Mining Product Reviews: Identifying Polarizing Themes in Video Games

This tutorial demonstrates how to use DocETL to analyze product reviews, specifically focusing on identifying polarizing themes across multiple video games. We'll walk through the process of building a pipeline that extracts insights from Steam game reviews, resolves common themes, and generates comprehensive reports.

Optimization Cost

Optimizing this pipeline can be computationally expensive and time-consuming, especially for large datasets. The process involves running multiple LLM calls and comparisons between different plans, which can result in significant resource usage and potential costs.

For reference, optimizing a pipeline of this complexity could cost up to $70 in OpenAI credits, depending on the size of your dataset and the specific models used. Always monitor your usage and set appropriate limits to avoid unexpected charges.

Task Overview

Our goal is to create a pipeline that will:

1. Identify polarizing themes within individual game reviews
2. Resolve similar themes across different games
3. Generate reports of polarizing themes common across games, supported by quotes from different game reviews

We'll be using a subset of the STEAM review dataset. We've created a subset that contains reviews for 500 of the most popular games, with approximately 400 reviews per game, balanced between positive and negative ratings. For each game, we concatenate all reviews into a single text for analysis---so we'll have 500 input items/reviews, each representing a game. You can get the dataset sample here.

Pipeline Structure

Let's examine the pipeline structure and its operations:

```yaml
pipeline:
  steps:
    - name: game_analysis
      input: steam_reviews
      operations:
        - identify_polarizing_themes
        - unnest_polarizing_themes
        - resolve_themes
        - aggregate_common_themes

  output:
    type: file
    path: "output_polarizing_themes.json"
    intermediate_dir: "intermediates"
```

Full Pipeline Configuration

```yaml
default_model: gpt-4o-mini

system_prompt:
  dataset_description: a collection of reviews for video games
  persona: a marketing analyst analyzing player opinions and themes

datasets:
  steam_reviews:
    type: file
    path: "path/to/top_apps_steam_sample.json"

operations:
  - name: identify_polarizing_themes
    optimize: true
    type: map
    prompt: |
      Analyze the following concatenated reviews for a video game and identify
polarizing themes that divide player opinions. A polarizing theme is one that
some players love while others strongly dislike.

      Game: {{ input.app_name }}
      Reviews: {{ input.concatenated_reviews }}

      For each polarizing theme you identify:
      1. Provide a summary of the theme
      2. Explain why it's polarizing
      3. Include supporting quotes from both positive and negative perspectives

      Aim to identify ~10 polarizing themes, if present.

    output:
      schema:
        polarizing_themes: "list[{theme: str, summary: str,
polarization_reason: str, positive_quotes: str, negative_quotes: str}]"

  - name: unnest_polarizing_themes
    type: unnest
    unnest_key: polarizing_themes
    recursive: true
    depth: 2

  - name: resolve_themes
    type: resolve
    optimize: true
    comparison_prompt: |
      Are the themes "{{ input1.theme }}" and "{{ input2.theme }}" the same?
Here is some context to help you decide:

      Theme 1: {{ input1.theme }}
      Summary 1: {{ input1.summary }}

      Theme 2: {{ input2.theme }}
      Summary 2: {{ input2.summary }}
    resolution_prompt: |
      Given the following themes, please come up with a theme that best
captures the essence of all the themes:
      {% for input in inputs %}
      Theme {{ loop.index }}: {{ input.theme }}
      {% if not loop.last %}
      ---
      {% endif %}
      {% endfor %}

      Based on these themes, provide a consolidated theme that captures the
essence of all the above themes. Ensure that the consolidated theme is concise
yet comprehensive.
    output:
      schema:
        theme: str

  - name: aggregate_common_themes
    type: reduce
    optimize: true
    reduce_key: theme
    prompt: |
      You are given a theme and summary that appears across multiple video
games, along with various apps and review quotes related to this theme. Your
task is to consolidate this information into a comprehensive report.

      For each input, you will receive:
      - theme: A specific polarizing theme
      - summary: A brief summary of the theme
      - app_name: The name of the game
      - positive_quotes: List of supporting quotes from positive perspectives
      - negative_quotes: List of supporting quotes from negative perspectives

      Create a report that includes:
      1. The name of the common theme
      2. A summary of the theme and why it's common across games
      3. Representative quotes from different games, both positive and negative

      Here's the information for the theme:
      Theme: {{ inputs[0].theme }}
      Summary: {{ inputs[0].summary }}

      {% for app in inputs %}
      Game: {{ app.app_name }}
      Positive Quotes: {{ app.positive_quotes }}
      Negative Quotes: {{ app.negative_quotes }}
      {% if not loop.last %}
      ----------------------------------------
      {% endif %}
      {% endfor %}

    output:
      schema:
        theme_summary: str
        representative_quotes: "list[{game: str, quote: str, sentiment: str}]"

pipeline:
  steps:
    - name: game_analysis
      input: steam_reviews
      operations:
        - identify_polarizing_themes
        - unnest_polarizing_themes
        - resolve_themes
```

```
          - aggregate_common_themes

    output:
      type: file
      path: "path/to/output_polarizing_themes.json"
      intermediate_dir: "path/to/intermediates"
```

Pipeline Operations

1. Identify Polarizing Themes

This map operation processes each game's reviews to identify polarizing themes:

```
- name: identify_polarizing_themes
  optimize: true
  type: map
  prompt: |
    Analyze the following concatenated reviews for a video game and identify
polarizing themes that divide player opinions. A polarizing theme is one that
some players love while others strongly dislike.

    Game: {{ input.app_name }}
    Reviews: {{ input.concatenated_reviews }}

    For each polarizing theme you identify:
    1. Provide a summary of the theme
    2. Explain why it's polarizing
    3. Include supporting quotes from both positive and negative perspectives

    Aim to identify ~10 polarizing themes, if present.

  output:
    schema:
      polarizing_themes: "list[{theme: str, summary: str, polarization_reason:
str, positive_quotes: str, negative_quotes: str}]"
```

2. Unnest Polarizing Themes

This operation flattens the list of themes extracted from each game:

```
- name: unnest_polarizing_themes
  type: unnest
  unnest_key: polarizing_themes
  recursive: true
  depth: 2
```

3. Resolve Themes

This operation identifies and consolidates similar themes across different games:

```yaml
- name: resolve_themes
  type: resolve
  optimize: true
  comparison_prompt: |
    Are the themes "{{ input1.theme }}" and "{{ input2.theme }}" the same?
Here is some context to help you decide:

    Theme 1: {{ input1.theme }}
    Summary 1: {{ input1.summary }}

    Theme 2: {{ input2.theme }}
    Summary 2: {{ input2.summary }}
  resolution_prompt: |
    Given the following themes, please come up with a theme that best captures
the essence of all the themes:

    {% for input in inputs %}
    Theme {{ loop.index }}: {{ input.theme }}
    {% if not loop.last %}
    ---
    {% endif %}
    {% endfor %}

    Based on these themes, provide a consolidated theme that captures the
essence of all the above themes. Ensure that the consolidated theme is concise
yet comprehensive.
  output:
    schema:
      theme: str
```

4. Aggregate Common Themes

This reduce operation generates a comprehensive report for each common theme:

```yaml
- name: aggregate_common_themes
  type: reduce
  optimize: true
  reduce_key: theme
  prompt: |
    You are given a theme and summary that appears across multiple video
games, along with various apps and review quotes related to this theme. Your
task is to consolidate this information into a comprehensive report.

    For each input, you will receive:
    - theme: A specific polarizing theme
    - summary: A brief summary of the theme
    - app_name: The name of the game
    - positive_quotes: List of supporting quotes from positive perspectives
    - negative_quotes: List of supporting quotes from negative perspectives

    Create a report that includes:
    1. The name of the common theme
    2. A summary of the theme and why it's common across games
    3. Representative quotes from different games, both positive and negative
```

```
      Here's the information for the theme:
      Theme: {{ inputs[0].theme }}
      Summary: {{ inputs[0].summary }}

      {% for app in inputs %}
      Game: {{ app.app_name }}
      Positive Quotes: {{ app.positive_quotes }}
      Negative Quotes: {{ app.negative_quotes }}
      {% if not loop.last %}
      ----------------------------------------
      {% endif %}
      {% endfor %}

    output:
      schema:
        theme_summary: str
        representative_quotes: "list[{game: str, quote: str, sentiment: str}]"
```

Optimizing the Pipeline

After writing the pipeline, we can use the DocETL `build` command to optimize it:

```
docetl build pipeline.yaml
```

This command, with `optimize: true` set for the map and resolve operations, provides us with:

1. A chunking-based plan for the map operation: This helps handle the large input sizes (up to 380,000 tokens) by breaking them into manageable chunks. The optimizer gives us a chunking plan of 87,776 tokens per chunk, with 10% of the previous chunk as peripheral context.
2. Blocking statements and thresholds for the resolve operation: This optimizes the theme resolution process, making it more efficient when dealing with a large number of themes across multiple games. The optimizer provided us with blocking keys of `summary` and `theme`, and a threshold of 0.596 for similarity (to get 95% recall of duplicates).

These optimizations are crucial for handling the scale of our dataset, which includes 500 games with an *average* of 66,000 tokens per game, and 12% of the items/reviews exceeding the context length limits of the OpenAI LLMs (128k tokens).

Optimized Pipeline

```
default_model: gpt-4o-mini

datasets:
  steam_reviews:
    type: file
    path: "/path/to/steam_reviews_dataset.json"

operations:
  - name: split_identify_polarizing_themes
    type: split
    split_key: concatenated_reviews
    method: token_count
    method_kwargs:
      token_count: 87776
    optimize: false

  - name: gather_concatenated_reviews_identify_polarizing_themes
    type: gather
    content_key: concatenated_reviews_chunk
    doc_id_key: split_identify_polarizing_themes_id
    order_key: split_identify_polarizing_themes_chunk_num
    peripheral_chunks:
      previous:
        tail:
          count: 0.1
    optimize: false

  - name: submap_identify_polarizing_themes
    type: map
    prompt: |
      Analyze the following review snippet from a video game {{ input.app_name
}} and identify any polarizing themes within it. A polarizing theme is one that
diverges opinions among players, where some express strong approval while
others express strong disapproval.

      Review Snippet: {{ input.concatenated_reviews_chunk_rendered }}

      For each polarizing theme you identify:
      1. Provide a brief summary of the theme
      2. Explain why it's polarizing
      3. Include supporting quotes from both positive and negative
perspectives.

      Aim to identify and analyze 3-5 polarizing themes within this snippet.
Only process the main chunk.
    model: gpt-4o-mini
    output:
      schema:
        polarizing_themes: "list[{theme: str, summary: str,
polarization_reason: str, positive_quotes: str, negative_quotes: str}]"
    optimize: false

  - name: subreduce_identify_polarizing_themes
    type: reduce
    reduce_key: ["split_identify_polarizing_themes_id"]
    prompt: |
      Combine the following results and create a cohesive summary of ~10
```

```
polarizing themes for the video game {{ inputs[0].app_name }}:

      {% for chunk in inputs %}
          {% for theme in chunk.polarizing_themes %}
              {{ theme }}
              ----------------------------------------
          {% endfor %}
      {% endfor %}

      Make sure each theme is unique and not a duplicate of another theme. You
should include summaries and supporting quotes (both positive and negative) for
each theme.
    model: gpt-4o-mini
    output:
      schema:
        polarizing_themes: "list[{theme: str, summary: str,
polarization_reason: str, positive_quotes: str, negative_quotes: str}]"
    pass_through: true
    associative: true
    optimize: false
    synthesize_resolve: false

  - name: unnest_polarizing_themes
    type: unnest
    unnest_key: polarizing_themes
    recursive: true
    depth: 2

  - name: resolve_themes
    type: resolve
    blocking_keys:
      - summary
      - theme
    blocking_threshold: 0.596
    optimize: true
    comparison_prompt: |
      Are the themes "{{ input1.theme }}" and "{{ input2.theme }}" the same?
Here is some context to help you decide:

      Theme 1: {{ input1.theme }}
      Summary 1: {{ input1.summary }}

      Theme 2: {{ input2.theme }}
      Summary 2: {{ input2.summary }}
    resolution_prompt: |
      Given the following themes, please come up with a theme that best
captures the essence of all the themes:

      {% for input in inputs %}
      Theme {{ loop.index }}: {{ input.theme }}
      {% if not loop.last %}
      ---
      {% endif %}
      {% endfor %}

      Based on these themes, provide a consolidated theme that captures the
essence of all the above themes. Ensure that the consolidated theme is concise
yet comprehensive.
    output:
      schema:
        theme: str
```

```yaml
  - name: aggregate_common_themes
    type: reduce
    reduce_key: theme
    prompt: |
      You are given a theme and summary that appears across multiple video
games, along with various apps and review quotes related to this theme. Your
task is to consolidate this information into a comprehensive report.

      For each input, you will receive:
      - theme: A specific polarizing theme
      - summary: A brief summary of the theme
      - app_name: The name of the game
      - positive_quotes: List of supporting quotes from positive perspectives
      - negative_quotes: List of supporting quotes from negative perspectives

      Create a report that includes:
      1. The name of the common theme
      2. A summary of the theme and why it's common across games
      3. Representative quotes from different games, both positive and negative

      Here's the information for the theme:
      Theme: {{ inputs[0].theme }}
      Summary: {{ inputs[0].summary }}

      {% for app in inputs %}
      Game: {{ app.app_name }}
      Positive Quotes: {{ app.positive_quotes }}
      Negative Quotes: {{ app.negative_quotes }}
      {% if not loop.last %}
      ----------------------------------------
      {% endif %}
      {% endfor %}

    output:
      schema:
        theme_summary: str
        representative_quotes: "list[{game: str, quote: str, sentiment: str}]"

pipeline:
  steps:
    - name: game_analysis
      input: steam_reviews
      operations:
        - split_identify_polarizing_themes
        - gather_concatenated_reviews_identify_polarizing_themes
        - submap_identify_polarizing_themes
        - subreduce_identify_polarizing_themes
        - unnest_polarizing_themes
        - resolve_themes
        - aggregate_common_themes

  output:
    type: file
    path: "/path/to/output/output_polarizing_themes.json"
    intermediate_dir: "/path/to/intermediates"
```

Running the Pipeline

With our optimized pipeline in place, we can now run it:

```
docetl run pipeline.yaml
```

This command will process the game reviews, extract polarizing themes, resolve similar themes across games, and generate comprehensive reports for each common theme. The results will be saved in output_polarizing_themes.json, providing insights into the polarizing aspects of various video games based on user reviews.

The output costs for running this pipeline will depend on the size of the dataset and the specific models used. We used gpt-4o-mini and had ~200,000 reviews we were aggregating. Here's the logs from my terminal:

```
docetl run workloads/steamgames/pipeline_opt.yaml
[11:05:46] Performing syntax check on all operations...
           Syntax check passed for all operations.
           Running Operation:
             Type: split
             Name: split_identify_polarizing_themes
[11:06:08] Intermediate saved for operation 'split_identify_polarizing_themes'
in step 'game_analysis'
           Running Operation:
             Type: gather
             Name: gather_concatenated_reviews_identify_polarizing_themes
[11:06:10] Intermediate saved for operation
'gather_concatenated_reviews_identify_polarizing_themes' in step
'game_analysis'
           Running Operation:
             Type: map
             Name: submap_identify_polarizing_themes
⠙ Running step game_analysis...
[11:06:14] Intermediate saved for operation
'submap_identify_polarizing_themes' in step 'game_analysis'
           Running Operation:
             Type: reduce
             Name: subreduce_identify_polarizing_themes
⠴ Running step game_analysis...
[11:06:16] Intermediate saved for operation
'subreduce_identify_polarizing_themes' in step 'game_analysis'
           Running Operation:
             Type: unnest
             Name: unnest_polarizing_themes
[11:06:25] Intermediate saved for operation 'unnest_polarizing_themes' in step
'game_analysis'
           Running Operation:
             Type: resolve
             Name: resolve_themes
[11:06:37] Comparisons saved by blocking: 6802895 (97.50%)
⠦ Running step game_analysis...
[13:05:58] Number of keys before resolution: 3736
           Number of distinct keys after resolution: 1421
⠹ Running step game_analysis...
[13:06:23] Self-join selectivity: 0.1222
```

```
[13:06:36] Intermediate saved for operation 'resolve_themes' in step
'game_analysis'
           Running Operation:
             Type: reduce
             Name: aggregate_common_themes
⠼ Running step game_analysis...
[13:08:05] Intermediate saved for operation 'aggregate_common_themes' in step
'game_analysis'
           Flushing cache to disk...
           Cache flushed to disk.
  Step game_analysis completed. Cost: $13.21
  Operation split_identify_polarizing_themes completed. Cost: $0.00
  Operation gather_concatenated_reviews_identify_polarizing_themes completed.
Cost: $0.00
  Operation submap_identify_polarizing_themes completed. Cost: $5.02
  Operation subreduce_identify_polarizing_themes completed. Cost: $0.38
  Operation unnest_polarizing_themes completed. Cost: $0.00
  Operation resolve_themes completed. Cost: $7.56
  Operation aggregate_common_themes completed. Cost: $0.26
           💾 Output saved to output_polarizing_themes.json
           Total cost: $13.21
           Total time: 7339.11 seconds
```

Upon further analysis, 1421 themes is still a lot! I realized that my resolve operation was not exactly what I wanted---it did not merge together themes that I believed were similar, since the comparison prompt only asked if theme X or Y were the same. I should have given context in the comparison prompt, such as "Could one of these themes be a subset of the other?" This underscores the iterative nature of pipeline development and the importance of refining prompts to achieve the desired results; we don't really know what the desired results are until we see the output.

Something else we could have done is included a list of themes we care about in the original map operation, e.g., graphics. Since our map prompt was very open-ended, the LLM could have generated themes that we didn't care about, leading to a large number of themes in the output.

Anyways, we've filtered the 1421 reports down to 65 themes/reports that contain quotes from 3 or more different games. You can check out the output here.