Presidential Debate Themes Analysis

This tutorial explains how to analyze themes in presidential debates using the DocETL pipeline. We'll cover the pipeline structure, explain each operation, and discuss the importance of theme resolution.

Pipeline Overview

Our goal is to build a pipeline that will:

1. Extract key themes and viewpoints from presidential debate transcripts
2. Analyze how these themes have evolved over time, with references to specific debates and quotes

You can take a look at the raw data here.

Let's examine the pipeline structure and its operations:

```yaml
pipeline:
  steps:
    - name: debate_analysis
      input: debates
      operations:
        - extract_themes_and_viewpoints
        - unnest_themes
        - summarize_theme_evolution
```

Full Pipeline Configuration

```yaml
datasets:
  debates:
    type: file
    path: "data.json"

system_prompt:
  dataset_description: a collection of transcripts of presidential debates
  persona: a political analyst

default_model: gpt-4o-mini

operations:
  - name: extract_themes_and_viewpoints
    type: map
    output:
      schema:
        themes: "list[{theme: str, viewpoints: str}]"
    prompt: |
      Analyze the following debate transcript for {{ input.title }} on {{
input.date }}:

      {{ input.content }}

      Extract the main themes discussed in this debate and the viewpoints of
the candidates on these themes.
      Return a list of themes and corresponding viewpoints (including the
specific quotes from the debate) in the following format:
      [
        {
          "theme": "Theme 1",
          "viewpoints": "Candidate A's viewpoint... Candidate B's viewpoint..."
        },
        {
          "theme": "Theme 2",
          "viewpoints": "Candidate A's viewpoint... Candidate B's viewpoint..."
        },
        ...
      ]

  - name: unnest_themes
    type: unnest
    unnest_key: themes
    recursive: true

  - name: summarize_theme_evolution
    type: reduce
    reduce_key: theme
    output:
      schema:
        theme: str
        report: str
    prompt: |
      Analyze the following viewpoints on the theme "{{ inputs[0].theme }}"
from various debates over the years:

      {% for item in inputs %}
      Year: {{ item.year }}
```

```
      Date: {{ item.date }}
      Title: {{ item.title }}
      Viewpoints: {{ item.viewpoints }}

      {% endfor %}

      Generate a comprehensive summary of how Democratic and Republican
viewpoints on this theme have evolved through the years. Include supporting
quotes from the debates to illustrate key points or shifts in perspective.

      Your summary should:
      1. Identify *all* major trends or shifts in each party's stance over time
      2. Highlight any significant agreements or disagreements between the
parties
      3. Note any external events or factors that may have influenced changes
in viewpoints
      4. Use specific quotes to support your analysis
      5. The title should contain the start and end years of the analysis

      Format your response as a well-structured report.

pipeline:
  steps:
    - name: debate_analysis
      input: debates
      operations:
        - extract_themes_and_viewpoints
        - unnest_themes
        - summarize_theme_evolution

  output:
    type: file
    path: "theme_evolution_analysis.json"
    intermediate_dir: "checkpoints"
```

Pipeline Operations

1. Extract Themes and Viewpoints

```
- name: extract_themes_and_viewpoints
  type: map
  output:
    schema:
      themes: "list[{theme: str, viewpoints: str}]"
  prompt: |
    Analyze the following debate transcript for {{ input.title }} on {{
input.date }}:

    {{ input.content }}

    Extract the main themes discussed in this debate and the viewpoints of the
candidates on these themes.
    Return a list of themes and corresponding viewpoints in the following
format:
    [
      {
```

```
        "theme": "Theme 1",
        "viewpoints": "Candidate A's viewpoint... Candidate B's viewpoint..."
      },
      {
        "theme": "Theme 2",
        "viewpoints": "Candidate A's viewpoint... Candidate B's viewpoint..."
      },
      ...
    ]
```

This operation processes each debate transcript to identify main themes and candidates' viewpoints. It uses AI to analyze the content and structure the output in a consistent format.

2. Unnest Themes

```yaml
- name: unnest_themes
  type: unnest
  unnest_key: themes
  recursive: true
```

The unnest operation flattens the list of themes extracted from each debate. This step prepares the data for further analysis by creating individual entries for each theme.

3. Summarize Theme Evolution

```yaml
- name: summarize_theme_evolution
  type: reduce
  reduce_key: theme
  output:
    schema:
      theme: str
      report: str
  prompt: |
    Analyze the following viewpoints on the theme "{{ inputs[0].theme }}" from
various debates over the years:

    {% for item in inputs %}
    Year: {{ item.year }}
    Date: {{ item.date }}
    Title: {{ item.title }}
    Viewpoints: {{ item.viewpoints }}

    {% endfor %}

    Generate a comprehensive summary of how Democratic and Republican
viewpoints on this theme have evolved through the years. Include supporting
quotes from the debates to illustrate key points or shifts in perspective.

    Your summary should:
    1. Identify all major trends or shifts in each party's stance over time
```

```
    2. Highlight any significant agreements or disagreements between the
parties
    3. Note any external events or factors that may have influenced changes in
viewpoints
    4. Use specific quotes to support your analysis
    5. The title should contain the start and end years of the analysis

    Format your response as a well-structured report.
```

This operation analyzes how each theme has evolved over time. It considers viewpoints from multiple debates, identifies trends, and generates a comprehensive summary of the theme's evolution.

The Need for Theme Resolution

An important consideration in this pipeline is the potential for similar themes to be generated with slightly different wording (e.g., "Climate Change Policy" vs. "Environmental Regulations"). To address this, we need to add a resolve operation before the summarization step.

To synthesize a resolve operation, we can use the `docetl build` command:

```
docetl build pipeline.yaml
```

This command adds a resolve operation to our pipeline, resulting in an optimized version:

```yaml
operations:
    ...
    - name: synthesized_resolve_0
      type: resolve
      blocking_keys:
        - theme
      blocking_threshold: 0.6465
      comparison_model: gpt-4o-mini
      comparison_prompt: |
        Compare the following two debate themes:

        [Entity 1]:
        {{ input1.theme }}

        [Entity 2]:
        {{ input2.theme }}

        Are these themes likely referring to the same concept? Consider the
following attributes:
        - The core subject matter being discussed
        - The context in which the theme is presented
        - The viewpoints of the candidates associated with each theme

        Respond with "True" if they are likely the same theme, or "False" if
they are likely different themes.
```

```
      embedding_model: text-embedding-3-small
      compare_batch_size: 1000
      output:
        schema:
          theme: string
      resolution_model: gpt-4o-mini
      resolution_prompt: |
        Analyze the following duplicate themes:

        {% for key in inputs %}
        Entry {{ loop.index }}:
        {{ key.theme }}

        {% endfor %}

        Create a single, consolidated key that combines the information from
all duplicate entries. When merging, follow these guidelines:
        1. Prioritize the most comprehensive and detailed viewpoint available
among the duplicates. If multiple entries discuss the same theme with varying
details, select the entry that includes the most information.
        2. Ensure clarity and coherence in the merged key; if key terms or
phrases are duplicated, synthesize them into a single statement or a cohesive
description that accurately represents the theme.

        Ensure that the merged key conforms to the following schema:
        {
          "theme": "string"
        }

        Return the consolidated key as a single JSON object.


pipeline:
  steps:
    - name: debate_analysis
      input: debates
      operations:
        - extract_themes_and_viewpoints
        - unnest_themes
        - synthesized_resolve_0
        - summarize_theme_evolution
```

The new `synthesized_resolve_0` operation groups similar themes together, ensuring a more accurate and comprehensive analysis of each theme's evolution.

Running the Optimized Pipeline

With the resolve operation in place, we can now run our optimized pipeline:

```
docetl run pipeline_opt.yaml
```

This command processes the debate transcripts, extracts themes, resolves similar themes, and generates summaries of theme evolution over time. The results will be

saved in `theme_evolution_analysis.json`, providing insights into the changing landscape of topics discussed in presidential debates. Since we've also set an `intermediate_dir` in our pipeline configuration, intermediate results will be saved in the `intermediate_dir` directory.

Here's the output from running our optimized pipeline:

```
$ docetl run pipeline_opt.yaml
[09:28:17] Performing syntax check on all operations...
           Syntax check passed for all operations.
           Running Operation:
             Type: map
             Name: extract_themes_and_viewpoints
⠸ Running step debate_analysis...
[09:28:36] Intermediate saved for operation 'extract_themes_and_viewpoints'
           Running Operation:
             Type: unnest
             Name: unnest_themes
           Intermediate saved for operation 'unnest_themes'
           Running Operation:
             Type: resolve
             Name: synthesized_resolve_0
[09:28:38] Comparisons saved by blocking: 56002 (97.75%)
⠼ Running step debate_analysis...
[09:29:02] Number of keys before resolution: 339
           Number of distinct keys after resolution: 152
⠼ Running step debate_analysis...
[09:29:04] Self-join selectivity: 0.0390
           Intermediate saved for operation 'synthesized_resolve_0'
           Running Operation:
             Type: reduce
             Name: summarize_theme_evolution
⠸ Running step debate_analysis...
[09:29:54] Intermediate saved for operation 'summarize_theme_evolution'
           Flushing cache to disk...
           Cache flushed to disk.
  Step debate_analysis completed. Cost: $0.29
  Operation extract_themes_and_viewpoints completed. Cost: $0.16
  Operation unnest_themes completed. Cost: $0.00
  Operation synthesized_resolve_0 completed. Cost: $0.04
  Operation summarize_theme_evolution completed. Cost: $0.09
           💾 Output saved to theme_evolution_analysis_baseline.json
           Total cost: $0.29
           Total time: 97.25 seconds
```

This output shows the progress of our pipeline execution, including the different operations performed, intermediate saves, and the final results. Note the total cost was only $0.29!

Initial Results

Our pipeline generated reports on various themes discussed in the presidential debates. We've put the results up here. However, upon inspection, we found that these reports were lacking in depth and recency. Let's look at a few examples:

Example Reports Lacking in Recent Quotes

Infrastructure Development

```
# Infrastructure Development: A Comparative Analysis of Democratic and
Republican Viewpoints from 1992 to 2023

## Introduction
Infrastructure development has long been a pivotal theme in American political
discourse, with varying perspectives presented by major party candidates. This
report analyzes shifts and trends in Democratic and Republican viewpoints from
1992, during the second presidential debate between George Bush, Bill Clinton,
and Ross Perot, to 2023.

## Republican Viewpoints
### Early 1990s
In 1992, President George Bush emphasized a forward-looking approach to
infrastructure, stating, "We passed this year the most furthest looking
transportation bill in the history of this country...$150 billion for improving
the infrastructure." This statement indicated a commitment to substantial
federal investment in infrastructure aimed at enhancing transportation
networks.

### 2000s
Moving into the early 2000s, the Republican party maintained a focus on
infrastructure but began to frame it within the context of economic growth and
public-private partnerships. However, after the 2008 financial crisis, there
was a noticeable shift. The party emphasized tax cuts and reducing regulation
over large public investments in infrastructure.

### Recent Years
By 2020 and 2021, under the Trump administration, the emphasis returned to
infrastructure. However, the tone shifted towards emphasizing private sector
involvement and deregulation rather than large public spending. The Republican
approach became more fragmented, with some factions calling for aggressive
infrastructure investment, while others remained cautious about expenditures.

## Democratic Viewpoints
### Early 1990s
In contrast, Governor Bill Clinton in 1992 proposed a more systematic
investment strategy, noting, "My plan would dedicate $20 billion a year in each
of the next 4 years for investments in new transportation." This highlighted a
stronger emphasis on direct federal involvement in infrastructure as a means of
fostering economic opportunity and job creation.

### 2000s
Through the late 1990s and early 2000s, the Democratic party continued to push
for comprehensive federal infrastructure plans, often attached to broader
economic initiatives aimed at reducing inequality and spurring job growth. The
party emphasized sustainable infrastructure and investments that address
climate change.

### Recent Years
By 2020, under the Biden administration, the Democrat viewpoint strongly
advocated for significant infrastructure investments, combining traditional
infrastructure with climate resilience. The American Jobs Plan symbolized this
shift, proposing vast funds for transit systems, renewable energy projects, and
rural broadband internet. The framing increasingly included social equity as a
```

```
core component of infrastructure, influenced by movements advocating for racial
and economic justice.

## Agreements and Disagreements
### Agreements
Despite inherent differences, both parties have historically acknowledged the
necessity of infrastructure investments for economic growth. Both Bush and
Clinton in 1992 recognized infrastructure as vital for job creation, but
diverged on the scope and funding mechanisms.

### Disagreements
Over the years, major disagreements have surfaced, particularly in funding
approaches. The Republican party has increasingly favored private sector
involvement and tax incentives, while Democrats have consistently pushed for
robust federal spending and the incorporation of progressive values into
infrastructure projects.

## Influencing Factors
The evolution of viewpoints has often mirrored external events such as economic
recessions, technological advancement, and social movements. The post-9/11 era
and the 2008 financial crisis notably shifted priorities, with bipartisan
discussions centered around recovery through infrastructure spending.
Additionally, increasing awareness of climate change and social justice has
over the years significantly influenced Democratic priorities, leading to a
more inclusive and sustainable approach to infrastructure development.

## Conclusion
The comparative analysis of Democratic and Republican viewpoints on
infrastructure development from 1992 to 2023 reveals significant shifts in
priorities and strategies. While both parties agree on the need for
infrastructure improvements, their approaches and underlying philosophies
continue to diverge, influenced by economic, social, and environmental factors.
```

Crime and Gun Control

```
## The Evolution of Democratic and Republican Viewpoints on Crime and Gun
Control: 1992-2023

### Introduction
This report analyzes the shifting perspectives of the Democratic and Republican
parties on the theme of "Crime and Gun Control" from 1992 to 2023. The
exploration encompasses key debates, significant shifts in stance, party
alignments, and influences from external events that shaped these viewpoints.

### Democratic Party Viewpoints
1. **Initial Stance (1992)**: In the early 1990s, the Democratic viewpoint, as
exemplified by Governor Bill Clinton during the Second Presidential Debate in
1992, supported individual gun ownership but emphasized the necessity of
regulation: "I support the right to keep and bear arms...but I believe we have
to have some way of checking handguns before they're sold."
- **Trend**: This reflects a moderate position seeking to balance gun rights
with public safety—a common theme in Democratic rhetoric during this era that
resonated with many constituents.

2. **Shift Towards Stricter Gun Control (Late 1990s - 2000s)**: Following
events such as the Columbine High School shooting in 1999, the Democratic Party
increasingly advocated for more stringent gun control measures. The passing of
the Brady Bill and the Assault Weapons Ban in 1994 marked a peak in regulatory
measures supported by Democrats, emphasizing public safety over gun ownership
rights.
```

```
- **Quote Impact**: During this time, Democratic leaders often highlighted the need for legislative action to combat rising gun violence.

3. **Response to Mass Shootings (2010s)**: The tragic events of the Sandy Hook Elementary School shooting in 2012 ignited a renewed push for gun control from leading Democrats, including then-President Barack Obama. His call for "common-sense gun laws" marked a decisive moment in Democrat advocacy, focusing on background checks and bans on assault weapons.
- **Quote**: Obama stated, "We can't tolerate this anymore. These tragedies must end. And to end them, we must change."

4. **Current Stance (2020s)**: The Democratic viewpoint has continued to become increasingly aligned with comprehensive gun control measures, including calls for universal background checks and red flag laws. In the wake of ongoing gun violence, this approach highlights a commitment to addressing systemic issues related to crime and public safety.

### Republican Party Viewpoints
1. **Consistent Support for Gun Rights (1992)**: In the same 1992 debate, President George Bush emphasized the rights of gun owners, stating, "I'm a sportsman and I don't think you ought to eliminate all kinds of weapons." This illustrates a steadfast commitment to Second Amendment rights that has characterized Republican positions over the years.
- **Trend**: The Republican Party has traditionally promoted a pro-gun agenda, often resisting calls for stricter regulations or bans.

2. **Response to Gun Control Advocacy (2000s)**: As Democrats pushed for stricter gun laws, Republicans increasingly framed these measures as infringements on individual rights. The response to high-profile shootings tended to focus on mental health and crime prevention rather than gun regulation.
- **Disagreement**: Republicans consistently argued against the effectiveness of gun control, indicating belief in personal responsibility and the right to self-defense.

3. **Shift to Increased Resistance (2010s)**: In the wake of prominent mass shootings, the Republican party maintained its focus on supporting gun rights, opposing federal gun control initiatives. Notable figures, such as former NRA spokesperson Dana Loesch, articulated this resistance by stating, "We are not going to let tragedies be used to violate our rights."
- **Impact of External Events**: The rise of organizations like the NRA and increased gun ownership among constituents have fortified this pro-gun stance.

4. **Contemporary Stance (2020s)**: Currently, the Republican viewpoint remains largely unchanged with an emphasis on individual rights to bear arms and skepticism regarding the effectiveness of gun control laws. Recent discussions around gun violence often focus on addressing crime through law enforcement and community safety programs instead of legislative gun restrictions.

### Key Agreements and Disagreements
- **Common Ground**: Both parties, at different times, have recognized the necessity for addressing gun-related violence but diverge on methods—Democrats typically advocate for regulations while Republicans focus on rights preservation.
- **Disagreements**: A significant divide exists on whether stricter gun laws equate to reduced crime rates, with Republicans consistently refuting this correlation, arguing instead that law-abiding citizens need access to firearms for self-defense.

### Conclusion
The evolution of viewpoints on crime and gun control from 1992 to 2023
```

highlights a pronounced divergence between the Democratic and Republican parties. While Democrats have increasingly pursued stricter regulatory measures focused on public safety, Republicans have maintained a consistent advocacy for gun rights, underscoring a broader ideological conflict over individual freedoms and collective responsibility for public safety. The trajectories of both parties reflect their core values and responses to notable events impacting society.

Drug Policy

```
## Evolution of Drug Policy Viewpoints: 1996 to 2023

### Summary of Democratic and Republican Perspectives on Drug Policy
Over the years, drug policy has been a contentious issue in American politics, reflecting profound changes within both the Democratic and Republican parties. This report examines how views have evolved, highlighting significant trends, areas of agreement and disagreement, and influential external factors utilizing debates as primary reference points.

### Democratic Party Trends:
1. **Increased Emphasis on Comprehensive Approaches**:
   - In the 1996 Clinton-Dole debate, President Bill Clinton stated, "We have dramatically increased control and enforcement at the border." This reflects a focus on enforcement as part of drug policy. However, Clinton's later years signaled a shift towards recognizing the need for treatment and prevention.

2. **Emergence of Harm Reduction and Decriminalization**:
   - Moving into the 2000s and beyond, Democrats began to embrace harm reduction strategies. For instance, during the 2020 Democratic primary debates, candidates such as Bernie Sanders and Elizabeth Warren emphasized decriminalization and treatment over incarceration, signifying a notable shift from punitive measures.

3. **Growing Advocacy for Social Justice**:
   - Recent years have seen an alignment with social justice movements, arguing that drug policy disproportionately affects marginalized communities. Kamala Harris in 2020 stated, "We need to truly decriminalize marijuana and address the impact of the War on Drugs on communities of color."

### Republican Party Trends:
1. **Strong Focus on Law and Order**:
   - During the 1996 debate, Senator Bob Dole reflected a traditional Republican stance, highlighting concerns over drugs without markedly addressing the social implications. "The President doesn't want to politicize drugs, but it's already politicized Medicare..." displays a defensive posture toward the political ramifications of drug issues.

2. **Shift Towards Treatment and Prevention**:
   - By the mid-2010s, there was a growing recognition of the opioid crisis, leading to a bipartisan approach promoting treatment. For example, former President Donald Trump, in addressing the opioid epidemic, stated, "We have to take care of our people. We can't just lock them up."

3. **Conflict Between Hardline Stance and Pragmatism**:
   - Despite some shifts, many Republicans still emphasize law enforcement solutions. This tension was evident in the polarizing responses to marijuana legalization across states, with figures like former Attorney General Jeff Sessions taking a hardline stance against marijuana legalization, contrasting with more progressive approaches adopted by some Republican governors.
```

```
### Areas of Agreement:
1. **Opioid Crisis**:
   - Both parties acknowledged the severity of the opioid epidemic, leading to
legislation aimed at addressing addiction and treatment, indicating a rare
consensus on the need for health-focused solutions.

### Areas of Disagreement:
1. **Approach to Drug Policy**:
   - The Democratic party's shift towards decriminalization and harm reduction
contrasts sharply with segments of the Republican party that still advocate for
strict enforcement and criminalization of certain drugs.

### Influential External Events and Factors:
1. **The Opioid Crisis**:
   - The rise of the opioid epidemic has forced both parties to reevaluate
their positions on drug policy, pushing them towards more compassionate
approaches focusing on addiction treatment.

2. **Social Justice Movements**:
   - The Black Lives Matter movement and other social justice efforts have
altered the discourse surrounding drug policies, with increased focus on the
need to rectify injustices in enforcement practices, particularly among
minorities.

### Conclusion:
Through the years, drug policy viewpoints within the Democratic and Republican
parties have experienced significant evolution, characterized by complex layers
of agreement and disagreement. As social dynamics shift, both parties continue
to grapple with finding a balanced approach towards a more effective drug
policy that prioritizes health and social justice.
```

Upon inspecting the intermediates, it appears that the map operation is doing a good job at extracting relevant information. The issue seems to lie in the reduce operation, which is ignoring some of the analysis.

It's possible that trying to summarize all the insights across all debates for a topic in a single LLM call is too ambitious. To address this, we can set `optimize: true` for the reduce operation.

Let's update our `summarize_theme_evolution` operation in the pipeline:

```yaml
- name: summarize_theme_evolution
  type: reduce
  reduce_key: theme
  optimize: true
  output:
    schema:
      theme: str
      report: str
  prompt: |
    [... existing prompt ...]
```

Rerunning the build command `docetl build pipeline.yaml` will synthesize the optimized reduce operation (make sure `pipeline.yaml` is the pipeline you want to optimize).

Running the Pipeline (with Optimized Reduce)

In our optimized pipeline, we see that DocETL added a `gleaning` configuration to the reduce operation:

```
- name: summarize_theme_evolution
  type: reduce
  reduce_key: theme
  ...
  gleaning:
    num_rounds: 1
    validation_prompt: |
        1. Does the output adequately summarize the evolution of viewpoints on
the theme based on the
        provided debate texts? Are all critical shifts and trends mentioned?
        2. Are there any crucial quotes or data points missing from the output
that
        were present in the debate transcripts that could reinforce the
analysis?
        3. Is the output well-structured and easy to follow, following any
        formatting guidelines specified in the prompt, such as using headings
for
        sections or maintaining a coherent narrative flow?
```

Tip

Note that you should always feel free to edit the `validation_prompt` to better suit your specific needs! The optimizer uses LLMs to write all prompts, but you have the best context on your task and what you're looking for in the output, so you should adjust anything accordingly.

And when running the pipeline, we can observe the impact of this optimization; for example, one of the outputs gets amended to include more recent quotes:

```
Validator improvements (gleaning round 1):
1. **Coverage of Critical Shifts and Trends**: While the output captures
several significant trends and shifts in Democratic and Republican viewpoints,
it could benefit from a more thorough overview, especially about the changing
perceptions and proposals related to economic downturns and recoveries across
the decades. For instance, it could include how the response to the 2008
financial crisis tied back to historical precedents, linking back to earlier
debates about the importance of jobs and middle-class stability (like in the
1976 or 1992 debates).

2. **Missing Quotes and Data Points**: The response could further bolster the
analysis by incorporating additional quotes that illustrate the evolving
narrative, particularly surrounding tax fairness and job creation. For
example, quotes like George Bush in 1984 calling out "working man"
perspectives against seemingly progressive taxation could enhance the depth.
Additionally, quotes from debates emphasizing the impact of tax cuts on
```

```
economic recovery and job growth—such as Obama's focus on the automotive
industry's recovery in 2012 or McCain's 'putting homeowners first'—could
provide essential context to the arguments for both parties.

3. **Structure and Flow**: Overall, the output is fairly well-structured and
maintains a logical flow, using headings appropriately to signal key sections.
However, it may benefit from clearer subsections under each party's overview
to delineate specific key points, such as 'Tax Policy', 'Job Creation', and
'Response to Economic Crises'. This would enhance readability and assist the
reader in navigating the shifts in viewpoints. For example, adding bullet
points or more vivid transitions between historical periods could clarify the
evolution timeline. Moreover, resolving any redundancy (such as multiple
mentions of similar themes across years) would streamline the narrative.
```

Check out the new output here to see the improvements made by the optimized pipeline! Of course, we can probably optimize the initial map operation too, do prompt engineering, and more to further enhance the pipeline.

Interactive Pipeline Creation

We're currently building interfaces to interactively create and iterate on these pipelines after seeing outputs. This will allow for more intuitive pipeline development and refinement based on real-time results. If you're interested in this feature or would like to provide input, please reach out to us! Your feedback can help shape the future of DocETL's user experience.

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:

```
- 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:

```
- 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

```yaml
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
```

```
  - 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.

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/.

Split and Gather Example: Analyzing Trump Immunity Case

This example demonstrates how to use the Split and Gather operations in DocETL to process and analyze a large legal document, specifically the government's motion for immunity determinations in the case against former President Donald Trump. You can download the dataset we'll be using here. This dataset contains a single document.

Problem Statement

We want to analyze a lengthy legal document to identify all people involved in the Trump vs. United States case regarding presidential immunity. The document is too long to process in a single operation, so we need to split it into manageable chunks and then gather context to ensure each chunk can be analyzed effectively.

Chunking Strategy

When dealing with long documents, it's often necessary to break them down into smaller, manageable pieces. This is where the Split and Gather operations come in handy:

1. **Split Operation**: This divides the document into smaller chunks based on token count or delimiters. For legal documents, using a token count method is often preferable to ensure consistent chunk sizes.
2. **Gather Operation**: After splitting, we use the Gather operation to add context to each chunk. This operation can include content from surrounding chunks, as well as document-level metadata and headers, ensuring that each piece maintains necessary context for accurate analysis.

Pipeline Overview

Our pipeline will follow these steps:

1. Extract metadata from the full document
2. Split the document into chunks
3. Extract headers from each chunk
4. Gather context for each chunk
5. Analyze each chunk to identify people and their involvements in the case
6. Reduce the results to compile a comprehensive list of people and their roles

Example Pipeline and Output

Here's a breakdown of the pipeline defined in trump-immunity_opt.yaml:

1. **Dataset Definition**: We define a dataset (json file) with a single document.
2. **Metadata Extraction**: We define a map operation to extract any document-level metadata that we want to pass to each chunk being processed.
3. **Split Operation**: The document is split into chunks using the following configuration:

```yaml
- name: split_find_people_and_involvements
  type: split
  method: token_count
  method_kwargs:
    num_tokens: 3993
  split_key: extracted_text
```

   This operation splits the document into chunks of approximately 3993 tokens each. This size is chosen to balance between maintaining context and staying within model token limits. `split_key` should be the field in the document that contains the text to split.

4. **Header Extraction**: We define a map operation to extract headers from each chunk. Then, when rendering each chunk, we can also render the headers in levels above the headers in the chunk--ensuring that we can maintain hierarchical context, even when the headers are in other chunks.
5. **Gather Operation**: Context is gathered for each chunk using the following configuration:

```yaml
- name: gather_extracted_text_find_people_and_involvements
  type: gather
  content_key: extracted_text_chunk (1)
  doc_header_key: headers (2)
```

```
    doc_id_key: split_find_people_and_involvements_id 3
    order_key: split_find_people_and_involvements_chunk_num 4
    peripheral_chunks:
      next:
        head:
          count: 1
      previous:
        tail:
          count: 1
```

1. The field containing the chunk content; the split_key with "_chunk" appended. Automatically exists as a result of the split operation. **This is required.**
2. The field containing the extracted headers for each chunk. Only exists if you have a header extraction map operation. **This can be omitted if you don't have headers extracted for each chunk.**
3. The unique identifier for each document; the split operation name with "_id" appended. Automatically exists as a result of the split operation. **This is required.**
4. The field indicating the order of chunks; the split operation name with "_chunk_num" appended. Automatically exists as a result of the split operation. **This is required.**

This operation gathers context for each chunk, including the previous chunk, the current chunk, and the next chunk. We also render the headers populated by the previous operation.

6. **Chunk Analysis**: We define a map operation to analyze each chunk.
7. **Result Reduction**: We define a reduce operation to reduce the results of the map operation (applied to each chunk) to a single list of people and their involvements in the case.

Here is the full pipeline configuration, with the split and gather operations highlighted. Assuming the sample dataset looks like this:

```
[
  {
    "pdf_url":
"https://storage.courtlistener.com/recap/gov.uscourts.dcd.258148/gov.uscourts.
dcd.258148.252.0.pdf"
  }
]
```

Full Pipeline Configuration

```yaml
datasets:
  legal_doc:
    type: file
    path: /path/to/your/dataset.json
    parsing:
      - function: azure_di_read
        input_key: pdf_url
        output_key: extracted_text
        function_kwargs:
          use_url: true
          include_line_numbers: true

default_model: gpt-4o-mini

system_prompt:
  dataset_description: the Trump vs. United States case
  persona: a legal analyst

operations:
  - name: extract_metadata_find_people_and_involvements
    type: map
    model: gpt-4o-mini
    prompt: |
      Given the document excerpt: {{ input.extracted_text }}
      Extract all the people mentioned and summarize their involvements
in the case described.
    output:
      schema:
        metadata: str

  - name: split_find_people_and_involvements
    type: split
    method: token_count
    method_kwargs:
      num_tokens: 3993
    split_key: extracted_text

  - name: header_extraction_extracted_text_find_people_and_involvements
    type: map
    model: gpt-4o-mini
    output:
      schema:
        headers: "list[{header: string, level: integer}]"
    prompt: |
      Analyze the following chunk of a document and extract any headers
you see.

      { input.extracted_text_chunk }

      Examples of headers and their levels based on the document
structure:
      - "GOVERNMENT'S MOTION FOR IMMUNITY DETERMINATIONS" (level 1)
      - "Legal Framework" (level 1)
      - "Section I" (level 2)
      - "Section II" (level 2)
      - "Section III" (level 2)
      - "A. Formation of the Conspiracies" (level 3)
```

```
      - "B. The Defendant Knew that His Claims of Outcome-Determinative
Fraud Were False" (level 3)
      - "1. Arizona" (level 4)
      - "2. Georgia" (level 4)

  - name: gather_extracted_text_find_people_and_involvements
    type: gather
    content_key: extracted_text_chunk
    doc_header_key: headers
    doc_id_key: split_find_people_and_involvements_id
    order_key: split_find_people_and_involvements_chunk_num
    peripheral_chunks:
      next:
        head:
          count: 1
      previous:
        tail:
          count: 1

  - name: submap_find_people_and_involvements
    type: map
    model: gpt-4o-mini
    output:
      schema:
        people_and_involvements: list[str]
    prompt: |
      Given the document excerpt: {{ input.extracted_text_chunk_rendered
}}
      Extract all the people mentioned and summarize their involvements
in the case described. Only process the main chunk.

  - name: subreduce_find_people_and_involvements
    type: reduce
    model: gpt-4o-mini
    associative: true
    pass_through: true
    synthesize_resolve: false
    output:
      schema:
        people_and_involvements: list[str]
    reduce_key:
      - split_find_people_and_involvements_id
    prompt: |
      Given the following extracted information about individuals
involved in the case, compile a comprehensive list of people and their
specific involvements in the case:

      {% for chunk in inputs %}
      {% for involvement in chunk.people_and_involvements %}
      - {{ involvement }}
      {% endfor %}
      {% endfor %}

      Make sure to include all the people and their involvements. If a
person has multiple involvements, group them together.

pipeline:
  steps:
    - name: analyze_document
      input: legal_doc
      operations:
```

```
          - extract_metadata_find_people_and_involvements
          - split_find_people_and_involvements
          - header_extraction_extracted_text_find_people_and_involvements
          - gather_extracted_text_find_people_and_involvements
          - submap_find_people_and_involvements
          - subreduce_find_people_and_involvements

      output:
        type: file
        path: /path/to/your/output/people_and_involvements.json
        intermediate_dir: /path/to/your/intermediates
```

1. This is an example parsing function, as explained in the Parsing docs. You can define your own parsing function to extract the text you want to split, or just have the text be directly in the json file.

Running the pipeline with `docetl run pipeline.yaml` will execute the pipeline and save the output to the path specified in the output section. It cost $0.05 and took 23.8 seconds with gpt-4o-mini.

Here's a table with one column listing all the people mentioned in the case and their involvements:

Final Output

People Involved in the Case and Their Involvements

DONALD J. TRUMP: Defendant accused of orchestrating a criminal scheme to overturn the 2020 presidential election results through deceit and collaboration with private co-conspirators; charged with leading conspiracies to overturn the 2020 presidential election; made numerous claims of election fraud and pressured officials to find votes to overturn the election results; incited a crowd to march to the Capitol; communicated with various officials regarding election outcomes; exerted political pressure on Vice President Pence; publicly attacked fellow party members for not supporting his claims; involved in spreading false claims about the election, including through Twitter; pressured state legislatures to take unlawful actions regarding electors; influenced campaign decisions and narrative regarding the election results; called for action to overturn the certified results and demanded compliance from officials; worked with co-conspirators on efforts to promote fraudulent elector plans and led actions that culminated in the Capitol riot.

MICHAEL R. PENCE: Vice President at the time, pressured by Trump to obstruct Congress's certification of the election; informed Trump there was no evidence of significant fraud; encouraged Trump to accept election results; involved in discussions with Trump regarding election challenges and strategies; publicly asserted his constitutional limitations in the face of Trump's pressure; became the target of attacks from Trump and the Capitol rioters; sought to distance himself from Trump's efforts to overturn the election.

CC1: Private attorney who Trump enlisted to falsely claim victory and perpetuate fraud allegations; participated in efforts to influence political actions in targeted states; suggested the defendant declare victory despite ongoing counting; actively involved in making false fraud claims regarding the election; pressured state officials; spread false claims about election irregularities and raised threats against election workers; coordinated fraudulent elector meetings and misrepresented legal bases.

CC2: Mentioned as a private co-conspirator involved in the efforts to invalidate election results; proposed illegal strategies to influence the election certification; urged others to decertify legitimate electors; involved in discussions influencing state officials; pressured Mike Pence to act against certification; experienced disappointment with Pence's rejection of proposed strategies; presented unlawful plans to key figures.

CC3: Another private co-conspirator involved in scheming to undermine legitimate vote counts; promoted false claims during public hearings and made remarks

People Involved in the Case and Their Involvements

inciting fraud allegations; encouraged fraudulent election lawsuits and made claims about voting machines; pressured other officials regarding claims of election fraud.

CC5: Private political operative who collaborated in the conspiracy; worked on coordinating actions related to the fraudulent elector plan; engaged in text discussions regarding the electors and strategized about the fraud claims.

CC6: Private political advisor providing strategic guidance to Trump's re-election efforts; involved in communications with campaign staff regarding the electoral vote processes.

P1: Private political advisor who assisted with Trump's re-election campaign; advocated declaring victory before final counts; maintained a podcast spreading false claims about the election.

P2: Trump's Campaign Manager, providing campaign direction during the election aftermath; informed the defendant regarding false claims related to state actions.

P3: Deputy Campaign Manager, involved in assessing election outcomes; coordinated with team members discussing legal strategies post-election; marked by frequent contact with Trump regarding campaign operations.

P4: Senior Campaign Advisor, part of the team advising Trump on election outcome communication; expressed skepticism about allegations of fraud; contradicted Trump's claims about deceased voters in Georgia.

P5: Campaign operative and co-conspirator, instructed to create chaos during vote counting and incited unrest at polling places; engaged in discussions about the elector plan.

P6: Private citizen campaign advisor who provided early warnings regarding the election outcome; engaged in discussions about the validity of allegations.

P7: White House staffer and campaign volunteer who advised Trump on potential election challenges and outcomes; acted as a conduit between Trump and various officials; communicated political advice relevant to the election.

P8: Staff member of Pence, who communicated about the electoral process and advised against Trump's unlawful plans; was involved in discussions of political strategy surrounding election results.

People Involved in the Case and Their Involvements

P9: White House staffer who became a link between Trump and campaign efforts regarding fraud claims; provided truthful assessments of the situation; facilitated communications during post-election fraud discussions.

P12: Attended non-official legislative hearings; involved in spreading disinformation about election irregularities.

P15: Assistant to the President who overheard Trump's private comments about fighting to remain in power after the 2020 election; involved in discussions about various election-related strategies.

P16: Governor of Arizona; received calls from Trump regarding election fraud claims and the count in Arizona.

P18: Speaker of the Arizona State House contacted as part of efforts to challenge election outcomes; also expressed reservations about Trump's strategies.

P21: Chief of Staff who exchanged communications about the fraudulent allegations; facilitated discussions and logistics during meetings.

P22: Campaign attorney who verified that claims about deceased voters were false; participated in discussions around the integrity of the election results.

P26: Georgia Attorney General contacted regarding fraud claims; openly stated there was no substantive evidence to support fraud allegations; discussed Texas v. Pennsylvania lawsuit with Trump.

P33: Georgia Secretary of State; defended election integrity publicly; stated rumors of election fraud were false; involved in discussions about the impact of fraudulent elector claims in Georgia.

P39: RNC Chairwoman; advised against lobbying with state legislators; coordinated with Trump on fraudulent elector efforts; refused to promote inaccurate reports regarding election fraud.

P47: Philadelphia City Commissioner; stated there was no evidence of widespread fraud; targeted by Trump for criticism after his public statements.

People Involved in the Case and Their Involvements
P52: Attorney General who publicly stated that there was no evidence of fraud that would affect election results; faced pressure from Trump's narrative.
P50: CISA Director; publicly declared the election secure; faced backlash after contradicting Trump's claims about election fraud.
P53: Various Republican U.S. Senators participated in rallies organized by Trump; linked to his campaign efforts regarding the election process.
P54: Campaign staff member involved in strategizing about elector votes; discussed procedures and expectations surrounding election tasks and claims.
P57: Former U.S. Representative who opted out of the fraudulent elector plan in Pennsylvania; cited legal concerns about the actions being proposed.
P58: A staff member of Pence involved in communications directing Pence regarding official duties, managing conversations surrounding election processes.
P59: Community organizers who were engaged in discussions relating to Trump's electoral undertakings.
P60: Individual responses to Trump's directives aimed at influencing ongoing election outcomes and legislative actions.

Optional: Compiling a Pipeline into a Split-Gather Pipeline

You can also compile a pipeline into a split-gather pipeline using the `docetl build` command. Say we had a much simpler pipeline for the same document analysis task as above, with just one map operation to extract people and their involvements.

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

datasets:
  legal_doc: 1
    path: /path/to/dataset.json
    type: file
    parsing: 2
      - input_key: pdf_url
        function: azure_di_read
        output_key: extracted_text
        function_kwargs:
```

```
          use_url: true
          include_line_numbers: true

operations:
  - name: find_people_and_involvements
    type: map
    optimize: true
    prompt: |
      Given this document, extract all the people and their involvements in
the case described by the document.

      {{ input.extracted_text }}

      Return a list of people and their involvements in the case.
    output:
      schema:
        people_and_involvements: list[str]

pipeline:
  steps:
    - name: analyze_document
      input: legal_doc
      operations:
        - find_people_and_involvements

  output:
    type: file
    path: "/path/to/output/people_and_involvements.json"
```

1. This is an example parsing function, as explained in the Parsing docs. You can define your own parsing function to extract the text you want to split, or just have the text be directly in the json file. If you want the text directly in the json file, you can have your json be a list of objects with a single field "extracted_text".
2. You can remove this parsing section if you don't need to parse the document (i.e., if the text is already in the json file in the "extracted_text" field in the object).

In the pipeline above, we don't have any split or gather operations. Running `docetl build pipeline.yaml [--model=gpt-4o-mini]` will output a new pipeline_opt.yaml file with the split and gather operations highlighted--like we had defined in the previous example. Note that this cost us $20 to compile, since we tried a bunch of different plans...

Analyzing NTSB Airplane Crash Reports

This tutorial demonstrates how to analyze National Transportation Safety Board (NTSB) airplane crash reports using PDF processing capabilities of certain LLM providers. We'll build a pipeline that extracts crash causes and synthesizes common patterns across incidents.

> **LLM Requirements**
>
> PDF processing is only supported with Claude (Anthropic) or Gemini (Google) models.

Dataset Overview

The dataset contains 689 NTSB airplane crash reports--the reports corresponding to fatal accidents after 2020. You can download it here: NTSB Airplane Crashes

Pipeline Overview

Our pipeline will:

1. Process PDF crash reports from the NTSB database to extract causes and recommendations
2. Synthesize common patterns across all analyzed crashes

Let's examine the pipeline structure:

```yaml
pipeline:
  steps:
    - name: analyze_crashes
      input: crashes
      operations:
        - extract_crash_cause # this is a map operation
        - synthesize_findings # this is a reduce operation
```

Full Pipeline Configuration

```yaml
datasets:
  crashes:
    type: file
    path: "fatal.json"

default_model: gemini/gemini-2.0-flash

operations:
  - name: extract_crash_cause
    type: map
    pdf_url_key: ReportUrl
    skip_on_error: true # Skip llm calls where the PDF is malformed or not
found
    output:
      schema:
        cause: str
        contributing_factors: "list[str]"
        recommendations: str
    prompt: |
      Analyze this NTSB airplane crash report and extract:
      1. The primary cause of the crash (2-3 sentences)
      2. Any contributing factors (list)
      3. Key safety recommendations made

  - name: synthesize_findings
    type: reduce
    reduce_key: _all
    output:
      schema:
        summary: str
    prompt: |
      Analyze the following airplane crash reports:

      {% for item in inputs %}
      Report {{loop.index}}:
      Cause: {{ item.cause }}
      Contributing Factors: {{ item.contributing_factors | join(", ") }}
      Recommendations: {{ item.recommendations }}

      {% endfor %}

      Generate a comprehensive analysis that:
      1. Identifies common causes across incidents
      2. Lists recurring contributing factors
      3. Synthesizes key safety recommendations
      4. Highlights any notable patterns

      Format your response as a structured report.

pipeline:
  steps:
    - name: analyze_crashes
      input: crashes
      operations:
        - extract_crash_cause
        - synthesize_findings
```

```
  output:
    type: file
    path: "crash_analysis.json"
    intermediate_dir: "checkpoints"
```

Sample Output

Here's the output we get from running the pipeline:

Sample Analysis Output

Airplane Crash Report Analysis:

1. Common Causes:

After analyzing the provided airplane crash reports, the most common primary causes include:

- Loss of Control (often due to aerodynamic stall, spatial disorientation, or pilot incapacitation)
- Engine Failure (often due to fuel exhaustion, mechanical issues, or improper maintenance)
- Controlled Flight Into Terrain (CFIT) (often in IMC or low visibility)
- Pilot Error (poor decision-making, failure to maintain airspeed, inadequate pre-flight planning).

2. Recurring Contributing Factors:

Several contributing factors recur across multiple reports:

- Improper Maintenance (inadequate inspections, incorrect repairs)
- Fuel Issues (fuel exhaustion, fuel contamination, improper fuel management)
- Adverse Weather Conditions (IMC, icing, turbulence, low visibility)
- Pilot Impairment (fatigue, alcohol/drug use, medical conditions)
- Failure to Maintain Airspeed (leading to stalls)
- Low Altitude Maneuvering
- Lack of Instrument Proficiency
- Poor Decision-Making (continuing flight into adverse conditions, improper risk assessment)
- Spatial Disorientation (particularly in IMC or at night)
- Inadequate Pre-flight Planning (weather, fuel, weight and balance).
- Exceeding Aircraft Limitations (weight, structural, etc.)

3. Synthesized Key Safety Recommendations:

Based on the analyzed reports, key safety recommendations can be synthesized:

- **Enhanced Pilot Training:**
  - Stall recognition and recovery techniques
  - Instrument meteorological conditions (IMC) flight procedures and spatial disorientation awareness.
  - Mountain flying techniques and high-density altitude operations

    - Emergency procedures training, particularly related to engine failures.
    - Aerobatic Maneuver training
- **Improved Maintenance Practices:**
    - Adherence to manufacturer's recommended maintenance schedules and procedures
    - Thorough inspections of critical components (fuel systems, control cables, engines)
    - Proper documentation of maintenance and repairs
    - Emphasis on proper installation and torquing of critical parts.
- **Robust Pre-flight Planning:**
    - Thorough weather briefings and in-flight weather monitoring
    - Accurate fuel planning and management
    - Weight and balance calculations
    - Familiarization with terrain, obstacles, and airport characteristics.
- **Sound Aeronautical Decision-Making:**
    - Avoid flying under the influence of alcohol or drugs
    - Avoid self-induced pressure to complete a flight
    - Recognize personal limitations and make conservative go/no-go decisions
    - Proper risk management
- **Effective Use of Technology:**
    - Installation and proper use of angle-of-attack indicators
    - Use of autopilot systems and electronic flight displays
    - Ensure aircraft has and is broadcasting ADS-B signals, and use traffic advisory systems when available
- **Awareness of Physiological Factors:**
    - Understanding of spatial disorientation and how to mitigate its effects
    - Awareness of the effects of fatigue, medical conditions, and medications on pilot performance.
    - Use of oxygen at night.
- **Adherence to Regulations and Procedures:**
    - Compliance with minimum safe altitudes and approach procedures
    - Proper use of checklists
    - Following air traffic control instructions.
- Maintain proper aircraft certification.

4. Notable Patterns:

- **VFR into IMC:** A significant number of accidents involve pilots without instrument ratings continuing visual flight into instrument meteorological conditions.
- **Loss of Control on Approach:** A recurring theme is loss of control during the approach phase, often related to stalls, wind shear, or unstable approaches.
- **Pilot Actions Under Stress:** Many accidents involve pilots making poor decisions under stressful situations, such as engine failures or adverse weather conditions.
- **Experimental Aircraft Issues:** Several reports involve experimental amateur-built aircraft, highlighting potential risks associated with construction, maintenance, and pilot familiarity.
- **Medical Incapacitation:** Several accidents were potentially caused by medical incapacitation of the pilot, suggesting that it may be necessary to have health safety standards, especially for older pilots.
- **Power Lines:** A concerning number of incidents involve collision with power lines during aerial application or low-altitude maneuvering.

The pipeline costs < $0.05 USD to run.