Chat Log Classification Using LLM

Anna Gornyitzki
Undergraduate Research Assistant, Computer Science
UC Santa Barbara
Mentored by Dr. Nir Chemaya (Ph.D., Economics, UCSB)

July 2025

1 Background and Objective

Since January 2025, I have been working as an undergraduate research assistant under Dr. Nir Chemaya. My role focuses on engineering a workflow to classify chat logs from multiplayer strategic trading games conducted as part of a socioeconomic experiment. The objective is to evaluate whether GPT-4o can reliably categorize textual summaries of chat logs based on content relevance and game behavior. The central part of this project involves designing and refining prompts, as well as building a data processing workflow to evaluate the model's outputs for consistency and reproducibility. This ensures the classified data can be confidently used in later statistical analysis of participant behavior.

2 Data Processing and Summarization Process

I was given 87 raw CSV files containing chat logs from Dr. Chemaya's experimental game trials. Each file was tagged with a unique session ID (e.g., `1bx2zszj`) and labeled by structure type, such as `3-player 1`, `3-player 2`, or `6-player`. As a human annotator I manually reviewed each file and labeled every line of dialogue as **relevant** (contributing to strategy or gameplay), **irrelevant** (e.g., small talk), or **unsure**. After annotating each file, I wrote a brief summary describing the strategic conversation. This process was repeated across all 87 files to produce clean, structured versions suitable for further analysis.

After completing the summaries, I received additional context about the experiment, including treatment numbers and gameplay mechanics. A second research assistant had independently summarized the same chats, and I merged both sets into a single `master.csv` file containing session IDs, treatments, and RA identifiers. This dataset served as the basis for evaluating classification consistency.

The `master.csv` file contains 174 rows for each summary, along with the following columns:

- **id_chat**: A unique numeric ID (1–174) for easy reference.
- **session**: An alphanumeric session tag (e.g., `1bx2zszj`).
- **type**: The session structure (e.g., `3-player 1` or `6-player`).
- **treatment_number**: A numeric code for the assigned treatment (its meaning was not disclosed to me).
- **ra**: Indicates which research assistant wrote the summary (`1` = me, `2` = second RA).
- **text**: The summary of the chat session, manually written based on the annotated dialogue.

id_chat	session	type	treatment_number	ra	text
1	1bx2zszj	3-player 1	2	1	Participants agreed to say yes...
2	1qlb120r	3-player 1	3	1	Participants agreed to always ...
3	33s2ujm3	3-player 1	2	1	Participants agreed to trade a...
4	3qanq77h	3-player 1	3	1	Participants agreed that every...
5	54w4p43u	3-player 1	0	1	Participants at first were unsu...
			...		
170	tl0gvaye	6-player	0	2	The players agree that always ...
171	uoqs3jzq	6-player	3	2	The players initially agree to...
172	vkmyo6a1	6-player	2	2	The players debate the best st...
173	ycqfkz1o	6-player	2	2	The group discusses their stra...
174	z5qyo0u8	6-player	0	2	The group discusses sticking w...

Table 1: Preview of the `master.csv` dataset showing the first and last few summaries.

3 Workflow for Classifying Summaries with GPT-4o

Dr. Chemaya and I collaborated to design a GPT-4o prompt to categorize textual summaries from game chat logs, aiming to test whether a large language model could produce consistent outputs across repeated runs. GPT-4o was the most advanced OpenAI language model available at the time. A **simulation** refers to one execution of the prompt in GPT-4o, returning a structured CSV with binary category labels. By running multiple simulations with identical inputs and prompts, we evaluated the model's consistency in producing stable classifications.

Model 4 Prompt and Input

Along with the prompt, we provided the LLM with a simplified input CSV file named `id_chat_input.csv`, which included only two columns: `id_chat` and `text`, extracted from `master.csv`.

Our first version of the prompt referred to as **Model 4** was as follows:

Model 4 Prompt

Use the CSV file with summaries. I want you to classify each summary based on the following categories. The classification is not required to be mutually exclusive—each summary can be classified into multiple categories if needed.
Categories: Coordination, Efficiency, Conflict, Inequality, Unknown (if it does not fit elsewhere). The output should include binary category columns.

A preview of the input CSV file, `id_chat_input.csv`, is shown below:

id_chat	text
1	Participants agreed to say yes in each trial and discussed how many times they were Red.
2	Participants agreed to always vote yes to trades and that it's advantageous for the decision...
3	Participants agreed to trade and to vote yes, so that at some point everyone will get the...
4	Participants agreed that everyone should trade and vote yes because this will give...
5	Participants at first were unsure if they would trade, but later agreed to trade and to vote yes.
	...
170	The players agree that always trading and choosing "yes" maximizes earnings for everyone...
171	The players initially agree to always trade and vote "yes" to maximize earnings, but tensions...
172	The players debate the best strategy for maximizing earnings, initially considering voting ...
173	The group discusses their strategy, agreeing that always trading and voting "yes" benefits...
174	The group discusses sticking with their previous strategy of always voting "yes" and awarding...

Table 2: Preview of the `id_chat_input.csv` dataset showing the first and last few summaries.

The first 20 simulations were run manually using ChatGPT-4o via the OpenAI web interface. The remaining 80 were automated using Python and the OpenAI API, with the `temperature` parameter set to `0.7`. While temperature can technically be set to any positive value, it is commonly used within the range of 0 to 1. Lower values (e.g., `0`) make the model's responses more deterministic and repetitive, while higher values increase randomness and creativity. A setting of `0.7` provides a balanced trade-off, enabling the model to detect subtle contextual differences between summaries while still producing consistent, reproducible outputs. Here is a code snippet:

```
...
input_df = pd.read_csv("id_chat_input.csv")
csv_input_string = input_df.to_csv(index=False)

user_prompt = """
*Model4 Prompt*
"""

# Output directory where GPT responses will be saved
output_dir = "/Users/*your path*/chatgpt_outputsM4"
os.makedirs(output_dir, exist_ok=True)

# Run 80 simulations (e.g., simulation 21 to 100)
for i in range(21, 101):
    try:
        print(f"Running simulation {i}...")

        # Submit request to GPT-4o via API
        response = client.chat.completions.create(
            model="gpt-4o",
            messages=[
                {"role": "system", "content": "You are a helpful assistant."},
                {"role": "user", "content": user_prompt},
                {"role": "user", "content": f"CSV input:\n{csv_input_string}"}
            ],
            temperature=0.7
        )

        # Save each simulation output as its own CSV file
        filename = os.path.join(output_dir, f"model4_{i}.csv")
        with open(filename, "w", encoding="utf-8") as f:
            f.write(response.choices[0].message.content)

        print(f"Saved: {filename}")
        sleep(1) # Wait 1 second between calls

    except Exception as e:
        print(f"Error during simulation {i}: {e}")
```

Listing 1: Automated GPT-4o simulations with OpenAI API

Model 4 Results

All 100 simulation csv outputs from Model 4 were then concatenated into a single CSV file. The Python script below reads and checks each file, standardizes the format, adds metadata columns (`Model` and `Simulation`), and merges them into `sum_model_4.csv` to make it easier to compare results across runs.

```
...
input_folder = 'chatgpt_outputsM4'
output_file = 'sum_model_4.csv'
dfs = []

csv_files = sorted([f for f in os.listdir(input_folder) if f.endswith('.csv')])

for file in csv_files:
    match = re.search(r'model4_(\d+)\.csv', file)
    if match:
        sim_num = match.group(1)
        df = pd.read_csv(os.path.join(input_folder, file))

        df.columns = df.columns.str.title()
        df = df.rename(columns={
            'Id_Chat': 'id_chat',
            'Text': 'text'
        })

        df['Model'] = 4
        df['Simulation'] = sim_num
        df = df[['id_chat', 'text', 'Coordination', 'Efficiency', 'Conflict', 'Inequality', 'Unknown', 'Model',
              'Simulation']]
        dfs.append(df)

pd.concat(dfs, ignore_index=True).to_csv(output_file, index=False)
```

Listing 2: Python concatenating Model 4 simulation outputs

id_chat	text	Coord.	Effici.	Conflict	Inequal.	Unknown	Model	Simulation
1	Participants agreed...	1	0	0	0	0	4	1
2	Participants agreed...	1	1	0	0	0	4	1
3	Participants agreed...	1	1	0	0	0	4	1
4	Participants agreed...	1	1	0	0	0	4	1
5	Participants at first...	1	1	0	0	0	4	1
				...				
170	The players agree...	1	0	0	0	0	4	100
171	The players initially...	1	1	0	0	0	4	100
172	The players debate...	1	1	0	0	0	4	100
173	The group discusses...	1	0	0	0	0	4	100
174	The group discusses...	0	1	0	0	0	4	100

Table 3: Preview of the `sum_model_4.csv` dataset showing the first and last few summaries.

Upon visually inspecting the data in `sum_model_4.csv`, we noticed that the LLM frequently applied multiple overlapping labels, particularly favoring combinations like **Coordination** and **Efficiency**, while rarely using categories such as **Conflict** or **Inequality**. This uneven distribution and lack of exclusivity made the results difficult to interpret and reproduce reliably across simulations.

Model 5 Prompt and Input

To address these issues, we introduced Model 5, which modified the prompt to require a single category assignment per summary. This adjustment enforced mutual exclusivity and allowed for a clearer, more consistent evaluation of model behavior across repeated trials.

The prompt used for **Model 5** was as follows:

Model 5 Prompt

Use the CSV file with summaries. I want you to classify each summary based on the following categories. The classification must be mutually exclusive—each summary can be classified into only one category.
Categories: Coordination, Efficiency, Conflict, Inequality, Unknown (if it does not fit elsewhere). The output should include binary category columns.

The prompt used the same input CSV file, `id_chat_input.csv`.

First, 20 manual simulations were run manually using ChatGPT-4o via the OpenAI web interface, then we paused to evaluate the results before deciding whether to proceed with more automated simulations.

Model 5 Results

A modified code from Listing 2 was adapted for Model 5 to generate a concatenated file, `sum_model_5.csv`, containing all simulation outputs. This file was then imported into MySQL, where a query was used to aggregate category counts per summary. The result, `totals_model5.csv`, provided a combined view of all classifications, making consistency calculations easier and more organized.

```
SELECT
    id_chat,
    text,
    SUM(Coordination) AS Coordination,
    SUM(Efficiency) AS Efficiency,
    SUM(Conflict) AS Conflict,
    SUM(Inequality) AS Inequality,
    SUM(Unknown) AS Unknown
FROM
    sum_model_5
GROUP BY
    id_chat, text
ORDER BY
    id_chat;
```

Listing 3: MySQL query used to aggregate classification results into `totals_model5.csv`

id_chat	text	Coordination	Efficiency	Conflict	Inequality	Unknown
1	Participants...	20	0	0	0	0
2	Participants...	19	0	0	1	0
3	Participants...	20	0	0	0	0
4	Participants...	18	0	0	0	2
5	Participants...	20	0	0	0	0
			...			
170	The players...	11	2	0	0	7
171	The players...	10	2	7	0	1
172	The players...	11	0	4	0	5
173	The group...	11	2	0	0	7
174	The group...	3	4	0	0	13

Table 4: Preview of the `totals_model5.csv` dataset showing the first and last few summaries.

Using the `totals_model5.csv` file, the following results were generated and visualized in the two charts below:

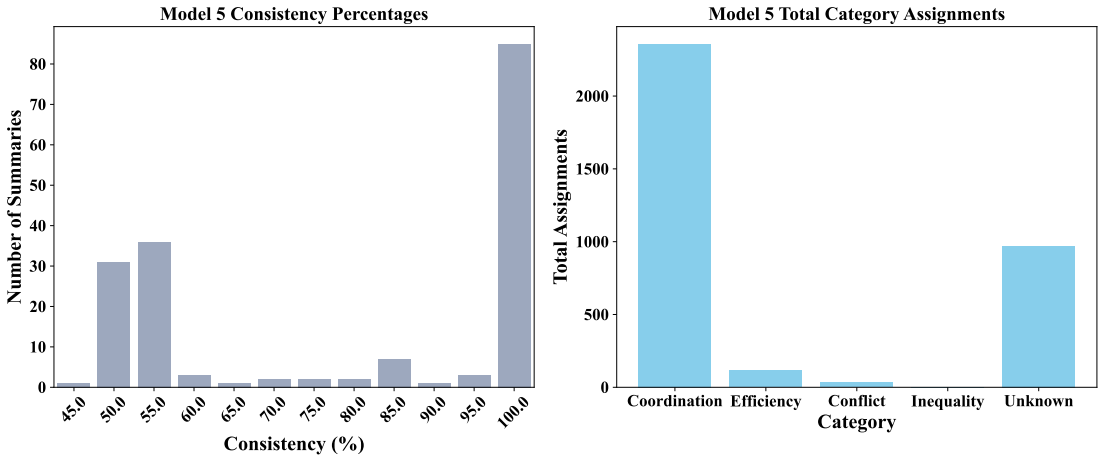

Figure A: Model 5 Consistency Percentages. Figure B: Model 5 Total Category Assignments.

In Figure A, for each summary, consistency was computed as the maximum count among the five category labels divided by 20:

$$\text{Consistency}_i = \frac{\max(\text{Coord}_i, \text{Eff}_i, \text{Conf}_i, \text{Ineq}_i, \text{Unk}_i)}{20}$$

These values were converted to percentages and grouped into bins for visualization. The left plot shows how many summaries achieved each level of consistency (e.g., 100%, 95%, etc.). Higher consistency means a summary was classified into the same category across most or all runs.

Figure B summarizes how often each category was assigned across all 20 simulations. The model strongly favored the labels **Coordination** and **Unknown**, which together accounted for **95.7%** of all classifications:

- **Coordination:** 67.8%
- **Unknown:** 27.9%
- **Efficiency:** 3.3%
- **Inequality:** 0.9%
- **Conflict:** 0.1%

This indicates a strong skew toward the **Coordination** and **Unknown** categories, suggesting that the model consistently preferred these classifications over the others.

Model 6 Prompt and Input

To improve consistency, we moved from Model 5 to Model 6. Since 95.7% of Model 5 outputs were already classified as **Coordination** or **Unknown**, Model 6 restricted labels to just these two categories.

The prompt used for **Model 6** was as follows:

Model 6 Prompt

Use the CSV file with summaries. I want you to classify each summary based on the following categories. The classification must be mutually exclusive—each summary can be classified into only one category.
Categories: Coordination, Unknown. The output should include binary category columns.

The prompt used the same input CSV file, `id_chat_input.csv`.

Following the same workflow as Listing 1, 20 simulations were run manually using ChatGPT-4o via the OpenAI web interface, followed by 80 automated simulations using Python and the OpenAI API.

Model 6 Results

Following the same workflow as Listings 2 and 3, the individual simulation outputs were combined into a single file, `sum_model_6.csv`. A MySQL query was then used to aggregate these results into `totals_model6.csv`, which summarizes how many times each summary was labeled as **Coordination** or **Unknown** in all 100 simulations.

id_chat	text	Coordination	Unknown
1	Participants agreed to say yes...	100	0
2	Participants agreed to always ...	100	0
3	Participants agreed to trade a...	100	0
4	Participants agreed that every...	100	0
5	Participants at first were uns...	100	0
...	...	...	...
170	The players agree that always ...	91	9
171	The players initially agree to...	93	7
172	The players debate the best st...	90	10
173	The group discusses their stra...	90	10
174	The group discusses sticking w...	83	17

Table 5: Preview of the `totals_model6.csv` dataset showing the first and last few summaries.

Using the `totals_model6.csv` file, the following results were generated and visualized in the two charts below:

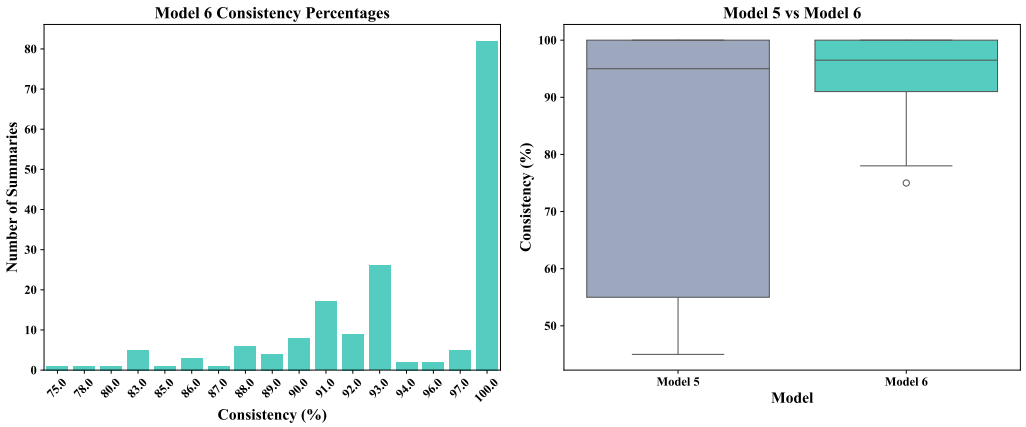

Figure C: Model 6 Consistency Percentages. Figure D: Model 5 vs Model 6.

In Figures C and D, to evaluate the consistency of classification results across multiple simulations, we define consistency for each summary i as the proportion of simulations in which the most frequently assigned category was chosen. The formulas differ based on the number of categories and simulations used in each model.

Model 5 (Five Categories, 20 Simulations):

$$\text{Consistency}_i^{(5)} = \frac{\max\left(\text{Coord}_i, \text{Eff}_i, \text{Conf}_i, \text{Ineq}_i, \text{Unk}_i\right)}{20}$$

Model 6 (Two Categories, 100 Simulations):

$$\text{Consistency}_i^{(6)} = \frac{\max\left(\text{Coord}_i, \text{Unk}_i\right)}{100}$$

After applying these formulas to `totals_model5.csv` and `totals_model6.csv`, we compute:

$$\bar{C}_{\text{Model5}} = \frac{1}{174}\sum_{i=1}^{174} \text{Consistency}_i^{(5)} = 0.6920 \quad \Rightarrow \quad \textbf{69.20\%}$$

$$\bar{C}_{\text{Model6}} = \frac{1}{174}\sum_{i=1}^{174} \text{Consistency}_i^{(6)} = 0.9547 \quad \Rightarrow \quad \textbf{95.47\%}$$

This shows that Model 6 produced significantly more consistent classifications across 100 simulations compared to Model 5's 20, even though Model 5 used a more complex, multi-category schema.

The box plot for Model 5 shows a wide range of consistency scores, with many summaries falling short of full agreement and a noticeable right skew, indicating inconsistent results across 20 simulations. In contrast, Model 6, run across 100 simulations, displays a highly concentrated distribution near 100% consistency, with minimal spread and few outliers. This highlights a clear improvement in classification reliability.

4 RA Summary Classification Comparison

To measure agreement between the two research assistants (RAs) on summaries of the same game, a *soft similarity score* was calculated for each of the 87 chat logs. Soft similarity is a metric that measures how close two sets of numerical scores are by capturing the degree of difference rather than requiring exact matches. It is mathematically appropriate for this analysis, as it is well-suited for comparing percentage-based classifications and accounts for partial agreement across multiple continuous categories. The similarity for each summary was computed using the following formula:

$$\text{Similarity}_i = 1 - \frac{1}{N}\sum_{j=1}^{N} \frac{|\text{RA1}_{ij} - \text{RA2}_{ij}|}{100}$$

where RA1_{ij} and RA2_{ij} represent the scores given by RA1 and RA2, respectively, for category j in summary i, and N is the total number of categories being compared. For Model 5, we used five categories ($N = 5$), and for Model 6, only two categories ($N = 2$) were available.

Applying this formula, per-summary similarity scores were computed and then averaged them across all 87 summaries.

The resulting mean similarity for Model 5 was:

$$\overline{\text{Similarity}}_{\text{Model 5}} = 96.34\%$$

For Model 6, the mean similarity was:

$$\overline{\text{Similarity}}_{\text{Model 6}} = 91.26\%$$

These percentages represent the average closeness between the two RAs' summary classification across all game chat logs. A 96.34% score for Model 5 indicates that the two RAs gave very similar classifications across five categories per summary. In contrast, the 91.26% score for Model 6 reflects slightly lower agreement, even though only two categories were considered. This suggests that, despite Model 6's simpler structure, interpretation differences particularly in the "Unknown" category, may have contributed to more variability in classifying.

5 Key Results and Conclusion

This project evaluated the reliability and consistency of GPT-4o in classifying textual summaries of chat logs from experimental game trials. Three classification models (Model 4, 5, and 6) were developed and tested, each with progressively refined prompts and category structures.

Key Results

- **Model 4** allowed multi-label classification across five categories. While conceptually flexible, it produced inconsistent and overlapping labels, particularly overusing *Coordination* and *Efficiency*, and rarely assigning *Conflict* or *Inequality*. The results were not reproducible across simulations.
- **Model 5** introduced mutually exclusive classification across the same five categories and was evaluated across 20 simulations. It showed a strong bias toward *Coordination* (67.8%) and *Unknown* (27.9%), together accounting for 95.7% of outputs. However, the average consistency across summaries was only **69.20%**, indicating instability in repeated model outputs.
- **Model 6** addressed this by restricting categories to only *Coordination* and *Unknown*. Across 100 simulations, it achieved an average consistency of **95.47%**, with a sharply concentrated distribution near 100% and minimal variability, demonstrating a significant improvement in classification stability.
- **RA similarity analysis**, using a soft similarity metric, revealed strong overall agreement between annotators: **96.34%** for Model 5 (five categories) and **91.26%** for Model 6 (two categories). The lower agreement in Model 6 indicates that, even after reducing the number of categories, there was higher variation in how summaries were interpreted.

Conclusion

Through iterative prompt refinement and category reduction, GPT-4o demonstrated highly consistent binary classification of textual chat summaries when limited to clear, distinguishable categories. While the broader, multi-category approach in Model 5 introduced ambiguity and variability, the simplified structure of Model 6 produced stable and reliable outputs suitable for further quantitative evaluation. This shows that with precise prompt engineering and careful task design, large language models like GPT-4o can effectively support automated classification in experimental research.