USDA LLM Model – Comment Clustering & Categorization Walkthrough

OVERVIEW:

This script takes the output from the first LLM model (`processed_comments.csv`) and:

1. Groups similar "issues" into broader categories using GPT.

2. Logs the raw GPT response and category mappings for transparency.

3. Adds new "high_level_issues" to the dataset.

4. Outputs:

   • A categorized CSV (`processed_with_categories.csv`)

   • A sorted version by issue category (`sorted_by_issue.csv`)\

1. SETUP STEPS

STEP 1: Install Required Python Packages

Run this in your terminal:

```
pip install openai pandas python-dotenv
```

STEP 2: Confirm Environment Setup

Make sure you already created a `.env` file in the same directory as your script (if not, do it now):

.env

OPENAI_API_KEY=sk-...your_openai_key_here...

STEP 3: Update File Paths

Edit the following values at the top of `comment_clustering.py`:

```
INPUT_FILE = r"C:\Path\To\processed_comments.csv"
CATEGORIZED_OUTPUT = "processed_with_categories.csv"
SORTED_OUTPUT = "sorted_by_issue.csv"
GPT_RAW_RESPONSE_LOG = "gpt_issue_grouping_raw.txt"
CATEGORY_MAPPING_LOG = "gpt_category_consolidation.txt"
```

These control:

• Where your source file comes from (output of first script)

• Where categorized and sorted CSVs go

• Where raw GPT responses and category cleanup mappings are saved

2. SCRIPT LOGIC (EXPLAINED)

Load Comments & Extract Unique Issues

- Reads the `processed_comments.csv` file
- Cleans and splits the "issues" field into lists
- Counts all unique issues

Group Issues into Broad Categories (GPT)

Function: `build_prompt(issues)`

• Builds a clean, instructional prompt asking GPT to group similar issues together

• Limits total categories to ~8–15 to avoid over-fragmentation

• Expects structured JSON back:

```
[
  {
    "category": "Public Health Risks",
    "related_issues": ["PCB contamination", "Fish consumption advisories"]
  },
  ...
]
```

Function: `extract_json_block(text)`

• Extracts and parses the GPT response as valid JSON

• If parsing fails, it writes the raw response to `gpt_issue_grouping_failed_batch.txt`

Batching:

• Issues are processed in batches of 500 to prevent token overload

• Each GPT response is saved to `gpt_issue_grouping_raw.txt`

Consolidate Similar Categories (Optional Cleanup Step)

Function: `consolidate_categories(categories)`

• Sends all high-level category names to GPT

• GPT returns a dictionary mapping similar categories to unified names:

```
 {
   "Worker Health and Safety": "Worker Safety",
   "Worker Protection": "Worker Safety"
 }
```

• Saves this mapping to `gpt_category_consolidation.txt`

Apply Category Mapping to Each Issue

Function: `map_to_categories(issues)`

• Maps each issue to its high-level category (using `issue_to_category` dictionary)

• Adds a new field to each row called `high_level_issues`

Output Categorized CSVs

----

1. `processed_with_categories.csv` — Main output with added `high_level_issues` column
2. `sorted_by_issue.csv` — Exploded so each comment appears once per category (for aggregation/grouping)

CUSTOMIZATION TIPS

→ Change GPT model:

model="gpt-4o"

(You may switch to "gpt-4-turbo" or other newer models based on cost/speed needs)

→ Change number of categories:

Edit the instructions in `build_prompt()` to set your desired range (e.g., 8–15).

→ Prompt refinement:

You can add more context to the prompt to change how GPT consolidates categories, such as:

- “Try to reflect USDA policy areas.”
- “Combine based on common regulatory impact.”

HOW TO RUN

Once your `.env` file and paths are set:

1. Run the script in the terminal or IDE:

```
python comment_clustering.py
```

2. Check for output:
   - processed_with_categories.csv
   - sorted_by_issue.csv
   - gpt_issue_grouping_raw.txt
   - gpt_category_consolidation.txt

3. Review any batches that failed to parse (check logs for formatting issues).

4. ANALYST TIPS

- Use `with open(..., "a")` logging to trace how GPT interpreted each batch.
- Watch for formatting inconsistencies in the GPT response — they can break parsing.
- If a batch fails:
   • Check the failed response in `gpt_issue_grouping_failed_batch.txt`

  • Retry manually or refine the batch

- Consolidated categories may still require manual touch-ups if GPT returns subtle variants.