Text Classification Using "Small" Large Language Models, Clues and Reasoning, and Low Computer Resources

Daniel Miller

Abstract

Traditional text classification (TC) techniques require significant time and computational resources to train models and evaluate results. Recently, Large Language Models (LLMs) such as GPT-3 have emerged as an alternative approach to solving classic TC problems. This method eliminates the need for extensive training by leveraging pre-trained LLMs to classify text directly. However, the primary drawback is that running full-scale LLMs requires expensive and often unattainable hardware, making them impractical for most individuals and small businesses.

In this paper, we demonstrate that using small, quantized versions of well-known LLMs on consumer-grade hardware is a feasible and efficient solution for common TC tasks. We also show that prompting the model to provide reasoning and clues during classification significantly improves accuracy. Additionally, our approach requires only a small dataset for evaluation and does not rely on labeled training data, making it particularly useful for TC problems with limited annotated examples.

Our experiments reveal that while this method does not achieve state-of-the-art results, it produces highly usable outcomes, especially when prompt engineering is applied to use clues and reasoning. Furthermore, we performed no text pre-processing, suggesting that additional refinements could further enhance performance without increasing computational costs. Our implementation is available at https://github.com/1danmi/text-mining-project.

1. Introduction

In recent years, Large Language Models (LLMs) have proven useful for a wide range of tasks, including those they were never explicitly trained for, such as text classification (TC), named entity recognition, information extraction, and more. Research has also shown that the performance of LLMs in these tasks can be significantly improved through In-Context Learning (ICL). This technique involves providing the model with examples alongside the task description, guiding it on how to handle similar cases.

Using LLMs for TC problems offers two main advantages: (1) Unlike traditional machine learning (ML) and deep learning (DL) models, which require significant time and computational resources for training, pre-trained LLMs can classify text without the need for extensive model training. (2) Collecting labeled training data for ML models is expensive and time-consuming, particularly in fields like healthcare and law enforcement, where TC systems could provide the greatest benefits. By leveraging a pre-trained LLM, classification can be performed with minimal or even no training examples, eliminating the need for large, labeled datasets.

Despite these advantages, a major drawback of LLMs is their size and hardware requirements. To achieve reasonable performance and responsiveness, an LLM must store its entire model in RAM, preferably high-bandwidth video memory such as GDDR6. For instance, running Meta’s LLaMA 3.1 model at full scale requires a minimum of 243GB of GPU memory, while the smaller 70B version still demands 43GB. In comparison, as of February 2025, Nvidia’s most powerful consumer GPU, the RTX 5090, has only 32GB of video memory and costs at least $1,999 before tax. Most consumer GPUs are equipped with just 4 to 12GB of video memory, making it impractical for individuals or small businesses to run large-scale LLMs locally.

While some companies offer limited free cloud access to commercial LLMs, API access often comes with significant costs. For example, GPT-4o charges $2.50 per million input tokens and $10 per million output tokens. Running the datasets used in this paper through such an API would cost approximately $250 per single run, making this approach unsustainable for many users.

To address these challenges, the open-source community has developed quantized versions of well-known LLMs. Quantization reduces the numerical precision of model weights and activations—typically from 32-bit floating point (FP32) to 8-bit integer (INT8) or other lower-precision formats. This dramatically decreases the model’s size and computational demands, enabling it to run more efficiently on consumer-grade hardware.

In this paper, we evaluate the feasibility of using quantized LLMs for TC tasks on off-the-shelf consumer hardware. We demonstrate that these models can achieve decent classification results without any training and that their performance can be further improved using two key techniques: (1) In-Context Learning (ICL) – Providing the LLM with an example of how the classification task should be performed. (2) Clue and Reasoning Prompts – Asking the LLM to provide explanations and clues during classification.

Our results show that even without extensive computational resources, quantized LLMs can deliver practical and efficient text classification performance.

2. Related Work

2.1 Large Language Models

Large Language Models (LLMs) have evolved significantly over the past decade, driven by advances in deep learning, computational power, and large-scale datasets. The foundation of LLMs can be traced to early neural network-based language models, such as word embeddings introduced by (Mikolov, Chen, Corrado, & Dean, 2013). These models were followed by Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks, which improved sequence modeling (Hochreiter & Schmidhuber, 1997). However, the most significant breakthrough came with the introduction of the Transformer architecture by (Vaswani, et al., 2017), which eliminated recurrence in favor of self-attention mechanisms, enabling efficient parallelization and superior performance on Natural Language Processing (NLP) tasks.

Following the Transformer model, researchers developed increasingly larger and more capable language models. OpenAI’s GPT (Radford, Narasimhan, Salimans, & Sutskever, 2018) demonstrated the power of unsupervised pretraining on vast corpora, while BERT (Devlin, Chang, Lee, & Toutanova, 2009) introduced bidirectional contextual embeddings, improving many NLP benchmarks. Subsequent iterations, such as GPT-3 (Brown, et al., 2020) and (Chowdhery, et al., 2023), expanded model sizes into the hundreds of billions of parameters, showcasing emergent abilities and prompting-based learning. These advancements have made LLMs integral to applications in text generation, code completion, and conversational AI, while also raising ethical concerns related to bias, misinformation, and computational cost.

2.2 Quantization

Quantization of LLMs is a technique used to reduce their memory footprint and computational requirements while maintaining acceptable performance. As LLMs have grown in size - often exceeding hundreds of billions of parameters - their deployment on resource-constrained hardware has become increasingly challenging. Quantization addresses this by representing model weights and activations with lower-precision numerical formats, such as 8-bit or 4-bit integers instead of the standard 32-bit floating-point values (FP32). This reduces the model's storage requirements and accelerates inference by enabling more efficient computation on hardware accelerators like GPUs and TPUs.

The foundations of neural network quantization can be traced to early work on efficient deep learning, such as (Han, Mao, & Dally, 2015), which demonstrated that neural networks contain substantial redundancy that can be exploited for compression. More recent advances have tailored quantization techniques specifically for Transformers and LLMs. Shen et al. (2020) proposed Q-BERT (Shen, et al., 2020), which showed that preserving key parameters in higher precision can mitigate performance degradation. Additionally, innovations like GPTQ (Frantar, Ashkboos, Hoefler, & Alistarh, 2022) and (Lin, et al., 2024) have introduced post-training quantization methods optimized for large-scale models, enabling near-lossless performance while significantly reducing computational overhead. These advancements make it possible to run LLMs efficiently on edge devices and consumer-grade hardware, broadening their accessibility and applicability.

The open-source community has played a crucial role in making quantized LLMs more accessible, particularly through platforms like Hugging Face. Many pre-trained models, such as (Touvron, et al., 2023) and Mistral, have been optimized using quantization techniques like GPTQ and AWQ, allowing them to run efficiently on consumer-grade GPUs and even some CPUs. Hugging Face hosts repositories with ready-to-use quantized models, enabling researchers and developers to deploy powerful language models without the need for expensive hardware. These models facilitate applications ranging from chatbots to code generation, demonstrating the practicality of quantization for real-world use cases

2.3 Text Classification

Text classification is a fundamental task in natural language processing (NLP) that involves assigning predefined labels to textual data based on its content (Joachims, 1998). It has widespread applications, including spam detection, sentiment analysis, topic categorization, and intent recognition in chatbots. Traditional approaches relied on machine learning algorithms such as Naïve Bayes, Support Vector Machines (SVM), and logistic regression, which required manual feature engineering using techniques like term frequency-inverse document frequency (TF-IDF) and n-grams (Robertson, 2004). However, deep learning has revolutionized text classification with models like Convolutional Neural Networks (CNNs) (Kalchbrenner, Grefenstette, & Blunsom, 2014) and recurrent architectures like Long Short-Term Memory (LSTM) networks (Hochreiter & Schmidhuber, 1997). More recently, Transformer-based models such as BERT (Devlin, Chang, Lee, & Toutanova, 2009) and RoBERTa (Liu, et al., 2019) have set new state-of-the-art benchmarks by leveraging contextual embeddings and self-attention mechanisms, enabling more accurate and robust classification across diverse text datasets.

2.4 In-Context Learning

In-context learning (ICL) is a capability of large language models (LLMs) that allows them to perform tasks without explicit fine-tuning by leveraging examples provided in the input prompt. First observed in models like GPT-3 (Brown, et al., 2020), ICL enables a model to understand and generalize from a few demonstrations, making it adaptable to various tasks, including text classification, summarization, and translation. Unlike traditional supervised learning, which requires extensive labeled datasets and model updates, ICL allows users to guide the model's behavior dynamically through prompt engineering. Recent research has explored ways to enhance ICL efficiency, such as prompt optimization and retrieval-augmented generation (RAG) (Lewis, et al., 2020)which improve model performance by incorporating relevant contextual information. This approach has become a key component in deploying LLMs for real-world applications, reducing the need for computationally expensive fine-tuning.

2.5 CARP

(Sun, et al., 2023) created a novel framework aimed at improving the text classification capabilities of large language models (LLMs) such as GPT-3. CARP addresses two critical challenges: the necessity for advanced reasoning to manage complex linguistic phenomena, including irony and negation, and the constraints imposed by token limitations in in-context learning. The proposed approach consists of a two-step process wherein LLMs first identify salient textual features, such as keywords, tone, and semantic relations, before engaging in a structured diagnostic reasoning process to determine the final classification. Furthermore, CARP integrates a k-nearest neighbors (kNN) demonstration search mechanism, which enhances the effectiveness of LLMs by incorporating task-specific insights from labeled datasets while maintaining their generalization capabilities.

Empirical evaluations demonstrate that CARP substantially enhances classification performance, particularly in low-resource and domain adaptation scenarios. The method effectively mitigates the need for extensive labeled data by leveraging structured reasoning, allowing LLMs to achieve competitive performance even with a limited number of training examples. These findings suggest that CARP provides a promising direction for improving the efficiency and accuracy of text classification using LLMs.

3. Approach

To evaluate our hypotheses, we tested several of the most common text classification (TC) tasks. Specifically, we selected four widely used datasets covering Sentiment Analysis, Topic Classification, and Spam Detection. These datasets serve as well-established benchmarks, providing a reliable basis for evaluating our results.

Given the time and computational constraints of our project, we sampled 5,000 documents from each dataset, ensuring an equal distribution across all class labels to maintain balance. Since our approach relies on using LLMs without additional fine-tuning - aside from possibly including a few in-context examples - the primary reason for selecting this sample size was to increase confidence in the evaluation results.

To maintain a realistic experimental setting and minimize preprocessing overhead, we opted not to apply common text-cleaning steps such as removing HTML tags, stemming, or reducing punctuation. While such preprocessing might influence classification accuracy, we assumed that modern LLMs are robust enough to handle raw, noisy data effectively.

For our experiments, we chose three quantized versions of well-known LLMs, specifically selecting the "instruct" variants, which are optimized for following prompts and responding to commands efficiently. Although we initially planned to run multiple iterations per prompt and apply majority voting to refine classification accuracy, time constraints prevented us from implementing this strategy.

Additionally, due to the limited input capacity of the quantized models we used - restricted to 512 tokens per prompt - we were constrained to experimenting only with zero-shot and one-shot classification methods.

3.1 Prompts

3.1.1 Zero-Shot Classification

3.1.1.1 Simple Prompt

In our initial experiments, we tested zero-shot classification, where the prompt contained only a task description without providing any examples of how the classification should be performed. The first type of prompt we used was a simple prompt that directly asked the model to classify the given text into one of the predefined categories. This straightforward approach allowed us to evaluate the baseline performance of each model without additional guidance. However, since LLMs rely heavily on how a task is framed, we observed that results varied significantly depending on the exact wording of the prompt.

3.1.1.2 Reasoning-Based Prompt

To explore whether reasoning and explanation could enhance classification accuracy, we tested a second type of zero-shot prompt, inspired by the approach outlined by Sun et al. Instead of simply asking for a classification label, this prompt instructed the model to first provide clues and reasoning behind its decision before outputting the final classification. The goal was to leverage the model's ability to articulate its reasoning process, potentially leading to more accurate or interpretable results. This method aimed to mimic human-like deductive reasoning, allowing us to assess whether encouraging explicit justification could improve classification performance.

3.1.2 One-Shot Classfication

We then moved to one-shot classification, where we provided the model with a single labeled example alongside the task description. The rationale behind this approach was that an explicit demonstration of how the classification should be performed might improve the model's ability to generalize to unseen samples. Similar to our zero-shot setup, we tested two versions of the one-shot prompt. The first was a simple one-shot prompt, where a single labeled example was included before asking the model to classify the new text.

Instead of selecting a random example for the one-shot classification, we applied a more sophisticated method to choose the most relevant example from the dataset. Specifically, we used SentenceTransformers to generate sentence embeddings for each document in our dataset, then identified the closest example to the input text using cosine similarity. By selecting an example that was semantically similar to the given text, we aimed to provide a more informative reference, potentially increasing classification accuracy. This approach allowed us to test whether a strategically chosen demonstration could help the model better understand the task, rather than relying on a generic example that might not be representative of the specific text being classified.

A list of all the prompts, can be found in the appendix of this paper and in the source code.

3.2 Datasets

3.2.1 IMDB Reviews Dataset

The IMDB Reviews dataset is a widely used benchmark for sentiment analysis, consisting of movie reviews labeled as either positive or negative. Each review is a free-form text expressing the reviewer's opinion, making this dataset particularly useful for evaluating how well models handle subjective language, varying sentence structures, and differing levels of sentiment intensity. Since movie reviews often contain nuanced opinions, sarcasm, or mixed sentiments, this dataset provides a challenging yet realistic test for sentiment classification models.

3.2.2 Amazon Reviews Dataset

The Amazon Reviews dataset is another essential benchmark for sentiment analysis, containing product reviews from Amazon with labels indicating whether the sentiment is positive or negative. Unlike the IMDB dataset, which focuses on movie reviews, this dataset includes opinions on a diverse range of products, from electronics to clothing. This variation in subject matter introduces additional complexity, as product reviews may include not only emotional expressions but also factual statements about features, usability, and quality. The dataset serves as an excellent test for a model’s ability to generalize sentiment classification across different domains.

3.2.3 SMS Spam Collection Dataset

The SMS Spam Collection dataset is designed for spam detection, containing a mix of real-world text messages labeled as either "spam" or "ham" (non-spam). This dataset is particularly useful for evaluating how well models can differentiate between unsolicited promotional messages, phishing attempts, and legitimate personal or business communications. Unlike traditional text classification datasets, spam detection requires handling informal language, abbreviations, and varying text lengths, making it a valuable benchmark for assessing robustness in real-time filtering tasks.

3.2.4 R8 Dataset

The R8 dataset is a subset of the Reuters 21578 corpus, containing news articles categorized into eight of the most common topics: "earnings," "acquisitions," "money," "oil," "grain," "trade," "monetary," "wheat," and "shipping." Unlike sentiment or spam classification, topic classification requires models to understand context and domain-specific terminology. Since news articles often contain jargon and industry-specific phrases, this dataset challenges models to correctly identify the overarching subject matter of a text. Additionally, R8 is a multi-class classification dataset, meaning each document is assigned to one of multiple categories rather than just two. This makes it a useful benchmark for evaluating how well models handle more complex classification tasks beyond binary classification. Given its well-structured nature and clearly defined categories, R8 is frequently used to assess the effectiveness of text classification approaches in a controlled yet practical setting.

3.3 Models

We selected these models based on two key considerations. First, we aimed to include a diverse range of models from well-known organizations, ensuring representation from major AI research groups such as Alibaba (QWen), Meta (Llama 3.1 via Hermes 3), and Mistral AI (Mistral 7B). This allowed us to evaluate different architectural approaches and training methodologies across a variety of LLMs. Second, we conducted a series of preliminary experiments to assess the classification performance of multiple quantized models, and these three demonstrated the best balance between efficiency and accuracy. While each of these models has even smaller quantized versions, with precision levels as low as 4-bit or 5-bit, our early tests revealed that such extreme quantization significantly degraded classification performance, making them unsuitable for our study. Thus, we focused on the Q8 quantized versions, which provided a reasonable trade-off between computational efficiency and classification accuracy.

3.3.1 QWen 2.5 7B Instruct

QWen 2.5 7B Instruct is a 7.61-billion-parameter model developed by Alibaba, optimized for instruction-following and various natural language processing (NLP) tasks. It consists of 28 transformer layers and is part of the QWen 2.5 series, known for its multilingual capabilities and efficiency in text generation. For our experiments, we used the quantized Q8 version, which reduces the model’s precision to 8-bit integers, significantly decreasing memory usage while maintaining reasonable performance. This version of the model has a total size of 8.1 GB, making it more feasible to run on low-cost hardware compared to its full-precision counterpart.

3.3.2 Hermes 3 Llama 3.1 8B

Hermes 3 is a fine-tuned, quantized version of Meta’s Llama 3.1, an advanced open-weight large language model designed for improved reasoning, knowledge retention, and efficiency. Llama 3.1 builds upon its predecessor, Llama 3, incorporating enhancements in instruction tuning and overall response quality. Hermes 3 refines this further, tailoring the model to perform better in structured tasks such as text classification and dialogue-based interactions. We used the Q8 quantized version, which reduces computational requirements while preserving output fidelity. This version requires 8.54 GB of memory, making it a viable option for lower-resource setups while still benefiting from the strengths of Llama 3.1.

3.3.3 Mistral 7B Instruct

Mistral 7B Instruct is an instruction-tuned version of the Mistral 7B model, developed by Mistral AI, a relatively new AI research company focused on building lightweight, high-performance open-weight models. Unlike larger models from Alibaba or Meta, Mistral AI has positioned itself as a key player in efficient AI research, creating models that achieve strong performance despite their smaller size. The Mistral 7B architecture is known for its dense transformer structure, and the instruct variant is fine-tuned for better alignment with user queries. For our experiments, we used the Q8 quantized version, which compresses the model to require only 7.7 GB of memory. This makes it one of the most memory-efficient models among those tested while still maintaining competitive classification performance.

4. Experimental Setup

4.1 Setup

We conducted our experiments in two phases: zero-shot classification and one-zero-shot classification. For each dataset, we selected $5000/n$ examples per class, where n is the number of classes. However, in the r8 dataset, some classes had fewer than 625 (5000/8) examples available, resulting in a total of 3,912 examples.

Each example was processed using three different LLMs, with two prompting strategies: (1) Simple Prompt – A straightforward classification request. (2) Clues and Reasoning Prompt (CARP) – A prompt that asks the model to provide clues and reasoning before making a classification decision.

4.2 Hardware and Software Setup

We used llama_cpp, an open-source library for running LLMs with minimal setup, leveraging CUDA GPU acceleration (version 12.4). The tests were performed on the following hardware:

- Motherboard: Gigabyte B660 Gaming X AX DDR4
- CPU: Intel Core i7-13700K
- GPU: Nvidia GeForce RTX 3080 (12GB VRAM)
- RAM: 32GB DDR4 (2133 MT/s)
- Storage: Samsung 980 PRO 2TB SSD
- Power Supply: Corsair RM850x (850W)

The experiments were conducted on Windows 11 24H2 (build 26100.3194) with Nvidia GeForce Game Ready Driver version 572.60. This hardware represents a practical setup for end users. While the RTX 3080 was a high-end GPU at launch, it is now two generations old, making it an affordable choice with relatively large VRAM for its price.

4.3 Evaluation Metric

To assess model performance, we used $accuracy$ as our primary evaluation metric. Accuracy is defined as the proportion of correctly classified examples out of the total number of examples. Since our datasets were balanced, meaning each class had an approximately equal number of examples, accuracy serves as a reliable indicator of model effectiveness without being skewed by class imbalance.

Additionally, one of our datasets required multi-class classification, where each instance belonged to one of several mutually exclusive categories. In such cases, accuracy remains a straightforward and interpretable metric, as it directly reflects the model's ability to assign the correct label from multiple possible choices. Unlike precision, recall, or F1-score - which are particularly useful in imbalanced datasets - accuracy alone provides a clear, holistic measure of classification performance in this context.

By using accuracy, we aim to offer an intuitive and easily comparable performance measure across different models, prompting strategies, and datasets. However, we acknowledge that future work could benefit from exploring complementary metrics such as macro-averaged F1-score or log-likelihood-based confidence scores, which may provide additional insights into model behavior, especially in cases where misclassification patterns vary between classes.

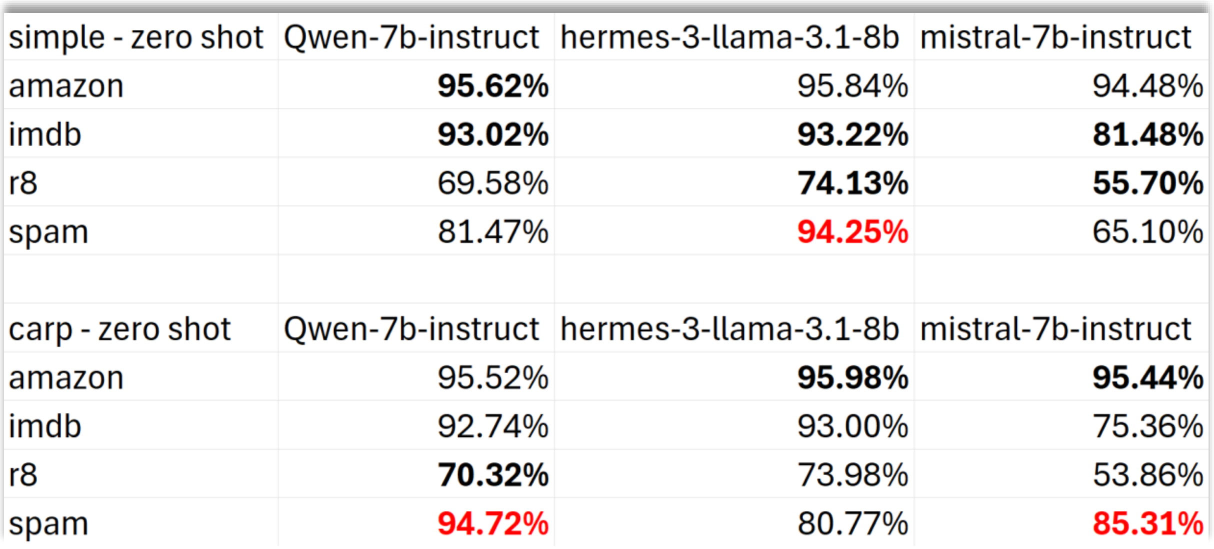
5. Results

The results are presented in two stages. First, we compare the simple prompt with CARP under the same conditions (either zero-shot or one-shot) to evaluate whether a more structured prompt improves performance. This helps determine whether explicitly refining the prompt leads to better model understanding and classification accuracy. After that, we compare zero-shot vs. one-shot performance within each prompting method (simple and CARP). This second comparison allows us to assess whether providing a single labeled example helps or hinders the model's ability to generalize the task. By structuring the results in this way, we can first analyze the impact of prompt design and then examine the effects of one-shot prompting independently.

5.1 Simple Prompt vs CARP

5.1.1 Zero-Shot

The results from the experiment, shown in Table 1, suggest that using Clues and Reasoning (CARP) did not yield consistent improvements over a simple zero-shot prompt across different datasets and models. While there were some cases of improvement, such as the increase in accuracy for the spam dataset using Mistral-7B (from 65.10% to 85.31%) and for R8 using Hermes-3-Llama-3.1-8B (from 74.13% to 73.98%), other cases showed minimal or even negative changes, such as the drop in IMDB accuracy for Mistral-7B (81.48% to 75.36%) and for R8 using Mistral-7B (55.70% to 53.86%). One possible explanation for this inconsistency is that CARP, while potentially beneficial for models with strong reasoning abilities, may introduce unnecessary complexity for smaller quantized models, leading to confusion rather than improved accuracy. Additionally, datasets with more straightforward classification tasks, such as Amazon and IMDB, may not benefit



simple - zero shot	Qwen-7b-instruct	hermes-3-llama-3.1-8b	mistral-7b-instruct
amazon	**95.62%**	95.84%	94.48%
imdb	**93.02%**	**93.22%**	**81.48%**
r8	69.58%	**74.13%**	**55.70%**
spam	81.47%	**94.25%**	65.10%
carp - zero shot	Qwen-7b-instruct	hermes-3-llama-3.1-8b	mistral-7b-instruct
amazon	95.52%	**95.98%**	**95.44%**
imdb	92.74%	93.00%	75.36%
r8	**70.32%**	73.98%	53.86%
spam	**94.72%**	80.77%	**85.31%**

Table 1 - Accuracy results: Zero-Shot: Simple vs CARP

significantly from additional reasoning steps since the sentiment is often explicitly clear. Meanwhile, more complex datasets like spam detection and R8 might show variable results depending on how well the model processes and applies the provided reasoning. Finally, computational constraints on lower-resource models might hinder their ability to effectively leverage CARP, making its impact less predictable.

5.1.2 One Shot

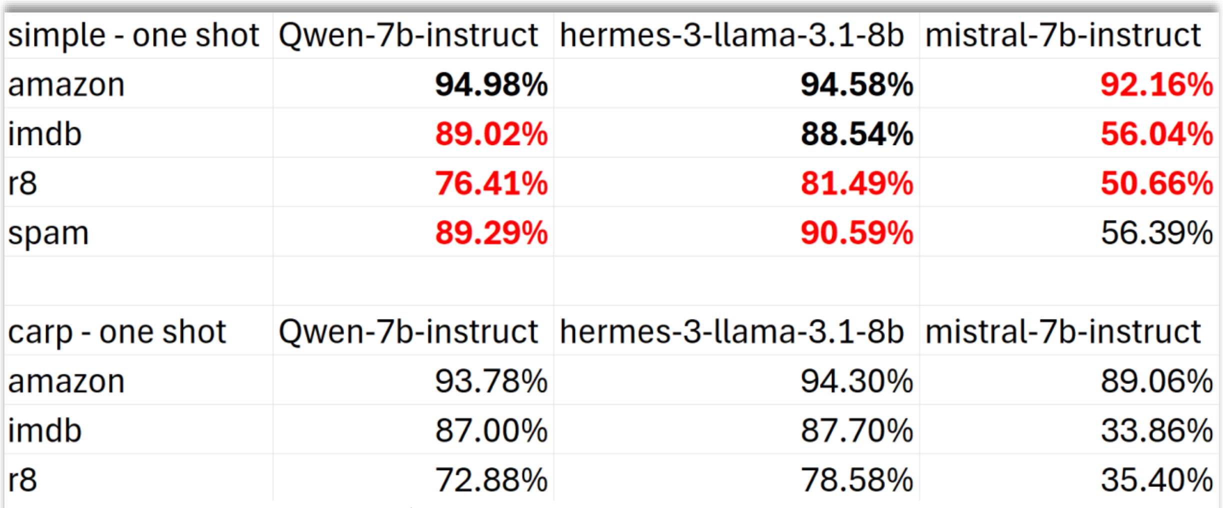
In the one-shot setting, shown in Table 2, the introduction of Clues and Reasoning (CARP) continued to show inconsistent effects, with a noticeable decline in accuracy across most datasets and models compared to the simple prompt. Unlike the zero-shot scenario, where some datasets benefited from CARP, here, performance degraded in almost all cases. Notably, Mistral-7B saw significant drops, especially in IMDB (56.04% to 33.86%) and R8 (50.66% to 35.40%). Even models that previously handled CARP well, such as Hermes-3-Llama-3.1-8B, experienced declines in datasets like spam (90.59% to 84.84%). A likely explanation for this is that adding an example before prompting, combined with the already complex reasoning structure of CARP, may have overwhelmed the models. The increased prompt length could have led to context dilution or misinterpretation, especially for quantized models with limited memory and reasoning capabilities. Additionally, one-shot examples may have constrained the model's flexibility in generalizing across different cases, making it harder for the models to apply learned patterns effectively.

5.2 One-Shot

5.2.1 Simple Prompt

When comparing the zero-shot and one-shot performances using a simple prompt (**שגיאה! מקור ההפניה לא נמצא.**), we can see that adding a single example before prompting had mixed effects depending on the dataset and model. In the Amazon dataset, performance remained nearly the same across all models, with only slight fluctuations. On the other hand, in the IMDB dataset, all models experienced a drop in accuracy when switching to one-shot, with Mistral-7B being the most affected (81.48% to 56.04%). A similar trend was observed in the R8 dataset, where one-shot prompting improved Hermes-3 slightly (74.13% to 81.49%) but caused declines in Qwen-7B (76.41% to 69.58%) and Mistral-7B (55.70% to 50.66%). In the Spam dataset the effects varied across models. Qwen-7B and Hermes-3 maintained relatively stable performance, while Mistral-7B saw a sharp decline (65.10% to 56.39%).

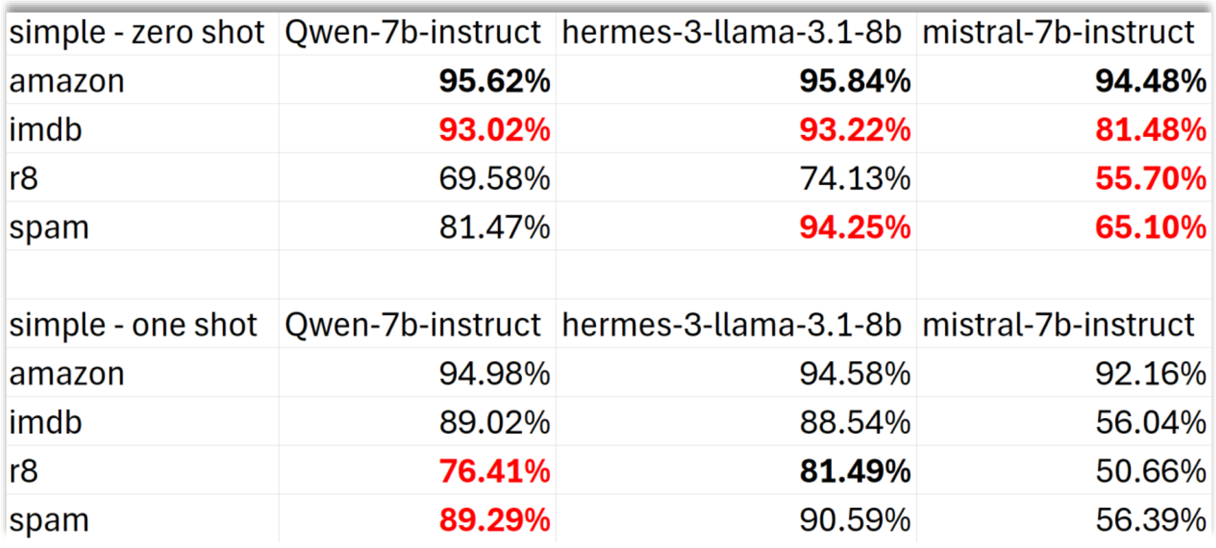
Overall, while the one-shot approach slightly helped in isolated cases, it mostly led to decreased accuracy. This suggests that, for



simple - one shot	Qwen-7b-instruct	hermes-3-llama-3.1-8b	mistral-7b-instruct
amazon	**94.98%**	**94.58%**	**92.16%**
imdb	**89.02%**	**88.54%**	**56.04%**
r8	**76.41%**	**81.49%**	**50.66%**
spam	**89.29%**	**90.59%**	56.39%
carp - one shot	Qwen-7b-instruct	hermes-3-llama-3.1-8b	mistral-7b-instruct
amazon	93.78%	94.30%	89.06%
imdb	87.00%	87.70%	33.86%
r8	72.88%	78.58%	35.40%

Table 2 - Accuracy results: One-Shot: Simple vs CARP

these models, adding a single example does not always provide a clearer learning signal and may instead introduce unnecessary complexity.



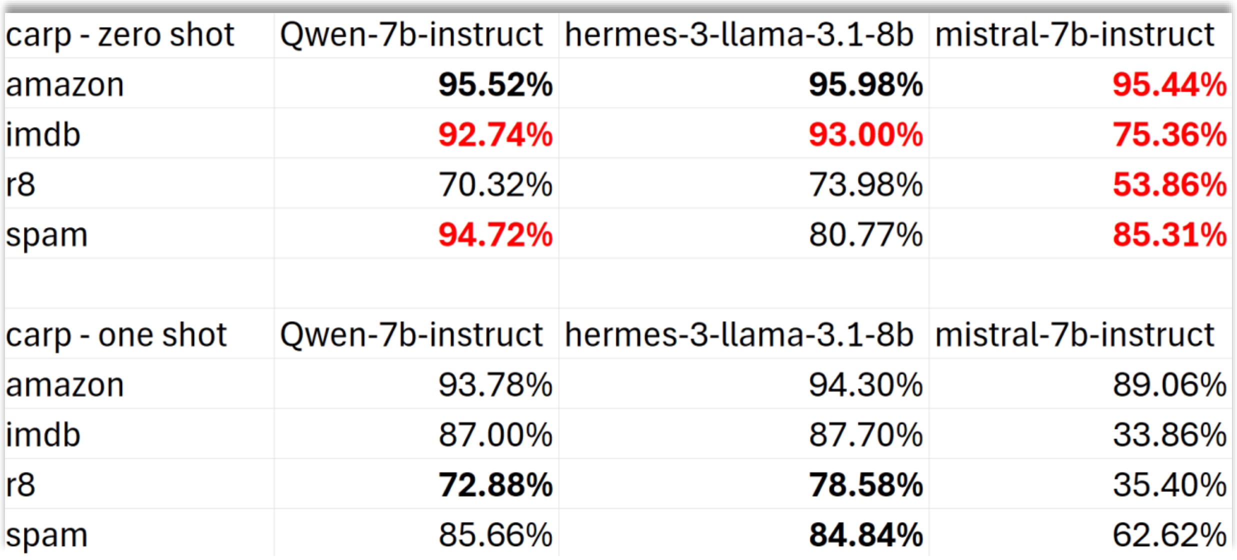
simple - zero shot	Qwen-7b-instruct	hermes-3-llama-3.1-8b	mistral-7b-instruct
amazon	**95.62%**	**95.84%**	**94.48%**
imdb	**93.02%**	**93.22%**	**81.48%**
r8	69.58%	74.13%	**55.70%**
spam	81.47%	**94.25%**	**65.10%**
simple - one shot	Qwen-7b-instruct	hermes-3-llama-3.1-8b	mistral-7b-instruct
amazon	94.98%	94.58%	92.16%
imdb	89.02%	88.54%	56.04%
r8	**76.41%**	**81.49%**	50.66%
spam	**89.29%**	90.59%	56.39%

Table 3 - Accuracy results: Simple Prompt: Zero-Shot vs One-Shot

5.2.2 CARP

When comparing the zero-shot and one-shot performances using CARP, we observe a consistent decline in accuracy across all models and datasets when switching to one-shot prompting. In the Amazon dataset: A slight decrease for all models. Qwen-7B dropped from 95.52% to 93.78%, Hermes-3 from 95.98% to 94.30%, and Mistral-7B from 95.44% to 89.06%. On the IMDB dataset we can see a more noticeable decline, especially for Mistral-7B, which fell sharply from 75.36% to 33.86%. Performance dropped across the board in the R8 dataset, with Mistral-7B being particularly affected (53.86% to 35.40%). In the Spam dataset, Qwen-7B and Hermes-3 maintained similar performance, but Mistral-7B suffered a sharp drop (85.31% to 62.62%).

Unlike with the simple prompt, the one-shot setting consistently hurt performance in CARP.



carp - zero shot	Qwen-7b-instruct	hermes-3-llama-3.1-8b	mistral-7b-instruct
amazon	**95.52%**	**95.98%**	**95.44%**
imdb	**92.74%**	**93.00%**	**75.36%**
r8	70.32%	73.98%	**53.86%**
spam	**94.72%**	80.77%	**85.31%**
carp - one shot	Qwen-7b-instruct	hermes-3-llama-3.1-8b	mistral-7b-instruct
amazon	93.78%	94.30%	89.06%
imdb	87.00%	87.70%	33.86%
r8	**72.88%**	**78.58%**	35.40%
spam	85.66%	**84.84%**	62.62%

Table 4 - Accuracy results: CARP: Zero-Shot vs One-Shot

The drop was more severe in Mistral-7B, particularly for IMDB, R8, and Spam. This suggests that CARP might be more sensitive to additional examples, possibly due to the complexity of the prompt structure or the increased length of the input. This aligns with the earlier trend we saw, where one-shot prompting did not necessarily improve performance and, in many cases, caused confusion or degraded accuracy.

5.3 Summary

The evaluation compared the performance of three models (Qwen-7B-Instruct, Hermes-3-Llama-3.1-8B, and Mistral-7B-Instruct) across four datasets (Amazon, IMDB, R8, and Spam) using both a simple and CARP prompting strategy. Additionally, each prompt style was tested under zero-shot and one-shot settings. The results indicate that one-shot prompting consistently led to a decline in accuracy across all models and datasets, with the drop being particularly severe when using CARP. Among the models, Mistral-7B exhibited the most significant performance degradation in the one-shot setting, especially on IMDB, R8, and Spam datasets. However, despite these fluctuations, most of the results remain quite strong, particularly considering that they were obtained without any fine-tuning or prior training, using only commodity hardware.

6. Conclusions

These findings suggest that one-shot prompting does not necessarily improve sentiment classification performance and, in many cases, can even degrade accuracy. This could be due to increased input complexity, where models struggle to generalize effectively when presented with an additional example. The impact was especially pronounced for Mistral-7B, implying that certain models may be more sensitive to input length and structure than others.

That said, the overall performance is still remarkably high, demonstrating that off-the-shelf LLMs running on commodity hardware can achieve strong results even without fine-tuning. This reinforces the viability of LLMs for real-world sentiment analysis tasks in low-budget scenarios. For future work, it would be beneficial to explore alternative few-shot strategies, such as using multiple examples, adjusting formatting, or testing different types of demonstrations. Additionally, truncating or simplifying prompts may help mitigate the negative effects of longer inputs. Ultimately, these results highlight the importance of empirical testing when selecting prompting strategies for LLM-based sentiment analysis, as performance can vary significantly depending on the model, dataset, and prompt design.

Given the results in this paper, future research may need to explore the impact of few-shot prompting (providing multiple examples instead of just one) and chain-of-thought prompting, where the model is guided through a reasoning process before making a classification decision. Additionally, testing how larger or more fine-tuned models perform under similar prompting conditions could provide insight into whether these trends hold across different architectures and scales. Another valuable direction would be optimizing prompts for efficiency on low-budget hardware, as well as investigating whether adaptive prompting strategies—where the model dynamically selects the best prompt based on its initial responses—can lead to improved performance without increasing computational costs.

7. Bibliography

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., . . . Askell, A. (2020). Language models are few-shot learners. *Advances in neural information processing systems, 33*, 1877--1901.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., . . . Gehrmann, S. (2023). Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research, 24*(240), 1--113.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2009). Bert: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 conference of the North American chapter of the association* *for computational linguistics: human language technologies, volume 1 (long and short papers)*, (pp. 4171--4186).

Frantar, E., Ashkboos, S., Hoefler, T., & Alistarh, D. (2022). Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*.

Han, S., Mao, H., & Dally, W. J. (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation, 8*(9), 1735--1780.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. *European conference on machine learning*, 137--142.

Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., . . . Rocktäschel, T. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems, 33*, 9459--9474.

Lin, J., Tang, J., Tang, H., Yang, S., Chen, W.-M., Wang, W.-C., . . . Han, S. (2024). Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of Machine Learning and Systems, 6*, 87--100.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., . . . Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training.

Robertson, S. (2004). Understanding inverse document frequency: on theoretical arguments for IDF. *Journal of documentation, 60*(5), 503--520.

Shen, S., Dong, Z., Ye, J., Ma, L., Yao, Z., Gholami, A., . . . Keutzer, K. (2020). Q-bert: Hessian based ultra low precision quantization of bert. *34*(05), 8815--8821.

Sun, X., Li, X., Li, J., Wu, F., Guo, S., Zhang, T., & Wang, G. (2023). Text classification via large language models. *arXiv preprint arXiv:2305.08377*.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., . . . Azhar, F. (2023). Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., & Kaiser, Ł. (2017). Attention is all you need. *Advances in neural information processing systems, 30*.

Appendix A – Prompt Templates

```
SPAM_EXAMPLE = """For example:

Text: "{text}"


SPAM or HAM: {label}

"""


SPAM_SIMPLE_PROMPT_TEMPLATE = """This is an overall spam classifier for input emails and messages.

Categorize the input as a "spam" or "ham".

{examples}

Text: "{dataset_text}"


SPAM or HAM:"""


SPAM_PROMPT_TEMPLATE = """This is an overall spam classifier for input emails and messages.

Present CLUES (i.e., keywords, phrases, contextual information, semantic meaning, semantic relationships, tones, references) that support the spam or ham determination of the input.

Next, deduce the diagnostic REASONING process from premises (i.e., clues, input) that support the sentiment determination.

Finally, based on clues, reasoning and the input, categorize the overall SENTIMENT of input as a "spam" or "ham".

Limit your answer 100 tokens or less.

{examples}

Text: "{dataset_text}"


SPAM or HAM:"""


IMDB_EXAMPLE = """For example:

Text: "{text}"
```

```
SENTIMENT: {label}
"""


IMDB_SIMPLE_PROMPT_TEMPLATE = """This is an overall sentiment classifier for movie
reviews. Classify the overall SENTIMENT of the INPUT as Positive or

Negative.

{examples}

Text: "{dataset_text}"


SENTIMENT:"""


IMDB_PROMPT_TEMPLATE = """This is an overall sentiment classifier for movie reviews

Present CLUES (i.e., keywords, phrases, contextual information, semantic meaning, semantic
relationships, tones, references) that support the spam or ham determination of the input.

Next, deduce the diagnostic REASONING process from premises (i.e., clues, input) that support the
sentiment determination.

Finally, based on clues, reasoning and the input, categorize the overall SENTIMENT of input as a
"negative" or "positive".

Limit your answer 100 tokens or less.

{examples}

Text: "{dataset_text}"


SENTIMENT:"""


AMAZON_EXAMPLE = """For example:

Text: {text}


SENTIMENT: {label}
"""



AMAZON_SIMPLE_PROMPT_TEMPLATE = """This is an overall sentiment classifier for amazon
products reviews. Classify the overall SENTIMENT of the INPUT as Positive or

Negative.
```

```
{examples}

Text: "{dataset_text}"

SENTIMENT:"""


AMAZON_PROMPT_TEMPLATE = """This is an overall sentiment classifier for amazon products reviews

Present CLUES (i.e., keywords, phrases, contextual information, semantic meaning, semantic relationships, tones, references) that support the spam or ham determination of the input.

Next, deduce the diagnostic REASONING process from premises (i.e., clues, input) that support the sentiment determination.

Finally, based on clues, reasoning and the input, categorize the overall SENTIMENT of input as a "negative" or "positive".

Limit your answer 100 tokens or less.

{examples}

Text: "{dataset_text}"


SENTIMENT:"""




R8_EXAMPLE = """For example:

Text: {text}


TOPIC: {label}

"""




R8_SIMPLE_PROMPT_TEMPLATE = """This is an overall text classifier for news article topics.

Categorize topic of the article to one of the following topics: "earnings", "acquisitions", "money", "oil", "grain", "trade", "monetary", "wheat", "shipping".

{examples}

Text: "{dataset_text}"


TOPIC:"""
```

```
R8_PROMPT_TEMPLATE = """This is an overall text classifier for news article topics.

Present CLUES (i.e., keywords, phrases, contextual information, semantic meaning, semantic
relationships, tones, references) that support the topic classfication of the input.

Next, deduce the diagnostic REASONING process from premises (i.e., clues, input) that support the
sentiment determination.

Finally, based on clues, reasoning and the input, categorize topic of the article to one of the following
topics: "earnings", "acquisitions", "money", "oil", "grain", "trade", "monetary", "wheat", "shipping".

Limit your answer 100 tokens or less.

{examples}

Text: "{dataset_text}"


TOPIC:"""
```