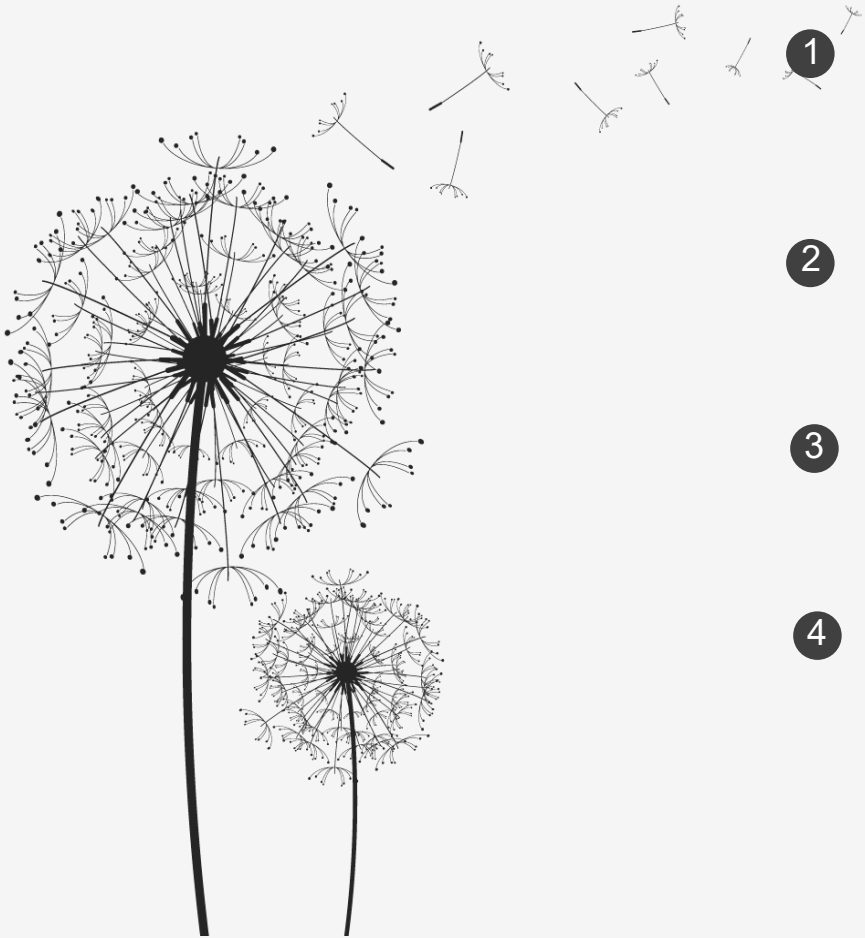
LLM-Assisted Qualitative Coding: A Hands-on Training Workshop

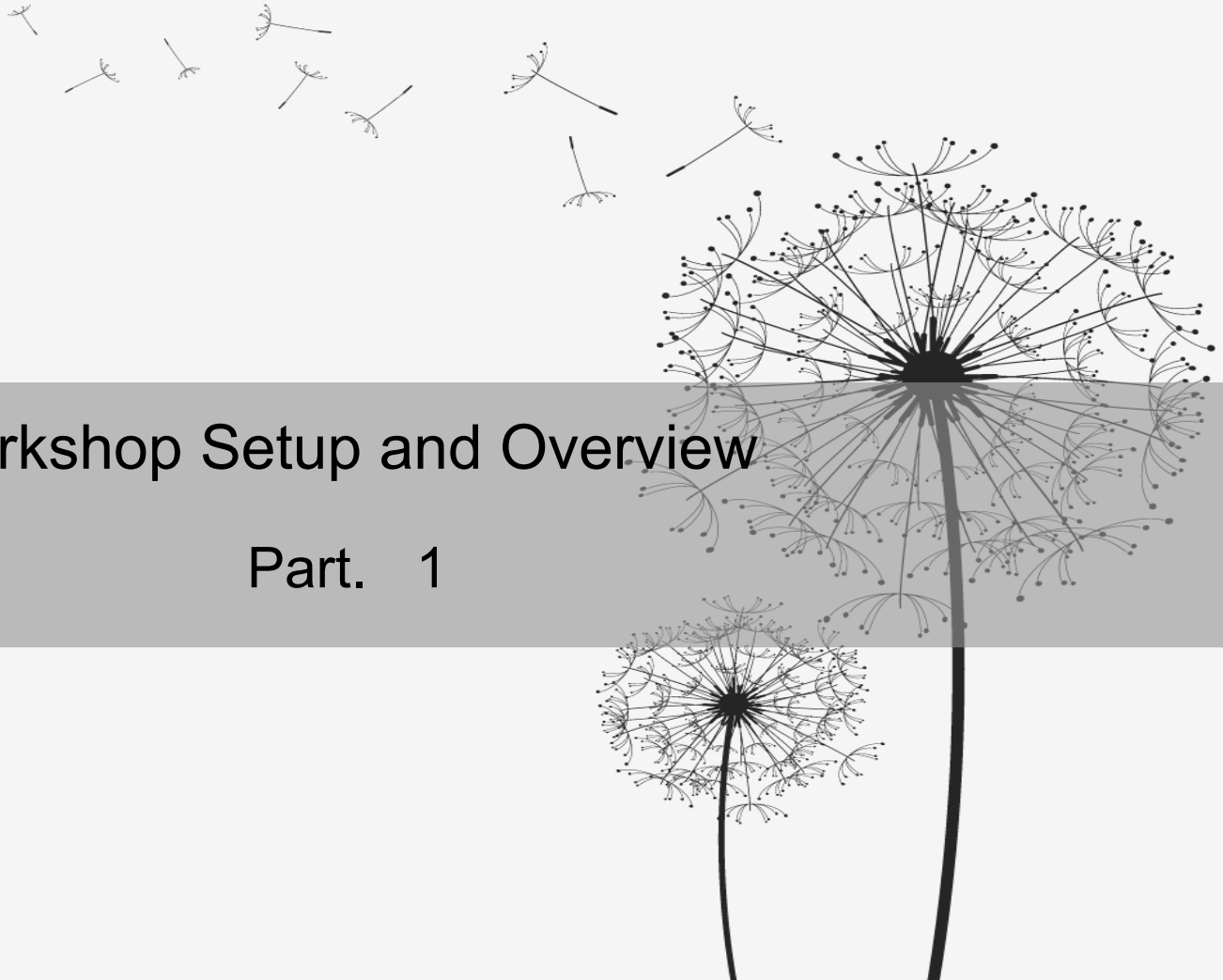
Alex Liu



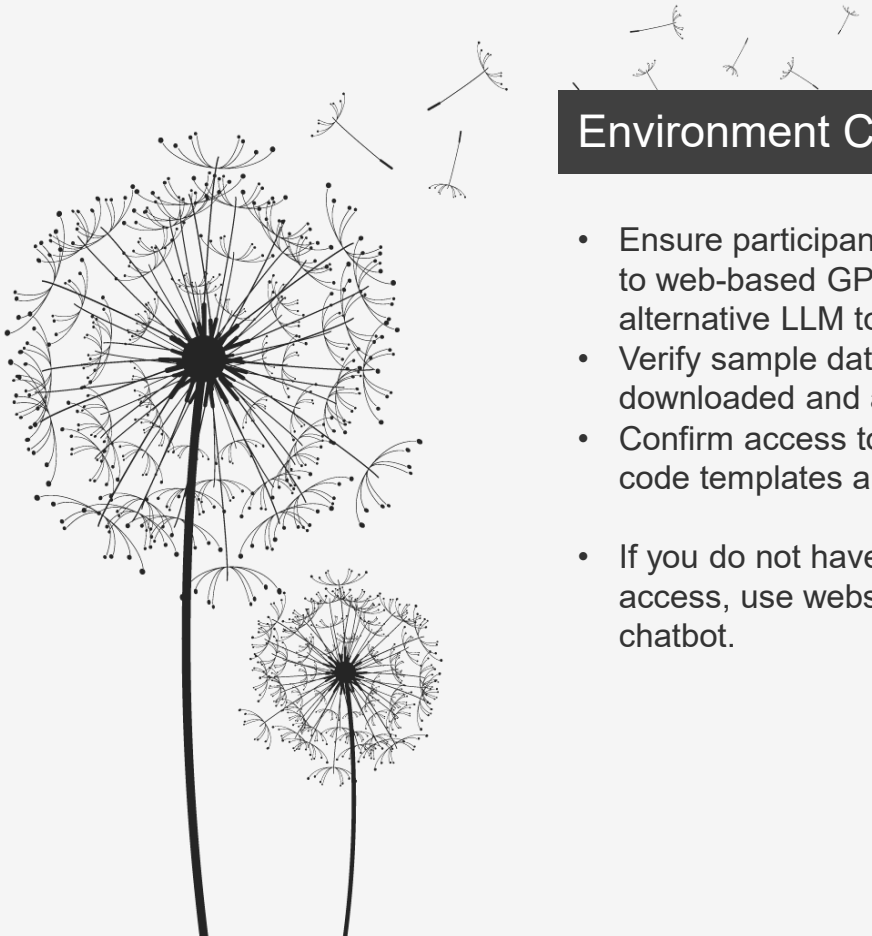
1. Workshop Setup and Overview
2. Exploratory Analysis Demo
3. Hands-on Demo: Codebook-Based Analysis
4. Practical Exercise

Workshop Setup and Overview

Part. 1



Getting Started



Environment Check

- Ensure participants have access to web-based GPT interface or alternative LLM tools
- Verify sample datasets are downloaded and accessible
- Confirm access to provided code templates and notebooks

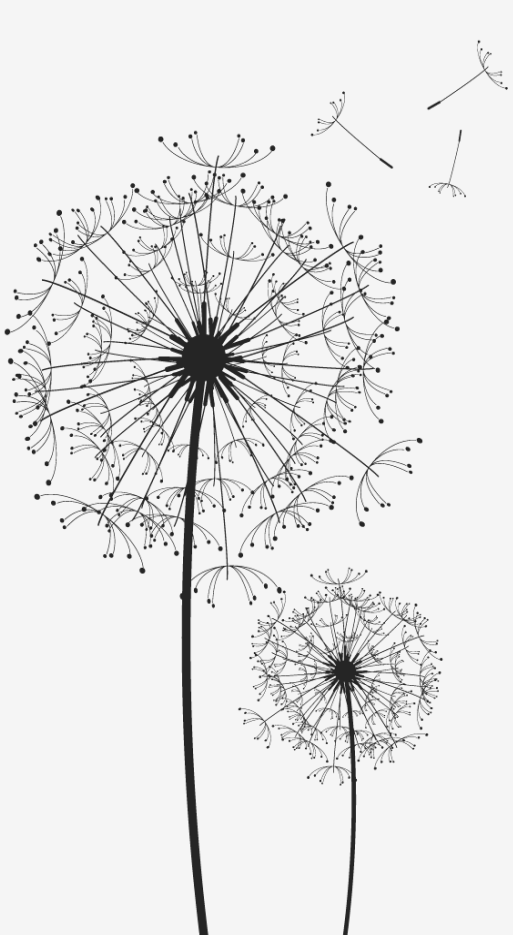
- If you do not have LLM API access, use website-based chatbot.

Workshop Goals

- Practice implementing LLM-assisted coding workflows
- Gain hands-on experience with both codebook and exploratory approaches
- Learn troubleshooting techniques for common challenges

- For each approach, I will describe the workflow first. Then I will show the code in Google Colab using LLM API. Last, I will show the equivalent approach using website-based chatbot.

Choosing Your Coding Approach



Open Coding Situations

examine data (like interviews, observations, or texts) without preconceived categories, allowing themes and patterns to emerge naturally from the data itself.

- Exploring new or understudied phenomena where existing frameworks may not apply
- Generating theory from ground-up through inductive analysis
- Initial phase of mixed-methods research to identify emerging themes
- Working with diverse or unexpected data patterns

Codebook-Based Situations

examine data (like interviews, observations, or texts) with preconceived categories

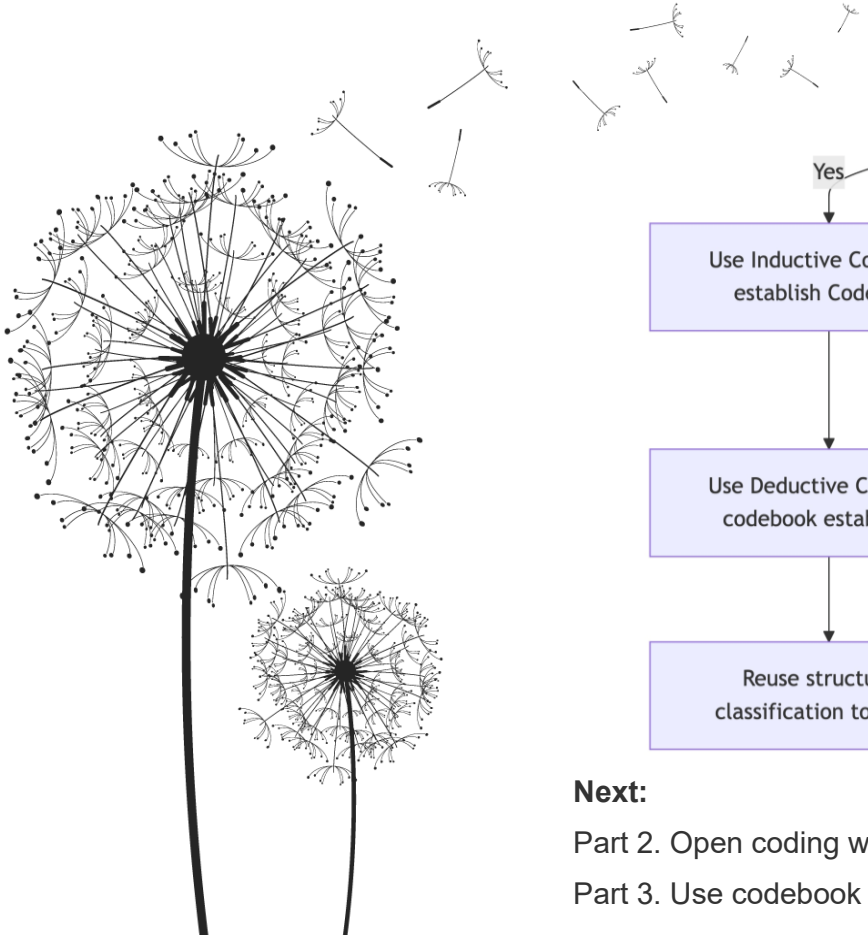
- Testing existing theoretical frameworks or hypotheses
- Replication studies requiring consistent coding schemes
- Large-scale projects with multiple coders needing standardization
- Comparative analysis across multiple datasets or time periods

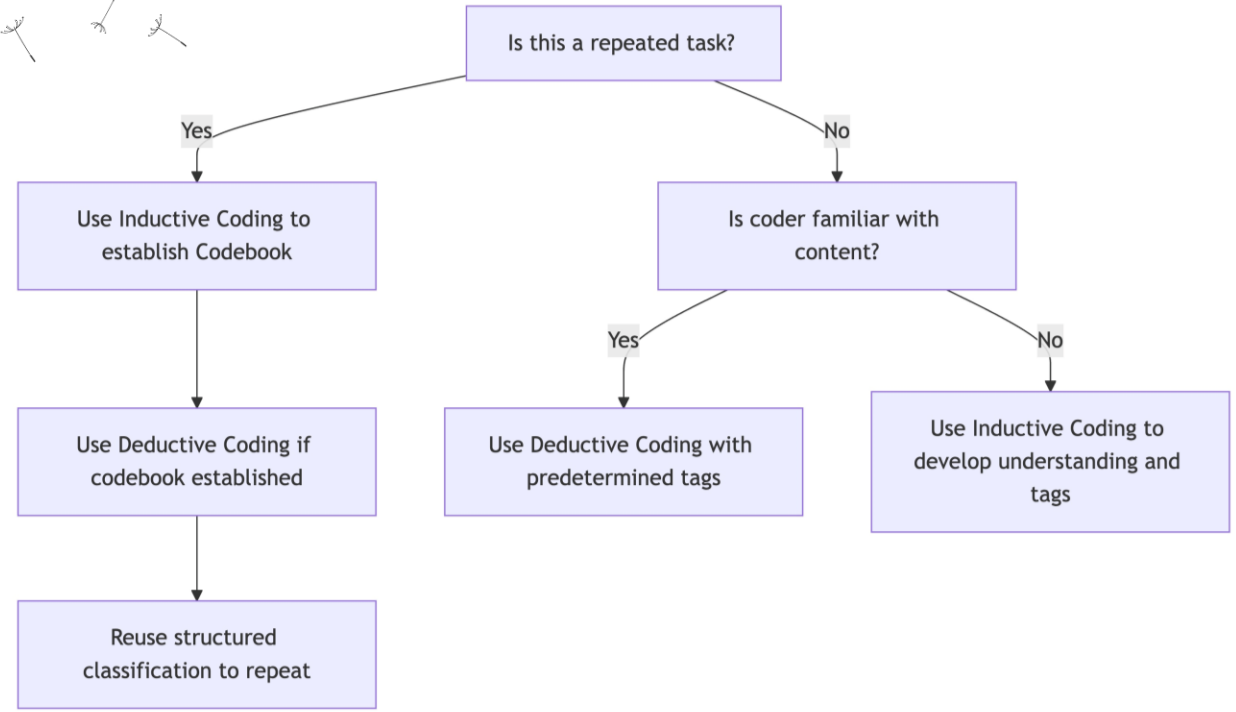
Next:

Part 2. Open coding with LLM to establish codebook

Part 3. Use codebook for codebook-based coding

Choosing Your Coding Approach





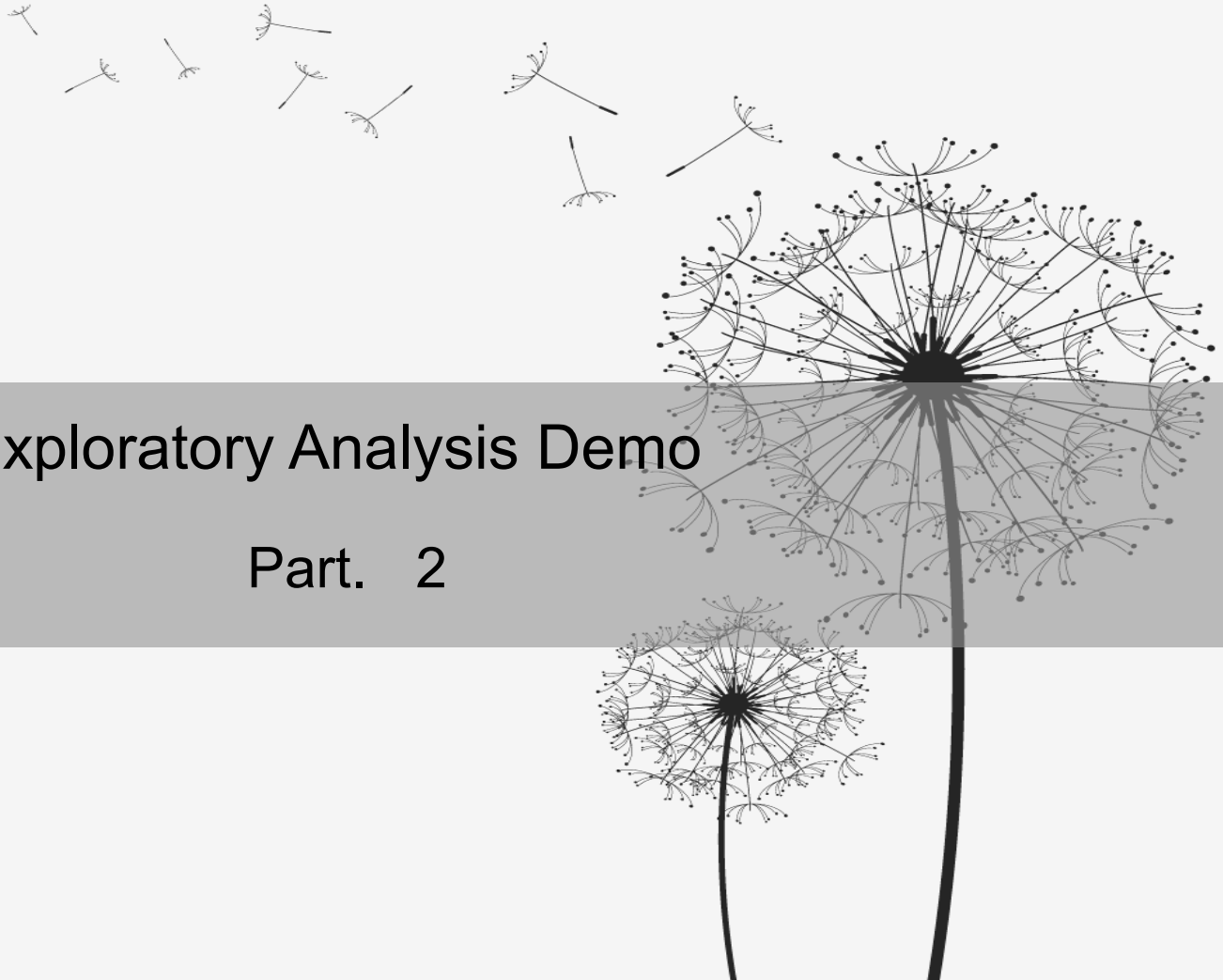
Next:

Part 2. Open coding with LLM to establish codebook

Part 3. Use codebook for codebook-based coding

Exploratory Analysis Demo

Part. 2



Open Coding with LLM Support

Topic discovery

- Clean up and preprocess text data
- Configure LLM for unsupervised analysis
- Alternative: STM or LDA topic modeling approaches
- Code in your folder:
- ISEA_LDA.ipynb
- ISEA_STM_topic_model.Rmd
- Generate initial topic/theme clusters

Two LLM Conceptualization Approaches

1. Provide context and one row/document at a time. Ask LLM to provide contextual labels and descriptions for each row.
2. Provide combined text to LLM. Ask LLM to generate themes and descriptions based on the entire text. To verify, you need to select a sample to label based using the identified themes.

Pattern Recognition

- Identify recurring concepts and relationships
- Map connections between themes
- If needed, create visual representation of findings

Validation Techniques

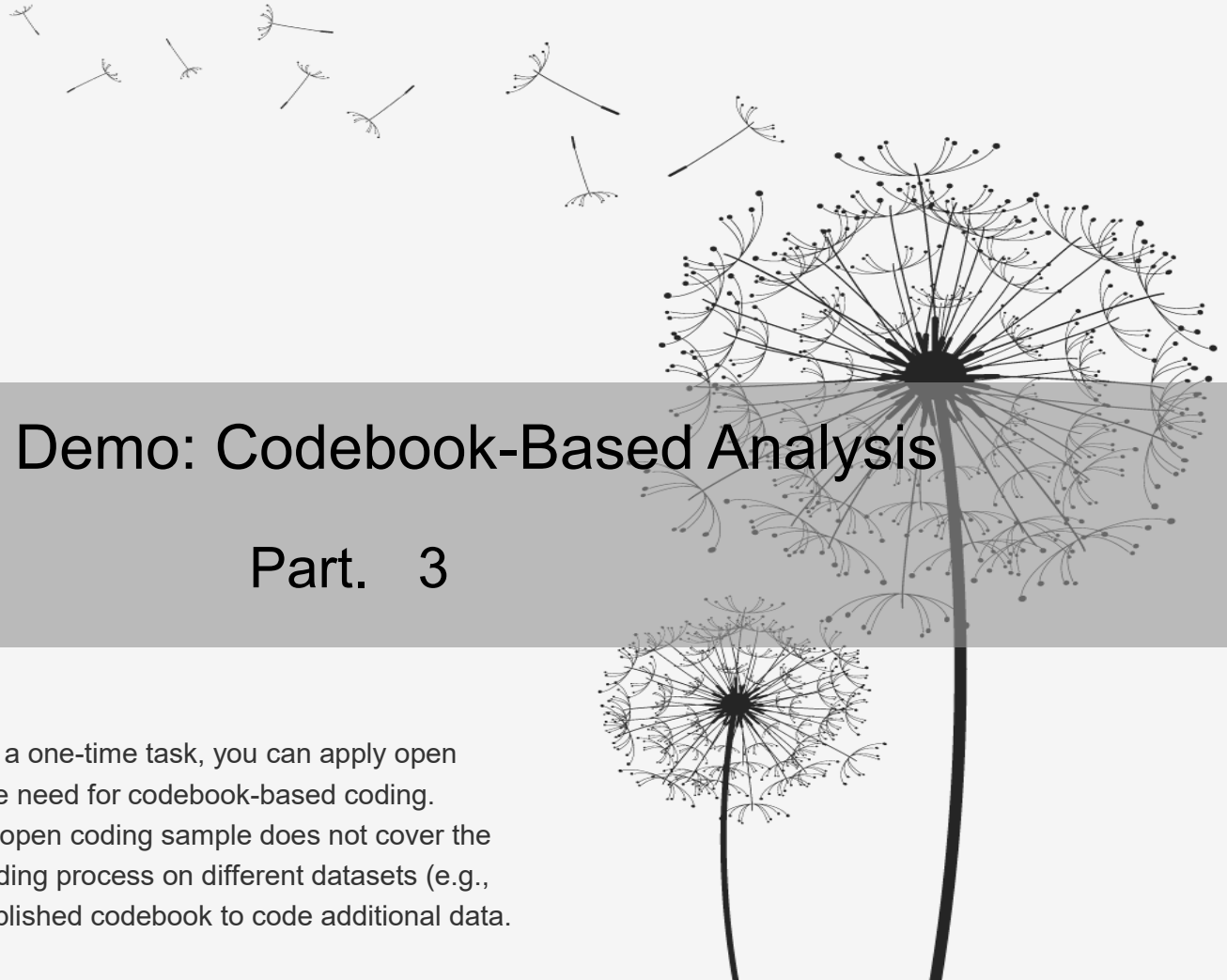
- Cross-reference with manual coding samples
- Apply intercoder reliability checks
- Document LLM decision patterns

DEMO - Open Coding with LLM Support

In ISEA_topic_code_using_openai.ipynb, you need section load data and LLM API Integration/open coding for **open coding** on a DataFrame using an **LLM API**.

The script follows these steps:

1. **Load Data**: Reads a CSV file containing qualitative data.
- **Preprocess Data**: Cleans and structures text for input into the LLM.
1. **LLM API Integration**: Set up functions and parameters to Configure LLM. Sends each text entry to an LLM for open coding.
2. **Extract Codes**: Parses and structures the generated codes.
3. **Create a Codebook**: Aggregates extracted codes into a structured format.
4. **Render Final Codebook**: Exports the final codebook as a structured DataFrame.

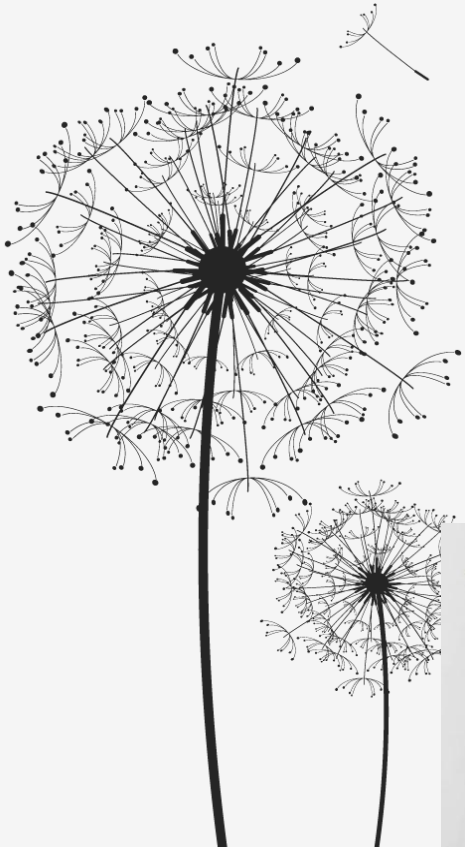


Hands-on Demo: Codebook-Based Analysis

Part. 3

If your dataset is small and open coding is a one-time task, you can apply open coding to the entire dataset, eliminating the need for codebook-based coding. However, if your dataset is large and your open coding sample does not cover the entire data, or if you need to repeat the coding process on different datasets (e.g., annual reviews), you need to use the established codebook to code additional data.

Working with Structured Codes



Setting Up the Environment

- Load sample interview transcripts into working environment
- Import pre-defined codebook
- Configure LLM parameters for optimal analysis

Running the Analysis

- Execute initial coding pass using LLM
- test on samples
- Review auto-coded segments
- you can request confidence scores
- If you are repeatedly coding similar but not identical data, provide an "Other" label option
- Implement human validation workflow
- Invite people who know the context of the work but not previously participated in the coding process
- and/or, manually label a random sample using the codebook and compare with AI labels

Quality Control

- Check for coding consistency
- Address edge cases and ambiguous segments
- *Document decision rules for future reference*

DEMO - Codebook-based Coding with LLM Support

In ISEA_topic_code_using_openai.ipynb, you need section load data, load codebook and LLM API Integration/codebook-based coding for **codebook-based coding** on a DataFrame using an **LLM API**.

The script follows these steps:

1. **Load Data**: Reads a CSV file containing qualitative data.
- **Preprocess Data**: Cleans and structures text for input into the LLM.
1. **LLM API Integration**:
2. Set up functions and parameters to Configure LLM.
3. Sends each text entry to an LLM for open coding.
4. Prompt design incorporating established codebook
5. **Extract Codes**: Parses and structures the coding results.

DEMO - Example of Zero-shot Prompt with confident scores and “Other”

Step 2: Code the Paragraph using the Codebook:

a. Read the Paragraph carefully, considering its context within the Washington State K-12 public school system.

b. Identify the **top three most salient themes** from the 'Child' categories, using 'Child_description' and 'Key words' as references.

c. If **no appropriate Child category fits**, use the Parent category instead.

d. If **neither Parent nor Child categories fit**, label the Paragraph as **"Other"** with brief specification.

e. Assign a **confidence score (0 to 1)** to each assigned theme, representing how certain you are about the classification.

f. Provide a reasoning statement justifying your theme selections.

Step 3: Review and ensure that the assigned themes accurately represent the Paragraph.

- If a Paragraph does not fit any Codebook category, return `"Other: specification` as the theme.

unchanged

Expected JSON Output Format:

{{ ***"Theme 1": "Selected Theme or 'Other'",***

"Theme 2": "Selected Theme or 'Other'",

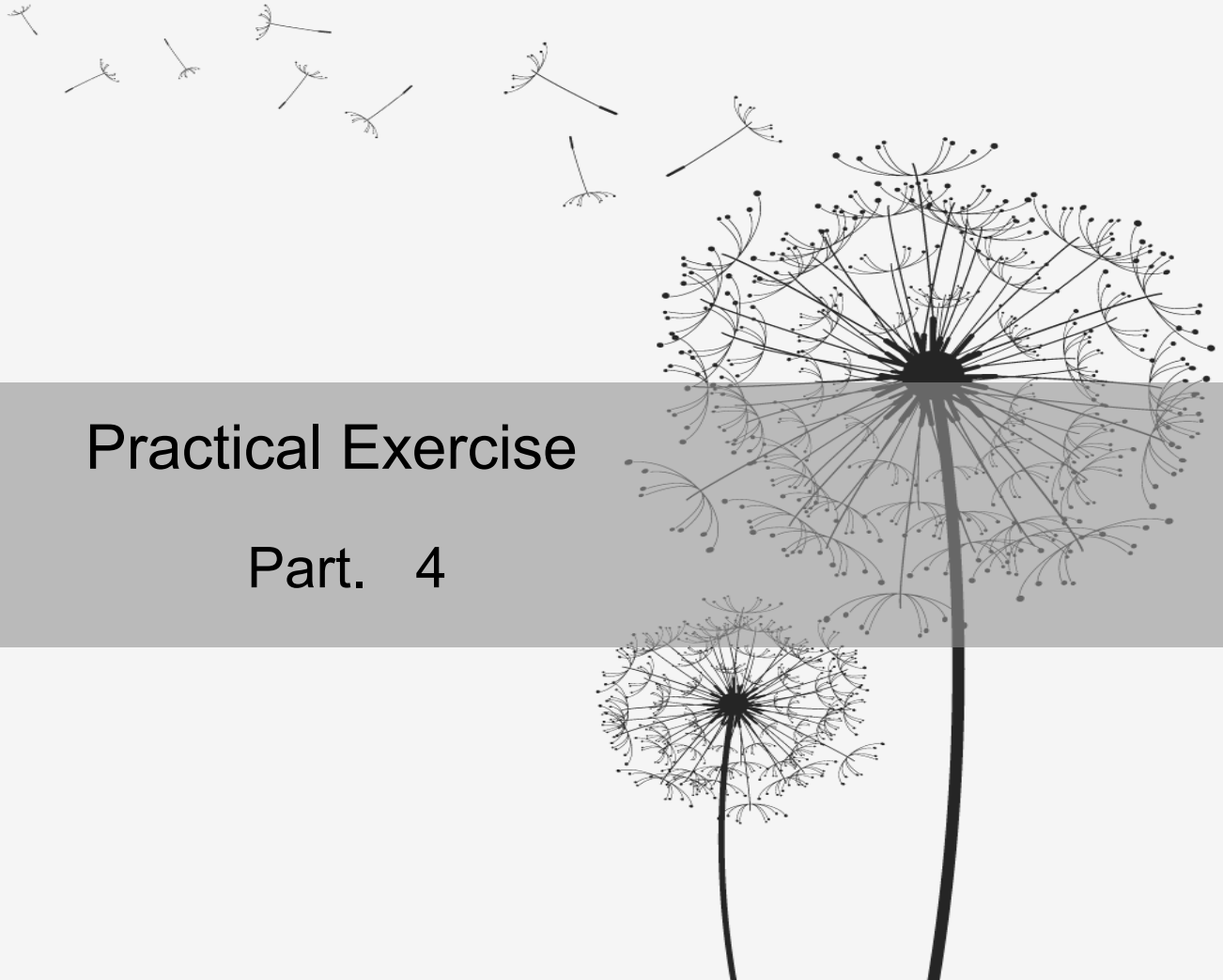
"Theme 3": "Selected Theme or 'Other'",

"Confidence Scores": [0.92, 0.85, 0.65],

"Reasoning": "Brief explanation of why these themes were selected."}}

Practical Exercise

Part. 4



Hands-on Application

- Apply learned techniques to sample_interview_data.csv.
- First, implement **exploratory approaches (open coding)** to establish a codebook. Then, apply the codebook for a more structured coding process.
- Your codebook does not need to follow the structure of the provided codebook. For example, it does not require a two-layer format. Instead, it should reflect the characteristics of the data sample and include **Codes** and **Descriptions**.
- Conduct sentiment analysis on the dataset. You can define any sentiment you want to explore.
- Discovery the connection between themes and sentiments.
- Experiment with different **LLM settings**, including various models, parameters (if applicable), and prompts. Document your **coding decisions, rationale, and validation process** throughout the analysis.
- Submit a **write-up** of your coding results and a **description of your codebook** (single-spaced, no longer than two pages).

Best Practices

- Maintain consistent validation protocols
- Document LLM configuration settings
- Create reproducible workflows