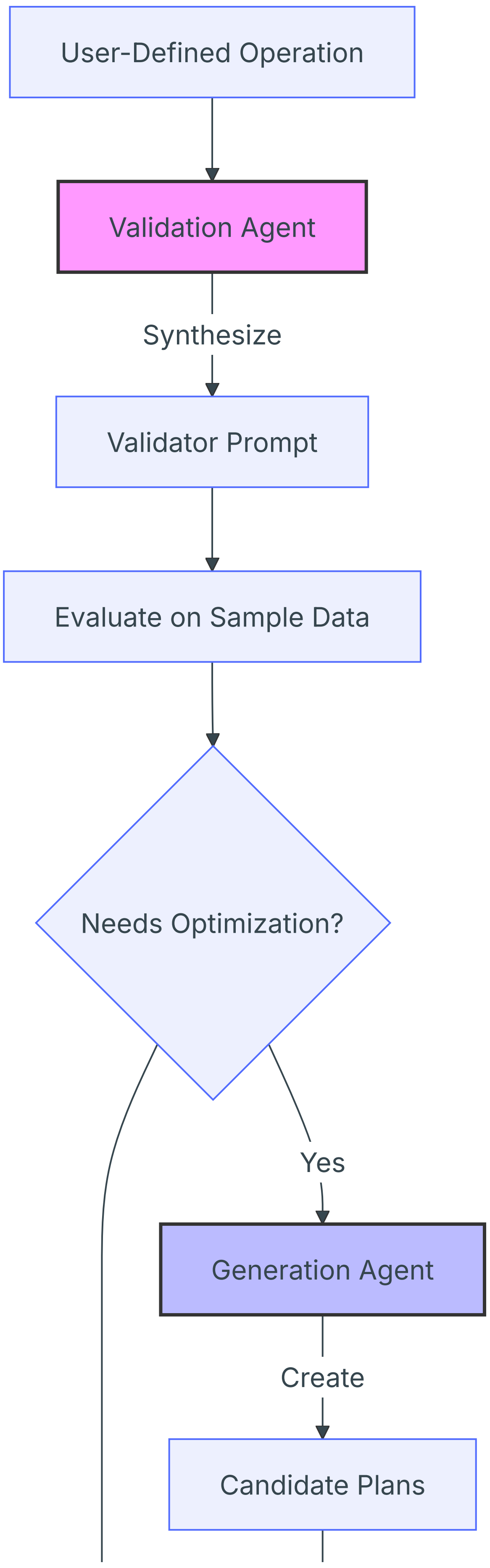
Key Features

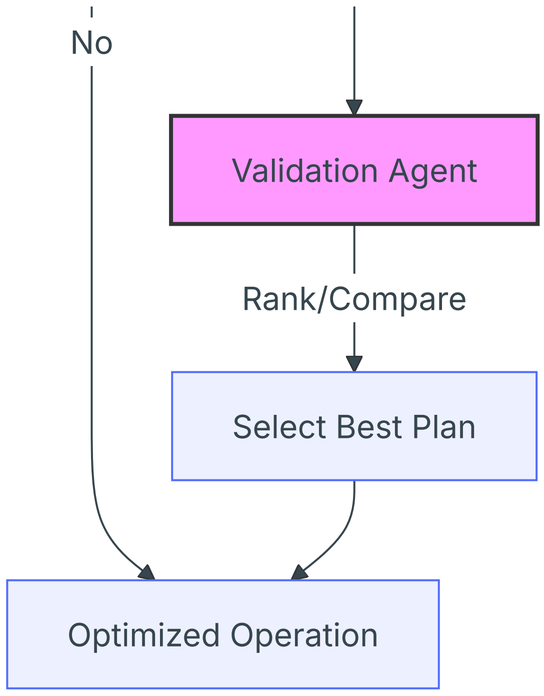
- Automatically decomposes complex operations into more efficient sub-pipelines
- Inserts resolve operations before reduce operations when beneficial
- Optimizes for large documents that exceed context limits
- Improves accuracy in high-volume reduce operations with incremental reduce

How It Works

The optimizer employs AI agents to generate and validate potential optimizations:

1. **Generation Agents**: Create alternative plans for operations, potentially breaking them down into multiple steps.
2. **Validation Agents**: Evaluate and compare the outputs of different plans to determine the most effective approach.


User-Defined Operation
Validation Agent
Synthesize
Validator Prompt
Evaluate on Sample Data
Needs Optimization?
Yes
Generation Agent
Create
Candidate Plans



Should I Use the Optimizer?

While any pipeline can potentially benefit from optimization, there are specific scenarios where using the optimizer can significantly improve your pipeline's performance and accuracy. When should you use the optimizer?

Large Documents

If you have documents that approach or exceed context limits and a map operation that transforms these documents using an LLM, the optimizer can help:

- Improve accuracy
- Enable processing of entire documents
- Optimize for large-scale data handling

Entity Resolution

The optimizer is particularly useful when:

```
- You need a resolve operation before your reduce operation
- You've defined a resolve operation but want to optimize it for speed using
blocking
```

High-Volume Reduce Operations

Consider using the optimizer when:

```
- You have many documents feeding into a reduce operation for a given key
- You're concerned about the accuracy of the reduce operation due to high
volume
- You want to optimize for better accuracy in complex reductions
```

Even if your pipeline doesn't fall into these specific categories, optimization can still be beneficial. For example, the optimizer can enhance your operations by adding gleaning to an operation, which uses an LLM-powered validator to ensure operation correctness. Learn more about gleaning.

Example: Optimizing Legal Contract Analysis

Let's consider a pipeline for analyzing legal contracts, extracting clauses, and summarizing them by type. Initially, you might have a single map operation to extract and tag clauses, followed by a reduce operation to summarize them. However, this approach might not be accurate enough for long contracts.

Initial Pipeline

In the initial pipeline, you might have a single map operation that attempts to extract all clauses and tag them with their types in one go. This is followed by a reduce operation that summarizes the clauses by type. Maybe the reduce operation accurately summarizes the clauses in a single LLM call per clause type, but the map operation might not be able to accurately extract and tag the clauses in a single LLM call.

Optimized Pipeline

After applying the optimizer, your pipeline could be transformed into a more efficient and accurate sub-pipeline:

1. **Split Operation**: Breaks down *each* long contract into manageable chunks.
2. **Map Operation**: Processes each chunk to extract and tag clauses.
3. **Reduce Operation**: For each contract, combine the extracted and tagged clauses from each chunk.

The goal of the DocETL optimizer is to try many ways of rewriting your pipeline and then select the best one. This may take some time (20-30 minutes for very complex tasks and large documents). But the optimizer's ability to break down complex tasks into more manageable sub-steps can lead to more accurate and reliable results.