STAT496 Final Paper

Team Wise Wyver

February 2026

1 Introduction

Large language models (LLMs) are now routinely used for classification, information extraction, and summarization in both research and industry. But recent work shows that LLM outputs can be surprisingly sensitive to how a prompt is worded and what generation parameters are used. This is a problem for reproducibility: if the same model gives different answers depending on minor prompt changes, any automated pipeline built on top of it may produce unreliable results.

Brown et al. (2020) showed that large language models can pick up new tasks from examples placed directly in the prompt, without any fine-tuning. Wei et al. (2022) later found that chain-of-thought prompting, where the model is asked to show intermediate reasoning steps, improves performance on complex tasks. These results show that prompt design matters a lot, but they also raise an uncomfortable question: if small changes to a prompt can shift model behavior, how stable are the outputs? Zhao et al. (2021) investigated this directly and found that even minor differences in prompt wording or example ordering can produce substantially different outputs, even when the task instructions stay the same. In short, LLM responses behave more like stochastic samples than deterministic computations.

Beyond prompt design, the temperature parameter also affects output variability. Higher temperature flattens the probability distribution over tokens, producing more diverse (and less predictable) responses. Lower temperature concentrates probability on the most likely tokens, pushing outputs toward determinism. Practitioners adjust temperature all the time, but there is little systematic work on how temperature interacts with prompt structure and model choice. Similarly, newer models are often assumed to be more consistent than older ones, but empirical comparisons of stability across model versions are sparse.

Our project focuses on reproducibility rather than raw accuracy. We treat the LLM as a black-box stochastic system and ask: what makes its outputs more or less stable? We vary three factors, the number of in-context examples, the model version (GPT-3.5-Turbo, GPT-4.1-mini, and GPT-4.1), and the temperature setting, then measure how consistent the outputs are across repeated

trials. The goal is practical: if you are building a system that depends on LLM outputs being repeatable, which settings should you use?

2 Experimental Design

We use a sentiment classification task on real-world Twitter data: each tweet is classified as positive, neutral, or negative. Sentiment classification is a good fit for this study because the output format is simple and unambiguous, which makes it straightforward to compare results across runs. A more open-ended task (e.g., summarization) would make it harder to tell whether differences between runs are meaningful or just stylistic.

2.1 Task and Dataset

The dataset is a collection of publicly available Twitter posts, each labeled by human annotators as positive (1), negative (-1), or neutral (0). We use the first 50 tweets as our test set. The remaining tweets are the example pool: when we need labeled examples for few-shot or many-shot prompts, we draw them from this pool. For evaluation, we convert the integer codes to text labels and compare them against model predictions.

2.2 Independent Variables

The experiment varies three independent variables in a fully crossed design, producing $4 \times 3 \times 3 = 36$ total experimental conditions:

1. **Prompt structure** (4 levels):
   - *Zero-shot*: Only the task instruction and tweets are provided.
   - *Definitions*: Task instruction plus explicit definitions of each sentiment category (e.g., positive: "support, praise, favorable views").
   - *Few-shot* (3 examples): One labeled example per sentiment class is included before the test tweets.
   - *Many-shot* (10 examples): Three positive, three negative, and four neutral labeled examples are included.
2. **Model version** (3 levels): GPT-3.5-Turbo, GPT-4.1-mini, and GPT-4.1. These represent three different generations and scales of OpenAI's language models.
3. **Temperature** (3 levels): 0, 0.5, and 1.0. Temperature controls the randomness of token sampling during generation, with 0 producing near-deterministic outputs and 1.0 allowing maximum diversity.

2.3 Prompt Templates

All prompts share the same base instruction: "Classify each tweet as positive, neutral, or negative. One label per line." The prompts differ only in what context precedes the list of test tweets. Below is an abbreviated example of the few-shot prompt template:

```
Classify each tweet as positive, neutral,
or negative. One label per line.

Examples:
Tweet: I love this new update!
Sentiment: positive

Tweet: Terrible service, never again.
Sentiment: negative

Tweet: The event starts at 5pm.
Sentiment: neutral

Tweets:
1. [test tweet 1]
2. [test tweet 2]
...
50. [test tweet 50]

Output:
```

The zero-shot prompt omits the examples section entirely. The definitions prompt replaces the examples section with explicit category definitions. The many-shot prompt includes 10 labeled examples instead of 3.

2.4 Execution Details

The prompt conditions follow the standard in-context learning setup. Zero-shot prompts give only the task instruction. Few-shot and many-shot prompts add labeled examples before the test tweets. Brown et al. (2020) showed that more examples tend to improve accuracy; we want to know whether they also make outputs more stable across runs.

We test three OpenAI models: GPT-3.5-Turbo (an older, smaller model), GPT-4.1-mini (a compact version of the GPT-4.1 family), and GPT-4.1 (the full-scale model). Most prior work compares models on accuracy benchmarks. Whether newer or larger models also produce more consistent outputs is less studied, and that is what we are after here.

We test three temperature levels: 0, 0.5, and 1.0. At temperature 0, the model almost always picks the highest-probability token, so outputs should be nearly deterministic. At temperature 1.0, the sampling distribution is unmodified, so we expect more variation between runs. Temperature 0.5 sits in between.

By crossing temperature with prompt type and model, we can see whether the effect of temperature depends on those other factors.

All 36 conditions are run through an automated Python pipeline that calls the OpenAI API. For each condition, the pipeline sends the prompt as a single user message and records the response. We cap responses at 250 tokens (enough for 50 one-word labels). The pipeline parses each response by splitting on numbered patterns (e.g., "1. Positive") and extracts the label text, keeping at most 50 predictions. Accuracy is the fraction of predictions that match the ground truth. Results are saved to CSV files for analysis.