Philip Georgis
January 9, 2025

IMDB Film Review Sentiment Analysis with SLMs

1. Overview

The present study represents a comparative prompt engineering experiment for performing sentiment analysis on film reviews with so-called “small language models” (henceforth “SLMs”). The aim of this study is to identify the prompt structure and SLM which achieves optimal performance on binary classification of film reviews as either positive or negative. Four prompting techniques are evaluated on a subset of the IMDB dataset of film reviews with two SLMs of differing sizes:

a) Zero-shot prompting
b) Few-shot prompting
c) Chain-of-thought prompting
d) Prompt chaining with keyword-based sentiment analysis

2. Development Process

The development process of this project proceeded according to the following workflow:

- Frame scope of experiment and select prompt techniques to test.
- Build flexible orchestration architecture for locally constructing prompts, querying SLMs, and logging or saving all required details for later analysis.
- Examine dataset structure and implement tools for balanced data loading, preprocessing, and selection.
- Build end-to-end framework for querying SLMs with multiple prompts and/or model hyperparameters and evaluating results on selected data.
    - Continually improve flexibility or efficiency as issues or concerns arise.
- Write preliminary versions of all prompts and conduct pilot tests on training data partition.
- Iteratively refine prompts and set appropriate hyperparameter values.
    - Review selection of errors or incorrectly classified results and adjust prompts, pre- or post-processing, or hyperparameters as needed.
    - Keep adjustments if they improve results on pilot test data from training partition.
- Once satisfied with results on pilot test data, perform final test run on test partition.

3. Data

The dataset chosen for this study is the IMDB movie review dataset accessed from HuggingFace, which has been a popular choice for benchmarking NLP sentiment analysis models. This dataset contains 50,000 film reviews classified binarily as either positive or negative. Each entry consists of a film review text and a binary sentiment label (0 for positive and 1 for negative reviews).[1] These 50,000 reviews are split between 40,000 training examples and 10,000 test examples. In accordance with standard data science and machine learning conventions, all development work was performed using only the training split while the results presented here are from the test split.

Due to time and resource constraints, only a small subset of this dataset was selected for analysis in this study. The test sample consisted of 500 reviews representing positive and negative reviews equally (250 of each), which could be evaluated against all tested prompts in under two hours. As film review texts can vary greatly in length and it was hypothesized that sentiment analysis might be more or less accurate depending on the length of the review, a sample including reviews of diverse lengths is desirable. A representative sample according to text length was ensured by dividing the relevant dataset partition into bins according to the standard deviation of all (preprocessed) film review word counts, with reviews from each bin sampled proportionately to the bin’s frequency in the full dataset partition.[2] This sampling

[1] As part of preprocessing, the binary labels were reversed such that 1 represents a positive review and 0 a negative review, a more intuitive labeling scheme.
[2] For example, given $N = 100$ and that Bin 3 represents 40% of all reviews, 40 reviews are sampled from Bin 3.

method ensures that texts of roughly all attested lengths are present in the sample, without overrepresenting unusually long or short texts.

Preprocessing was applied prior to sampling to remove HTML markup within the film review texts. Unlike sentiment analysis with more traditional NLP techniques, no further text preprocessing (e.g. stop word removal, normalization, etc.) was required for this application as LLMs and SLMs are expected to be robust to such raw natural language inputs.

4. Models

The present study was performed using two SLMs of differing sizes from the Qwen2.5 family. Both are quantized versions of an originally larger Qwen2.5 model.

Table 1. SLMs used in the present study

Name	**Parameters**	**Source**
Qwen2.5-0.5B	500 million	Qwen2.5-0.5B-Instruct-Q5_K_M.gguf
Qwen2.5-1.5B	1.5 billion	Qwen2.5-1.5B-Instruct-Q5_K_M.gguf

The models were queried using the Python library llama-cpp-python, which provides an API for submitting prompts with a variety of configurable hyperparameters and retrieving chat completion responses. A local implementation was created which enables flexible construction and submission of prompt payloads together with logging of token usage and latency.

a. Hyperparameters

For this study, the following SLM hyperparameter values were selected:

- `temperature = 0`

The `temperature` hyperparameter controls the degree of randomness in a generative language model's responses by scaling log probabilities by the temperature. Higher temperature values yield more variable and creative responses, whereas lower temperatures ensure more consistent outputs, with a temperature of zero always yielding the single most likely response. The minimum temperature of zero was selected for these experiments given that the binary sentiment classification task at hand requires only single-word responses whose possible values are limited to "positive" and "negative", therefore neither creativity nor diversity of responses is advantageous for this task.

- `top_k = 5`

The `top_k` hyperparameter filters the possible response space to only those `k` most likely tokens at each step. A fairly low value of 5 was selected to significantly restrict the model's possible outputs, according to the same rationale as for the low temperature value. A preliminary test of three possible `top_k` values [5, 10, 40] confirmed that 5 performed best for this application.

- `top_p = 0.1`

The `top_p` hyperparameter restricts the possible response space differently than `top_k`, instead by filtering the possible tokens at each step to the likeliest subset whose cumulative probability sums to `top_p`. According to the same logic as discussed for the other hyperparameters, a low value was selected here to further enforce responses belonging to the narrow set of likeliest tokens. A preliminary test on 50 examples showed improved accuracy with `top_p = 0.1` over `top_p = 0.99`.

The exception to the above hyperparameter configuration is the first prompt of the prompt chaining technique (see §5d), where default API values[3] for all three hyperparameters were used instead, in order not to restrict responses as described above.

[3] Default values from llama-cpp-python API: `temperature` = 0.2, `top_k` = 40, `top_p` = 0.95.

5. Prompts

Large language models (LLMs) and their smaller cousins SLMs operate on natural language inputs known as prompts to perform a huge variety of tasks. Since the advent of LLMs, a variety of prompting techniques have been developed and gained popularity, though the process of developing an appropriate prompt for a given application is highly specific to the task and data at hand. Nevertheless, some general principles are almost universally useful for obtaining effective prompts, namely:

- Use clear and simple wording to explain the task. Sometimes the best prompts are the simplest prompts.
- Avoid anaphora: rather than referring to an undefined "it" and expecting the model to understand what the earlier referent was, explicitly restate what "it" refers to.
- LLMs have been shown to struggle with negation, so use positive imperatives rather than negative imperatives (e.g. *avoid X* rather than *don't do X*) wherever possible.
- Clearly demarcate instructions from other prompt components or inputs.
- Provide instructions for and, ideally, examples of the expected response format.

In addition to the above guidelines for the prompt body, it is useful to provide a system prompt to prime the model as to the role it should assume before providing any task-specific instructions. All four prompting techniques in this study used the following system prompt that primes the SLM for the task at hand:

```
system: You are an expert in interpreting and classifying film reviews.
```

The following sections provide an overview of the four prompting techniques examined in this study.

a) Zero-Shot Prompt

The simplest prompt type is known as a zero-shot prompt and is characterized only by instructions for the task without any further demonstration or examples. The model is expected to be able to carry out the task largely based on its existing capabilities. This technique can be effective for straightforward tasks where limited external knowledge or logic is required and/or where possible responses are limited.

The zero-shot prompt template used in this study is shown below with a placeholder where the film review to be classified would appear:

~~~
Carefully read the following film review and decide whether the overall review is positive
or negative.
Return only the labels "positive" or "negative". No further explanation is needed.

```
{film review}
```
~~~

This prompt is concise and clearly conveys the most essential details of the task without frills. The instructions refer to the "overall" review, as some reviews contain subsections with statements that contradict the overall sentiment, e.g. an overall negative review which praises certain aspects of the film, or an overall positive review with some minor complaints or critique.

This prompt, along with the other prompt techniques in this study, explicitly indicates the expected output format, i.e. either "positive" or "negative". Based on prior prompting experience, the final phrase "No further explanation is needed" is quite effective for limiting the response to only the explicitly mentioned output labels and avoiding any further attempts by the model to explain or justify its choice. Indeed, only one out of 2000 (0.05%) SLM calls in this study (500 examples x 4 prompt techniques) resulted in a response other than "positive" or "negative".

b) Few-Shot Prompt

Unlike zero-shot prompt methods, one-shot and few-shot prompting methods provide at least one example of how to complete the task, typically in the form of a list of input and expected output pairs.

The few-shot prompt used in the present study is an extension of the zero-shot prompt with N examples each of positive and negative reviews from the train dataset partition. In this case, N was set to 3. Due to time constraints, the examples shown in the prompt are drawn at random from the train set. Given more time, a more principled approach might have tested multiple sets of examples to identify the most useful ones, or perhaps could have computed an embedding similarity measure to dynamically retrieve similar review texts from each category as examples to aid in classifying a specific film review.

The prompt is structured by first providing a preview of the examples to follow, which helps the model understand their structure. The series of clearly numbered and delimited examples is then introduced one-by-one, together with explicit annotation of the corresponding labels. Following this are the same instructions used for the zero-shot prompt, which immediately precede the review to be classified by the model. An example of this few-shot prompt is shown below with just one example from each class.

~~~
Carefully study the following 2 examples of film reviews with their respective classifications
as either positive or negative.

*** Example film review #1 ***
```
I was really impressed with this film. The writing was fantastic, and the characters were
all rich, and simple. It's very easy to get emotionally attached to all of them. The creators
of this movie really hit the nail right on the head when it comes to creating real life
characters, and getting the viewer sucked right into their world. Further, the music is
terrific. They employed some independents to do the score, and some of the soundtrack, and
they do a fantastic job adding to the movie. If you have a chance to catch this movie in a
small theater or at a film festival (like I did), I highly recommend that you go see it.
Also, on a personal note, Paget Brewster is beautiful in this movie. That's reason enough to
go check it out.
```
Classification: "positive"

*** Example film review #2 ***
```
This is one of the worst films I've ever seen. I looked into it mainly out of a morbid
curiosity since I loved the novel, and I wish I hadn't. I turned it off after a little less
than an hour, though I wanted to turn it off after five minutes. I wish I had. It disregards
the novel a lot and changes all sorts of factors. Unless the film managed to redeem itself
in the last 50 or so minutes (which would be impossible) I would in no way recommend this.
Its an insult to one of the greatest writers of the 20th century. I don't think, as many
people say that it is, that 'The Bell Jar' is necessarily unfilmable, but this particular
rendition could have been done without. I'd almost like to see this one day in the hands of
a director and screenwriter who can do it justice.
```
Classification: "negative"


Carefully read the following film review and decide whether the overall review is positive
or negative.
Return only the labels "positive" or "negative". No further explanation is needed.

```
{film review}
```
~~~

Prior to implementing example selection from the training data, synthetic reviews from each category generated by ChatGPT were used instead as a preliminary test of this method. This pilot testing revealed that the few-shot prompt performed, contrary to expectations, worse than all other prompts, especially with the smaller Qwen model (F1 ≈ 0.75-0.80). It was hypothesized that genuine reviews from the

IMDB dataset itself might constitute more representative examples of the texts the model would encounter, as opposed to externally generated synthetic reviews, and therefore improve performance. Indeed, this approach proved more successful and improved accuracy with both Qwen models to F1 ≈ 0.87-0.93.

c) Chain-of-Thought Prompt

Another prompting technique which has enjoyed widespread popularity is the so-called "chain-of-thought" prompting method, whereby the prompt encourages the model to "think step-by-step" and break down the task into component steps in order to use intermediate logic to arrive at a well-reasoned result. While some have claimed that including the emblematic phrase "let's think step-by-step" to the prompt can be enough to improve reasoning on its own, it is typically better to lay out these intermediate steps and/or logic explicitly within the prompt for the model to follow.

For the task of sentiment analysis, a series of intermediate steps might include identification of relevant keywords or phrases within a review text that reveal the author's stance, followed by an analysis of whether these suggest a positive or negative attitude. As previously discussed, some reviews contain both praise and critique, so it is also important to consider the entire review and verify that the overall tone is consistent with the sentiment of the identified key phrases.

~~~
Carefully read the following film review and decide whether the overall review is positive
or negative.
Let's think step-by-step:
- Identify the key words or phrases which reveal the author's attitude toward the film.
- Determine whether these attitudes are generally positive (e.g. impressed, pleased, moved,
excited) or generally negative (e.g. bored, disgusted, disappointed, confused).
- Confirm that this positive or negative sentiment matches the overall tone of the review.

Return only the labels "positive" or "negative". No further explanation is needed.

```
{film review}
```
~~~

In addition to the chain-of-thought steps, this prompt further provides several examples of possible author attitudes to look out for, which each might support the overall analysis of the review as positive or negative.

d) Prompt Chaining with Keyword-Based Sentiment Analysis

Prompt chaining refers to the technique of using multiple SLM or LLM queries, either back-to-back or with non-AI pre-/post-processing steps in between, to accomplish a task, especially a task which might ordinarily be too complex for a single prompt to handle end-to-end. Its theoretical underpinnings are similar to those of chain-of-thought prompting, in that it seeks to guide the model to follow a sequence of intermediate logic by splitting up tasks into component steps, though unlike chain-of-thought prompting these steps are performed through separate model calls. This often implies increased latency since more model calls are required to achieve the same result, though for complex tasks such an approach may be more effective than a single prompt.

The prompt chain used for this sentiment analysis task consists of two prompts and closely parallels the component steps mentioned in the chain-of-thought prompt. The goal of the first prompt is to identify and extract the keywords or key phrases from within the overall text upon which the ultimate classification decision will be based:

```
Carefully read the following film review and identify any keywords or key phrases which
reveal the author's attitude toward the film.
In particular, look for keywords and phrases which reveal:
- the author's opinion about the film
- the author's emotional reaction to the film, or how the film made the author feel
- criticism or praise of the film
```

~~~~
Return a list of relevant keywords or key phrases from the film review. No further
explanation is needed.

```
{film review}
```
~~~~

The first prompt was also used with a variation on the standard system prompt used elsewhere in this study (including for the second prompt of this method), which better primes it for this variation of the task:

~~~~
system: You are an expert in interpreting and summarizing film reviews. You are highly
skilled in identifying relevant key words and phrases from film reviews which reveal the
author's opinions.
~~~~

The second prompt takes the list of key expressions output from this first prompt and instructs the SLM to perform sentiment analysis on them, as opposed to on the whole review:

~~~~
Carefully read the following keywords and/or key phrases taken from a film review and
decide whether the overall review is positive or negative.
Return only the labels "positive" or "negative". No further explanation is needed.

```
{key expressions}
```
~~~~

For film reviews consisting of hundreds or thousands of words, this two-step keyword-based approach might represent a significantly simpler task compared to classifying the sentiment of a full review text, as irrelevant content (e.g. long-winded descriptions of the plot) will have been removed, leaving only the most relevant expressions. On the other hand, this runs the risk of taking phrases out of context, which could be crucial for their correct interpretation, especially when the author takes a sarcastic or joking tone.

Given time constraints and the extra time involved in making two separate SLM calls per test example, the first prompt for keyword extraction was submitted only to the better-performing Qwen2.5-1.5B model, and its responses were cached in order not to repeat this step. Therefore, any difference in performance between the Qwen2.5-1.5B and Qwen2.5-0.5B models with the prompt chain technique can be attributed solely to the second prompt for sentiment analysis.

6. Evaluation

Given that sentiment analysis with the IMDB dataset essentially amounts to a binary classification task, F1 score was selected as the primary evaluation metric as it can be suitably calculated for binary classification contexts from counts of true and false positives and negatives. A secondary evaluation metric was average latency, or the average time required for the SLM to respond to the prompt. Not considered as a performance metric in this study given that the SLMs used were free of charge to use, but ordinarily important for real-world production scenarios is token usage, as many production-grade SLMs/LLMs charge fees according to the number of tokens consumed, encompassing both input and output tokens. Table 2 below summarizes the performance of each prompt and model according to these metrics, with the best score per metric in boldface.[4]

[4] TP, TN, FP, FN refer to true positives, true negatives, false positives, and false negatives, respectively.

Table 2. Summary of prompt and model performance

Prompt Technique	Model	F1 Score	TP	TN	FP	FN	Latency (seconds)	Total Tokens
Zero-shot	Qwen0.5B	0.90	236	213	37	14	**0.47**	**357.3**
	Qwen1.5B	0.93	229	239	11	21	1.45	
Few-shot	Qwen0.5B	0.87	**245**	184	66	**5**	0.86	1720.3
	Qwen1.5B	0.93	223	241	9	27	2.11	
Chain-of-thought	Qwen0.5B	0.90	230	219	31	20	0.58	432.3
	Qwen1.5B	**0.94**	225	**244**	**6**	25	1.54	
Keyword-based prompt chaining	Qwen0.5B	0.84	243	167	83	7	4.35	670.7
	Qwen1.5B	0.89	227	215	34	23	4.76	670.8

Unsurprisingly, the larger 1.5 billion parameter SLM performs better than the smaller 500 million parameter SLM across the board, although both models and all four prompting techniques ultimately achieve impressive scores with F1 ≥ 0.84.

Also unsurprisingly, the larger SLM has considerably longer latencies than its smaller counterpart, in some cases requiring twice or three times as long to respond to the same prompt. In a real-world application where every second saved is crucial for ensuring a positive user experience, this could call into question the choice of best model-technique combination given these results. The smaller model performs only slightly worse than the 1.5 billion parameter model, and depending on the requirements of the application, this slightly decreased accuracy might be preferred over such a significant increase in latency. Figure 1 illustrates this tradeoff between F1 score and latency in a scatterplot.

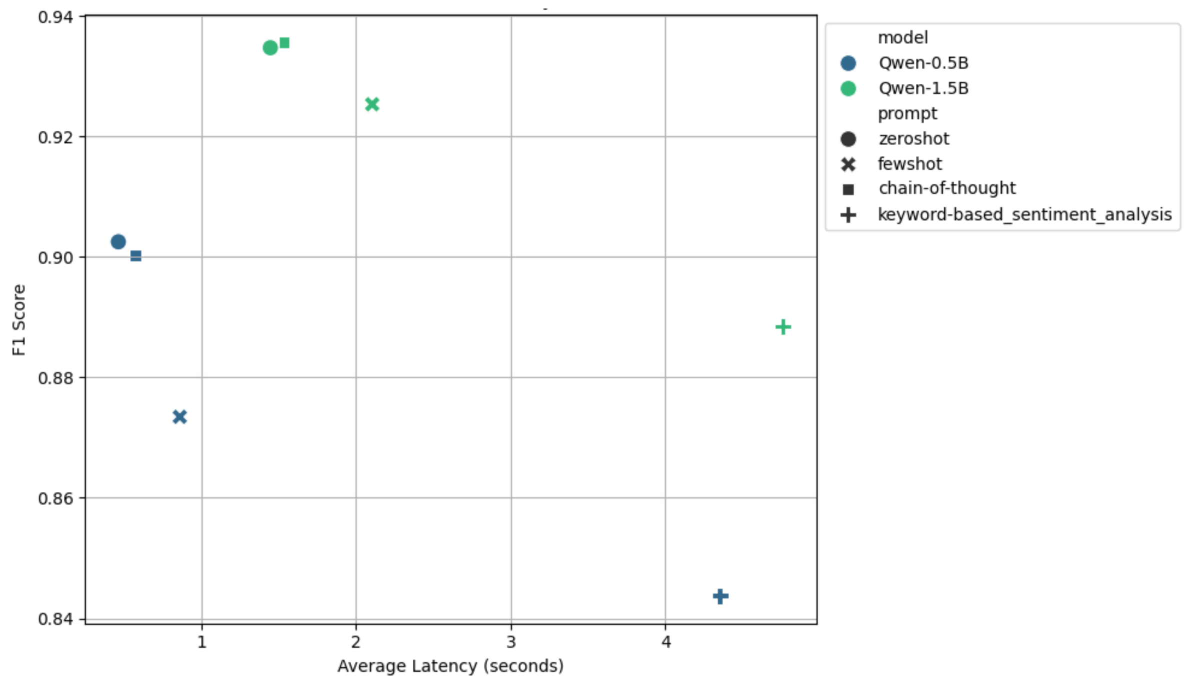

Figure 1. Scatterplot of model-prompt latency against F1 score

Considering the results presented in Table 2 and Figure 1, the zero-shot and chain-of-thought prompts appear to be the most effective for this context, both achieving F1 ≥ 0.9 with both SLMs. Amazingly, the zero-shot prompt was even the best performing technique with the smaller model. The few-shot prompting method also achieved F1 > 0.9, though only with the larger model. By far the worst-performing method, when considering both F1 and latency, is the keyword-based sentiment analysis using prompt chaining, which requires over 4 seconds on average and yet does not manage to match the performance of the faster prompts.

7. Analysis

It is perhaps surprising that the simplest prompting techniques achieved the best results, especially in the case of the zero-shot prompt, which was originally intended as a baseline for the most basic possible prompt technique. Indeed, a challenge encountered throughout these experiments was, in fact, improving the more sophisticated prompting techniques such that they could achieve even close to the same performance as the zero-shot method – it was expected that techniques such as few-shot and chain-of-thought prompting would easily outperform zero-shot prompting, which does not even cite examples! On the other hand, given that the task examined in this study is relatively simple and requires little to no domain-specific or external knowledge, it makes sense that the simplest, clearest instructions with the least amount of extraneous information or details to consider might end up being the most successful.

While unsurprising that it required so much longer compared to techniques that required only a single SLM call, it is nevertheless somewhat surprising that the prompt chaining approach performed so much more poorly than the others. The underlying logic behind this method posited that by removing irrelevant sections of film review texts and focusing only on relevant keywords and phrases for sentiment analysis, the model should be less distracted by extraneous details and better equipped to distinguish positive from negative reviews. The results suggest quite the opposite, that the full context of the review may actually be just as important as individual key words, or that the same phrasing could be interpreted differently depending on the wider context and tone. This is especially true for sarcastic comments, which appear in many reviews, and which appear to have the opposite meaning when removed from their context.[5] A final consideration for this technique is that it assumes that the output from the first prompt is well-formed and that the keywords are appropriate. Browsing some of the actual test results, this does not appear to always be the case.

Across the four prompt techniques, there appears to be a bias on the part of the model toward classifying reviews as positive, with false positive outnumbering false negatives in sum. This trend is reversed when looking only at the larger SLM, however. As the prompts and inputs are identical in both cases, it remains unclear why exactly this may be. However, in a real-world context, this, too, could factor into a decision of which model to select. In a production scenario where sentiment classification is, for instance, applied to customer reviews of a product, false positives would likely be judged as worse than false negatives as this would mean that negative customer feedback may go undetected.

A final interesting note is the performance gain mentioned in §5b by moving from synthetic to genuine examples in the few-shot prompt. Even though the examples for the few-shot prompt were not selected in any principled manner, simply including real examples from the IMDB dataset drastically improved F1 compared to using synthetic examples written by another AI tool. This highlights the importance of using real examples as opposed to AI-generated examples whenever possible, and calls to mind concerns about training future AI models using AI-generated content.

8. Conclusion

The current study has presented the results of several experiments with SLMs for the purpose of binary sentiment analysis on a subset of the IMDB movie review dataset. Four prompting techniques were evaluated with two SLMs of differing sizes. While the larger of the two models unsurprisingly outperformed the smaller model in accuracy, the smaller model nevertheless achieved respectable F1 scores almost as high as those of the larger model, typically in a fraction of the time. Though also the simplest, zero-shot and chain-of-though prompting techniques proved to be the most successful methods given this task and data, underscoring the importance of not overcomplicating tasks where a simple solution performs just as well as or better than a theoretically more sophisticated solution.

[5] An attempt was made to handle sarcastic language by including a warning message in the prompt, instructing the model to beware of sarcasm when determining the sentiment of a film review. This warning had no effect on the larger SLM and in the smaller model its effect was actually an overcorrection which yielded additional false negatives, while not reducing the number of false positives at all. A similar warning to ignore sarcastic comments when extracting relevant key phrases for sentiment analysis had the same counterproductive effect.