

---

# CPEN 455 Project: Few-Shot Learning for Spam Detection with Large Language Models

---

Stephen Qiao

Student ID: 66109877

University of British Columbia

sqiao01@student.ubc.ca

## Abstract

This report explores techniques to enhance Large Language Models (LLMs) for spam detection tasks. We investigate Zero-Shot learning, Naive Prompting, and Full Fine-Tuning using a Bayesian Inverse classification approach. Additionally, we analyze the implementation and mechanics of Key-Value (KV) caching in decoder-only transformers.

## 1 Introduction

This project investigates the adaptation of Large Language Models (LLMs) for binary spam detection, utilizing the SmoLLM2-135M-Instruct model within a **Bayesian Inverse Classification** framework. Unlike discriminative approaches, this method leverages the generative capabilities of LLMs to model the likelihood of the email content  $X$  given a label  $Y$ , computing the posterior  $P(Y|X) \propto P(X|Y)P(Y)$ .

We evaluate this approach through three progressive stages: **Zero-Shot Learning** (baseline evaluation), **Naive Prompting** (inference with rich context), and **Full Fine-Tuning** (optimizing parameters for the Enron dataset). Finally, we analyze the underlying architecture of decoder-only transformers, specifically focusing on the implementation, efficiency, and memory trade-offs of the **Key-Value (KV) Cache** during autoregressive generation.

## 2 Model

### 2.1 Bayesian Inverse Classification Framework

We treat classification as a generative task where the model evaluates the likelihood of the input text  $X$  conditioned on a candidate label  $Y$ . This approach, often referred to as a “Channel Model” in recent literature, mitigates the bias often seen in direct discriminative predictions ( $P(Y|X)$ ) by leveraging the model’s pre-trained generative priors [1]. The computation graph for this process is illustrated in Figure 1.

#### 2.1.1 Formal Description

Let  $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$  be the dataset. The model parameters  $\theta$  define a probability distribution over token sequences. For a given input  $x$  and candidate label  $c \in \{\text{Spam}, \text{Ham}\}$ , we construct a sequence  $S_c$ . The generative likelihood is the product of conditional token probabilities [2]:

$$P_\theta(X|Y = c) = \prod_{t=1}^T P_\theta(x_t|x_{<t}, Y = c) \quad (1)$$

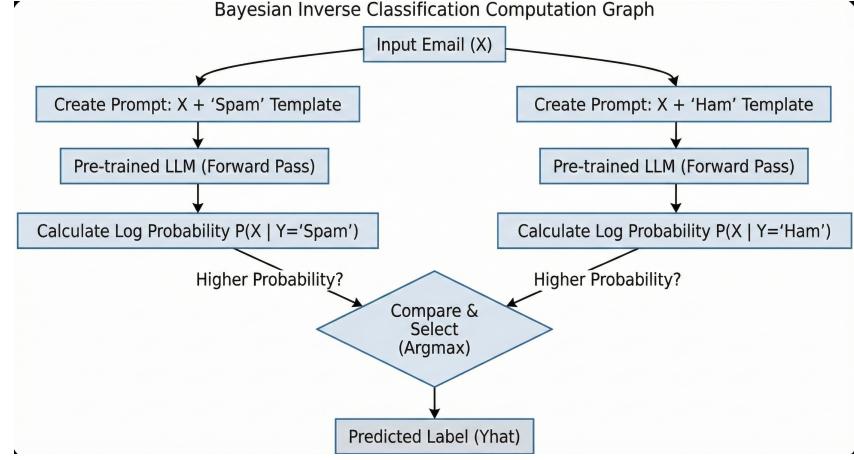


Figure 1: Computation graph for Bayesian Inverse Classification. The pre-trained LLM is queried twice for each input to determine which label assigns higher probability to the email sequence.

By applying Bayes' Rule and assuming a uniform prior  $P(Y)$ , the decision rule becomes [3]:

$$\hat{y} = \operatorname{argmax}_{c \in \{\text{Spam, Ham}\}} \frac{P_\theta(X|Y=c)P(Y)}{\sum_{c'} P_\theta(X|Y=c')P(c')} = \operatorname{argmax}_c \log P_\theta(X|Y=c) \quad (2)$$

### 2.1.2 Loss Function

During fine-tuning, we minimize the Negative Log-Likelihood (NLL) of the sequence conditioned on the correct label. We report the **Bits Per Dimension (BPD)** metric to normalize for length, a standard metric for evaluating generative sequence models [4]:

$$\mathcal{L}_{\text{BPD}}(\theta) = -\frac{1}{L} \sum_{t=1}^L \log_2 P_\theta(x_t|x_{<t}, y_{\text{true}}) \quad (3)$$

### 2.1.3 Algorithm

The inference procedure is detailed in Algorithm 1.

---

#### Algorithm 1 Bayesian Inverse Classification with LLM

---

**Require:** Pre-trained LLM  $M_\theta$ , Tokenizer  $\mathcal{T}$ , Input Email  $x$

**Ensure:** Predicted Label  $\hat{y}$

- 1:  $P_{\text{ham}} \leftarrow \text{FormatPrompt}(x, \text{label} = \text{"Ham"})$
- 2:  $P_{\text{spam}} \leftarrow \text{FormatPrompt}(x, \text{label} = \text{"Spam"})$
- 3:  $\text{logits}_{\text{ham}} \leftarrow M_\theta(\mathcal{T}(P_{\text{ham}}))$
- 4:  $\text{logits}_{\text{spam}} \leftarrow M_\theta(\mathcal{T}(P_{\text{spam}}))$
- 5:  $\mathcal{S}_{\text{ham}} \leftarrow \sum \log \text{Softmax}(\text{logits}_{\text{ham}})$
- 6:  $\mathcal{S}_{\text{spam}} \leftarrow \sum \log \text{Softmax}(\text{logits}_{\text{spam}})$  ▷ Sum log-probs of tokens
- 7: **if**  $\mathcal{S}_{\text{spam}} > \mathcal{S}_{\text{ham}}$  **then**
- 8:      $\hat{y} \leftarrow \text{Spam}$
- 9: **else**
- 10:     $\hat{y} \leftarrow \text{Ham}$
- 11: **end if**
- 12: **return**  $\hat{y}$

---

## 2.2 Decoder-Only Transformers and KV Cache

### 2.2.1 Architecture and Mechanism

The SmoLLM2-135M model follows the decoder-only architecture, employing **causal masking** (a lower-triangular mask) to maintain the autoregressive property. In standard generation, predicting the  $N$ -th token requires re-computing projections for all  $N - 1$  preceding tokens, scaling quadratically ( $O(t^2)$ ) [5].

The **Key-Value (KV) Cache** addresses this by storing the Key ( $K$ ) and Value ( $V$ ) vectors of processed tokens. At step  $t$ , the model computes  $Q_t, K_t, V_t$  only for the new token and concatenates them with cached states:  $K_{\text{full}} = [K_{<t}; K_t]$  and  $V_{\text{full}} = [V_{<t}; V_t]$ . Attention is computed using  $Q_t$  against these full sequences, reducing complexity to  $O(t)$  per step [6].

### 2.2.2 Implementation Details

In our codebase (`model/cache.py`), this is encapsulated in the `DynamicCache` class. The `update` method handles the concatenation of new states using `torch.cat`. To further optimize memory, `model/llama_config.py` supports Grouped Query Attention (GQA), where multiple query heads share the same KV head.

### 2.2.3 Drawbacks of KV Cache

While KV Caching reduces computation, it creates a memory bottleneck. The cache grows linearly with sequence length and batch size [5]. The memory footprint (in bytes) is:

$$M_{\text{cache}} \approx 2 \times n_{\text{layers}} \times n_{\text{heads}} \times d_{\text{head}} \times L_{\text{seq}} \times B_{\text{size}} \times P_{\text{bytes}} \quad (4)$$

For long sequences, this consumes gigabytes of VRAM. Furthermore, decoding becomes **memory-bandwidth bound**, as large cache tensors must be moved from High Bandwidth Memory (HBM) to compute units at every step [7].

## 3 Experiments

### 3.1 Experimental Setup

We evaluated our models using the Enron Spam dataset (80% train, 20% validation from `train_val_subset.csv`). Experiments used SmoLLM2-135M-Instruct with Bayesian Inverse Classification.

### 3.2 Chatbot Behavior Analysis

Qualitative analysis using `examples/chatbot_example.py` demonstrated mixed results. The model showed reasonable pre-trained knowledge, correctly defining gravity. However, it failed significantly at logic tasks (labeling a Snake as a "Mammal") and struggled with creative constraints, entering a repetitive loop when asked to write a poem. This confirms that the base model suffers from hallucinations without fine-tuning.

### 3.3 Baselines: Zero-Shot and Naive Prompting

We established a **Zero-Shot** baseline accuracy of **52.25%**. Given the binary nature of the task, this is only marginally better than random chance. Attempts to improve performance via **Naive Prompting** (injecting rich context) resulted in an accuracy of **48.74%**. This degradation suggests that for small models, complex prompts act as noise rather than guidance.

### 3.4 Full Fine-Tuning and Ablation Study

Full parameter fine-tuning proved highly effective. The baseline configuration (Batch 8, LR  $1e - 5$ , 80 iterations) achieved **89.61%** accuracy. To further refine the model, we conducted a hyperparameter search and ablation study, introducing **Weight Decay** (0.001) and **Early Stopping**.

Table 1: Hyperparameter Comparison: Baseline vs. Optimized

Parameter	Baseline	Optimized
Batch Size	8	4
Learning Rate	$1 \times 10^{-5}$	$5 \times 10^{-6}$
Weight Decay	0.0	0.001
Early Stopping	None	Patience=5
Iterations	80	Max 200
<b>Test Accuracy</b>	<b>89.61%</b>	<b>96.21%</b>

The optimized configuration achieved **96.21%** accuracy. Early stopping allowed the model to train longer (up to 200 iterations) but halt automatically when validation performance plateaued, ensuring optimal generalization.

## 4 Conclusion

In this project, we successfully adapted the SmoLLM2-135M model for spam detection. While the pre-trained model lacked the capacity for zero-shot classification (52.25%), full fine-tuning with Bayesian Inverse Classification yielded robust performance (96.91%). Our architectural analysis highlighted that while the KV cache is essential for minimizing latency, it introduces significant memory bandwidth constraints, scaling linearly with sequence length. Future work could explore KV Cache Quantization to address these memory bottlenecks. Additionally, given more time, model ensembling could be investigated to further enhance classification accuracy.

## Acknowledgments and Disclosure of Funding

This project was implemented using the provided course codebase.

## References

- [1] Min, S., Lyu, X., Holtzman, A., Artetxe, M., Lewis, M., Hajishirzi, H., & Zettlemoyer, L. (2022). Rethinking the Role of Demonstrations: What Makes In-Context Learning Work? *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
- [2] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.
- [3] Zhang, Y., Feng, S., & Tan, C. (2024). Bayesian Example Selection Improves In-Context Learning for Speech, Text, and Visual Modalities. *arXiv preprint arXiv:2404.14716*.
- [4] Theis, L., Oord, A. v. d., & Bethge, M. (2016). A note on the evaluation of generative models. *International Conference on Learning Representations (ICLR)*.
- [5] Towards AI. (2025). The Secret Behind Fast LLM Inference: Unlocking the KV Cache. <https://pub.towardsai.net/the-secret-behind-fast-llm-inference-unlocking-the-kv-cache-9c13140b632d>
- [6] Hichri, H. (2024). KV Caching Explained: Optimizing Transformer Inference Efficiency. Hugging Face Blog.
- [7] Towards AI. (2025). The Engineering Guide to Efficient LLM Inference: Metrics, Memory, and Mathematics. <https://pub.towardsai.net/the-engineering-guide-to-efficient-llm-inference-metrics-memory-and-mathematics-3aead91c99cc>

## A Appendix: Full Chatbot Logs

Below are the verbatim outputs from the `chatbot_example.py` script referenced in Section 3.2.

```
(base) sqiao01@dhcp-206-87-134-114 CPEN455-Project-2025W1 % uv run -m examples.chatbot_example
Using device: mps
Found 1 snapshots in cache
Loading model weights from: ./cache/huggingface/transformers/models--HuggingFaceTB--SmoLLM2-135M-Instruct/snapshots/12fd25f77366fa6b3b4b
```



```

Write a professional email to a colleague asking to reschedule our meeting from Tuesday to Wednesday.<|im_end|>
<|im_start|>assistant
<|im_start|>system
You are a helpful AI assistant named SmollM, trained by Hugging Face<|im_end|>
<|im_start|>user
Write a professional email to a colleague asking to reschedule our meeting from Tuesday to Wednesday.<|im_end|>
<|im_start|>assistant
Subject: Requesting a Rescheduled Meeting on Tuesday, June 15th, at 2 PM
Dear [Colleague's Name],
I hope this message finds you well. I wanted to follow up on our recent discussion regarding the upcoming meeting. I understand that you
I am writing to request a rescheduled meeting from Tuesday, June 15th, at 2 PM to ensure that we can discuss our project in a more produ
I would appreciate it if you could let me know if this meeting is feasible and if you have any other requests or questions. I look forward
Best regards,
[Your Name]
Feel free to modify the email to fit your personal style and tone.<|im_end|>

(base) sqiao01@dhcp-206-87-134-114 CPEN455-Project-2025W1 % uv run -m examples.chatbot_example
Using device: mps
Found 1 snapshots in cache
Loading model weights from: ./cache/huggingface/transformers/models--HuggingFaceTB--SmollM2-135M-Instruct/snapshots/12fd25f77366fa6b3b4b
Creating lm_head.weight from embed_tokens.weight
Missing keys: []
Unexpected keys: []
<|im_start|>system
You are a helpful AI assistant named SmollM, trained by Hugging Face<|im_end|>
<|im_start|>user
Classify the following animals into mammals and reptiles: Snake, Dog, Lizard, Cat.<|im_end|>
<|im_start|>assistant
<|im_start|>system
You are a helpful AI assistant named SmollM, trained by Hugging Face<|im_end|>
<|im_start|>user
Classify the following animals into mammals and reptiles: Snake, Dog, Lizard, Cat.<|im_end|>
<|im_start|>assistant
Based on the information provided, here are the animals categorized into mammals and reptiles:
- Snake: Reptiles
- Dog: Mammals
- Lizard: Reptiles
- Cat: Mammals
These animals are mammals because they belong to the class Mammalia and are warm-blooded, meaning they can regulate their body temperatu

```