Unnest Operation

The Unnest operation in DocETL is designed to expand an array field or a dictionary in the input data into multiple items. This operation is particularly useful when you need to process or analyze individual elements of an array or specific fields of a nested dictionary separately.

How Unnest Works

The Unnest operation behaves differently depending on the type of data being unnested:

- For list-type unnesting: It replaces the original key with each individual element from the list.
- For dictionary-type unnesting: It adds new keys to the parent dictionary based on the `expand_fields` parameter.

Unnest does not have an output schema. It modifies the structure of your data in place.

Motivation

The Unnest operation is valuable in scenarios where you need to:

- Process individual items from a list of products in an order
- Analyze separate entries in a list of comments or reviews
- Expand nested data structures for more granular processing
- Flatten complex data structures for easier analysis

Configuration

Required Parameters

Parameter	Description
type	Must be set to "unnest"
name	A unique name for the operation

Parameter	Description
unnest_key	The key of the array field to unnest

Optional Parameters

Parameter	Description	Default
keep_empty	If true, empty arrays being exploded will be kept in the output (with value None)	false
expand_fields	A list of fields to expand from the nested dictionary into the parent dictionary, if unnesting a dict	[]
recursive	If true, the unnest operation will be applied recursively to nested arrays	false
depth	The maximum depth for recursive unnesting (only applicable if recursive is true)	inf
sample	Number of samples to use for the operation	None

Output

The Unnest operation modifies the structure of your data:

- For list-type unnesting: It generates multiple output items for each input item, replacing the original array in the `unnest_key` field with individual elements.
- For dictionary-type unnesting: It expands the specified fields into the parent dictionary.

All other original key-value pairs from the input item are preserved in the output.

Note

When unnesting dictionaries, the original nested dictionary is preserved in the output, and the specified fields are expanded into the parent dictionary.

Use Cases

1. **Product Analysis in Orders**: Unnest a list of products in each order, then use a map operation to analyze each product individually.
2. **Comment Sentiment Analysis**: Unnest a list of comments for each post, enabling sentiment analysis on individual comments.
3. **Nested Data Structure Flattening**: Unnest complex nested data structures to create a flattened dataset for easier analysis or processing.
4. **Processing Time Series Data**: Unnest time series data stored in arrays to analyze individual time points.

Example: Analyzing Product Reviews

Let's walk through an example of using the Unnest operation to prepare product reviews for detailed analysis.

```
- name: extract_salient_quotes
  type: map
  prompt: |
    For the following product review, extract up to 3 salient quotes that best
represent the reviewer's opinion:

    {{ input.review_text }}

    For each quote, provide the text and its sentiment (positive, negative, or
neutral).
  output:
    schema:
      salient_quotes: list[string]

- name: unnest_quotes
  type: unnest
  unnest_key: salient_quotes

- name: analyze_quote
  type: map
  prompt: |
    Analyze the following quote from a product review:

    Quote & information: {{ input.salient_quotes }}
    Review text: {{ input.review_text }}

    Provide a detailed analysis of the quote, including:
    1. The specific aspect of the product being discussed
    2. The strength of the sentiment (-5 to 5, where -5 is extremely negative
and 5 is extremely positive)
    3. Any key terms or phrases that stand out

  output:
    schema:
```

```
        product_aspect: string
        sentiment_strength: number
        key_terms: list[string]
```

This example demonstrates how the Unnest operation fits into a pipeline for analyzing product reviews:

1. The first Map operation extracts salient quotes from each review.
2. The Unnest operation expands the 'salient_quotes' array, creating individual items for each quote. Each quote can now be accessed via `input.salient_quotes`.
3. The second Map operation performs a detailed analysis on each individual quote.

By unnesting the quotes, we enable more granular analysis that wouldn't be possible if we processed the entire review as a single unit.

Advanced Features

Recursive Unnesting

When dealing with deeply nested structures, you can use the `recursive` parameter to apply the unnest operation at multiple levels:

```
- name: recursive_unnest
  type: unnest
  unnest_key: nested_data
  recursive: true
  depth: 3 # Limit recursion to 3 levels deep
```

Dictionary Expansion

When unnesting dictionaries, you can use the `expand_fields` parameter to flatten specific fields into the parent structure:

```
- name: expand_user_data
  type: unnest
  unnest_key: user_info
  expand_fields:
    - name
    - age
    - location
```

In this case, `name`, `age`, and `location` would be added as new keys in the parent dictionary, alongside the original `user_info` key.

Best Practices

1. **Choose the Right Unnest Key**: Ensure you're unnesting the correct field that contains the array or nested structure you want to expand.
2. **Consider Data Volume**: Unnesting can significantly increase the number of items in your data stream. Be mindful of this when designing subsequent operations in your pipeline.
3. **Use Expand Fields Wisely**: When unnesting dictionaries, use the `expand_fields` parameter to flatten your data structure if needed, but be cautious of potential key conflicts.
4. **Handle Empty Arrays**: Decide whether empty arrays should be kept (using `keep_empty`) based on your specific use case and how subsequent operations should handle null values.
5. **Preserve Context**: When unnesting, consider whether you need to carry forward any context from the parent item. The unnest operation preserves all other fields, which helps maintain context.