Extract Operation

Why use Extract instead of Map?

The Extract operation is specifically optimized for isolating portions of source text without synthesis or summarization. Unlike Map operations, which typically transform content, Extract pulls out specific sections verbatim. This provides three key advantages:

1. **Cost efficiency**: Lower output token costs when extracting large chunks of text
2. **Precision**: Extracts exact text without introducing LLM interpretation or potential hallucination
3. **Simplified workflow**: No need to define output schemas - extractions maintain the original text format

The Extract operation identifies and extracts specific sections of text from documents based on provided criteria. It's particularly useful for isolating relevant content from larger documents for further processing or analysis.

Example: Extracting Key Findings from Research Reports

Here's a practical example of using the Extract operation to pull out key findings from research reports:

```yaml
- name: findings
  type: extract
  prompt: |
    Extract all sections that discuss key findings, results, or conclusions
from this research report.
    Focus on paragraphs that:
    - Summarize experimental outcomes
    - Present statistical results
    - Describe discovered insights
    - State conclusions drawn from the research

    Only extract the most important and substantive findings.
  document_keys: ["report_text"]
  model: "gpt-4.1-mini"
```

This operation converts text into a line-numbered format, uses an LLM to identify relevant content, and extracts the specified text ranges. The extracted content is added to the document with the suffix "_extracted_findings".

Sample Input and Output

Input:


```
[
  {
    "report_id": "R-2023-001",
    "report_text": "EXPERIMENTAL METHODS\n\nThe study utilized a mixed-methods
approach combining quantitative surveys (n=230) and qualitative interviews
(n=42) with participants from diverse demographic backgrounds. Data collection
occurred between January and March 2023.\n\nRESULTS\n\nThe analysis revealed
three primary patterns of user engagement. First, 68% of participants reported
daily interaction with the platform, significantly higher than previous
industry benchmarks (p<0.01). Second, user retention showed strong correlation
with personalization features (r=0.72). Finally, demographic factors such as
age and technical proficiency were not significant predictors of engagement,
contradicting prior research in this domain.\n\nDISCUSSION\n\nThese findings
suggest that platform design priorities should emphasize personalization
capabilities over demographic targeting. The high daily engagement rates
indicate market readiness for similar applications, while the lack of
demographic effects points to broad accessibility across user
segments.\n\nLIMITATIONS\n\nThe study was limited by its focus on early
adopters, which may not represent the broader potential user base.
Additionally, the three-month timeframe may not capture seasonal variations in
user behavior."
  }
]
```


Output:


```
[
  {
    "report_id": "R-2023-001",
    "report_text": "EXPERIMENTAL METHODS\n\nThe study utilized a mixed-methods
approach combining quantitative surveys (n=230) and qualitative interviews
(n=42) with participants from diverse demographic backgrounds. Data collection
occurred between January and March 2023.\n\nRESULTS\n\nThe analysis revealed
three primary patterns of user engagement. First, 68% of participants reported
daily interaction with the platform, significantly higher than previous
industry benchmarks (p<0.01). Second, user retention showed strong correlation
with personalization features (r=0.72). Finally, demographic factors such as
age and technical proficiency were not significant predictors of engagement,
contradicting prior research in this domain.\n\nDISCUSSION\n\nThese findings
suggest that platform design priorities should emphasize personalization
capabilities over demographic targeting. The high daily engagement rates
indicate market readiness for similar applications, while the lack of
demographic effects points to broad accessibility across user
segments.\n\nLIMITATIONS\n\nThe study was limited by its focus on early
adopters, which may not represent the broader potential user base.
Additionally, the three-month timeframe may not capture seasonal variations in
user behavior.",
    "report_text_extracted_findings": "The analysis revealed three primary
patterns of user engagement. First, 68% of participants reported daily
interaction with the platform, significantly higher than previous industry
benchmarks (p<0.01). Second, user retention showed strong correlation with
personalization features (r=0.72). Finally, demographic factors such as age and
technical proficiency were not significant predictors of engagement,
contradicting prior research in this domain.\n\nThese findings suggest that
```

```
  platform design priorities should emphasize personalization capabilities over
  demographic targeting. The high daily engagement rates indicate market
  readiness for similar applications, while the lack of demographic effects
  points to broad accessibility across user segments."
    }
  ]
```

Output Formats

The Extract operation offers two output formats controlled by the `format_extraction` parameter:

String Format (Default)

With `format_extraction: true`, extracted text segments are joined with newlines into a single string:

```yaml
- name: findings
  type: extract
  prompt: "Extract the key findings from this research report."
  document_keys: ["report_text"]
  format_extraction: true  # Default setting
```

The resulting output combines all extractions:

```
{
  "report_id": "R-2023-001",
  "report_text": "... original text ...",
  "report_text_extracted_findings": "Finding 1 about daily engagement
rates...\n\nFinding 2 about personalization features..."
}
```

This format works well for human readability, further LLM processing, and when extractions should be treated as a coherent unit.

List Format

With `format_extraction: false`, each extracted text segment remains separate in a list:

```yaml
- name: findings
  type: extract
  prompt: "Extract the key findings from this research report."
  document_keys: ["report_text"]
  format_extraction: false
```

The resulting output preserves each extraction as a distinct item:

```
{
  "report_id": "R-2023-001",
  "report_text": "... original text ...",
  "report_text_extracted_findings": [
    "Finding 1 about daily engagement rates...",
    "Finding 2 about personalization features..."
  ]
}
```

This format is better for individual processing of extractions, counting distinct items, or creating structured data from separate extractions.

Extraction Strategies

The Extract operation offers two main strategies:

Line Number Strategy

This strategy reformats the input text with line numbers, then asks the LLM to identify relevant line ranges. The system extracts those specific ranges, removes line number prefixes, and eliminates duplicates. This works well for extracting multi-line passages or entire sections.

Regex Strategy

This strategy asks the LLM to generate regex patterns matching the desired content. The system applies these patterns to find matches in the original text. This works well for extracting structured data like dates, codes, or specific formatted information.

Required Parameters

- `name`: Unique name for the operation
- `type`: Must be "extract"
- `prompt`: Instructions specifying what content to extract. This does **not** need to be a Jinja template.
- `document_keys`: List of document fields containing text to process

Optional Parameters

Parameter	Description	Default
`model`	Language model to use	Falls back to `default_model`
`extraction_method`	"line_number" or "regex"	"line_number"
`format_extraction`	Join with newlines (`true`) or keep as list (`false`)	`true`
`extraction_key_suffix`	Suffix for output field names	"*extracted*"
`timeout`	Timeout for LLM calls in seconds	120
`skip_on_error`	Continue processing if errors occur	false
`litellm_completion_kwargs`	Additional parameters for LiteLLM calls	{}

Best Practices

Create specific extraction prompts with clear criteria about what content to extract or exclude. Choose the appropriate extraction method based on your needs:

- Use `line_number` for extracting paragraphs, sections, or content spanning multiple lines
- Use `regex` for extracting specific patterns like dates, codes, or formatted data

Process only document fields containing relevant text, and consider enabling `skip_on_error` for batch processing where individual failures shouldn't halt the entire operation.

Format your output based on downstream requirements: - Use the default string format when the extractions form a coherent whole - Use the list format when each extraction needs individual processing or analysis

Use Cases

The Extract operation is valuable for:

- Document summarization - extracting executive summaries or key findings
- Data extraction - isolating dates, measurements, or specific values from text
- Content filtering - pulling relevant sections from lengthy documents
- Evidence gathering - collecting specific statements on given topics
- Preprocessing - creating focused inputs for downstream analysis