Email Routing with Large Language Models

Automating Customer Support Ticket Classification
with GPT-2 and DistilBERT

Jacopo Parretti – VR 536104
Cesare Fraccaroli – VR 533061
Stefano Giampaoli – VR 535385

Natural Language Processing 2025–26
University of Verona

1 Introduction

Customer support teams handle thousands of emails daily. Routing each email to the correct department is repetitive yet critical—misrouted tickets cause delays and customer frustration. In this project we explore whether Large Language Models can automatically classify customer support emails into one of five departments: *Technical Support*, *Customer Service*, *Billing and Payments*, *Sales and Pre-Sales*, and *General Inquiry*.

We design and compare three progressively stronger agents:

1. **Agent 1 – Frozen prompting:** zero-shot log-likelihood scoring with GPT-2 and DistilGPT-2 (no weight updates).
2. **Agent 2 – LoRA fine-tuning:** parameter-efficient adaptation of GPT-2 and DistilGPT-2 ($\sim$1% trainable parameters).
3. **Agent 3 – Discriminative classifier:** fully fine-tuned DistilBERT for sequence classification.

2 Dataset

We use the `Tobi-Bueck/customer-support-tickets` dataset from Hugging Face, filtered to English emails across the five target departments. After filtering, the data is shuffled (seed = 42) and split into training (80 %), validation (10 %), and test (10 %) sets, yielding 13 249 / 1 656 / 1 657 samples respectively.

The class distribution is heavily imbalanced: *Technical Support* accounts for $\sim$49% of training samples, while *General Inquiry* represents only $\sim$2.5%. This imbalance is consistent across all splits and affects model performance on minority classes.

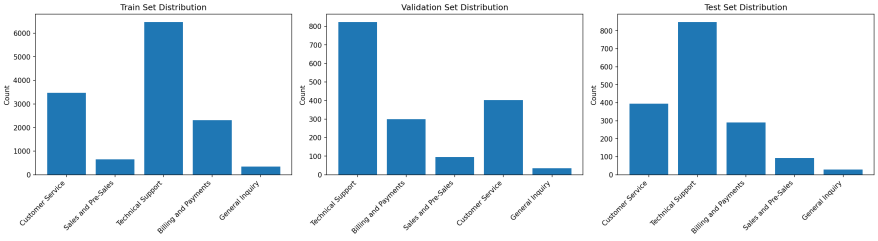

Figure 1: Label distribution across train, validation, and test sets.

3 Methods

3.1 Agent 1: Frozen Prompting (GPT-2 / DistilGPT-2)

What "frozen" means. A *frozen* model is one whose weights are never updated: we load the pretrained checkpoint and use it as-is, with no training step at all. The model must rely entirely on the general language knowledge it acquired during its original pretraining on large internet corpora. This is the simplest possible approach—no dataset, no GPU training time—but it also means the model has no task-specific understanding of email routing.

How it works. GPT-2 is an autoregressive (left-to-right) language model: given a sequence of tokens (sub-word units that the text is split into), it predicts the probability of the next token. We exploit this by constructing an instruction-style prompt that ends with `"Department:"` and then measuring how likely each of the five department names is as the next-token continuation. Concretely, for each candidate label ℓ we compute:

$$\text{score}(\ell) \;=\; \frac{1}{|\ell|} \sum_{t=1}^{|\ell|} \log P\big(\ell_t \mid \text{prompt},\, \ell_{<t}\big)$$

where $|\ell|$ is the number of tokens in the label, P is the model's softmax distribution (a function that converts raw model outputs into probabilities summing to 1), and we divide by $|\ell|$ to normalize for label length—otherwise labels with fewer tokens would be unfairly favoured. The department with the highest score is selected.

How it is applied. Each email's subject and body (truncated to 500 characters to fit GPT-2's 1 024-token context window) are inserted into the prompt template. We run this procedure with two models: GPT-2 (124 M parameters, 12 transformer layers) and DistilGPT-2 (82 M parameters, 6 layers—a *distilled* variant, meaning a smaller model trained to mimic the larger one, trading some capacity for faster inference). A **KV-cache** strategy avoids redundant computation: during a forward pass each transformer layer produces key and value vectors that encode the processed context; we cache these after the prompt is forwarded once, and then only each candidate label's tokens need to be forwarded incrementally, saving significant time.

We also perform a **prompt sensitivity analysis** with four prompt variants (instruction + department list, minimal, few-shot with one example per class, and role-based) to quantify how prompt wording affects classification accuracy when no fine-tuning is applied.

3.2 Agent 2: LoRA Fine-Tuning (GPT-2 / DistilGPT-2)

Why fine-tune, and what LoRA is. Agent 1 showed that a frozen model lacks the task-specific knowledge needed for email routing. Fine-tuning adapts the model's weights on our labeled dataset so it *learns* the mapping from emails to departments. However, updating all parameters of GPT-2 (124 M weights) is expensive in memory and risks overfitting on a moderately sized dataset. Low-Rank Adaptation (LoRA) solves this by *freezing* the original weight matrices and injecting small, trainable low-rank matrices beside them. Instead of learning a full update $\Delta W \in \mathbb{R}^{d\times d}$, LoRA decomposes it as $\Delta W = A \cdot B$ where $A \in \mathbb{R}^{d\times r}$ and $B \in \mathbb{R}^{r\times d}$ with rank $r \ll d$. Only A and B are trained, drastically reducing the number of trainable parameters.

How it works. We apply LoRA with rank r=16 and scaling factor α=32 to the attention projection layers (`c_attn` and `c_proj`—the matrices that compute queries, keys, values and the output projection inside each self-attention sub-layer) of every transformer block. The rank r controls the

size of the injected matrices: a larger r gives more expressive power but increases trainable parameters, while α is a scaling constant that modulates how strongly the low-rank update influences the original weights (effectively ΔW is scaled by α/r). With r=16 this yields only 1.62 M trainable parameters for GPT-2 (1.29% of 126 M total) and 0.81 M for DistilGPT-2 (0.98% of 83 M total). Training uses **causal language modeling loss** (the standard next-token-prediction objective): each training sample is the concatenation `prompt + " " + label`, and the loss is computed only on the label tokens. The prompt tokens are masked with -100, a special value that tells PyTorch's cross-entropy to ignore those positions. This teaches the model to produce the correct department name as a high-probability continuation of the prompt.

How it is applied. We train for 5 **epochs** (full passes through the training set) using **AdamW**, an optimizer that combines adaptive per-parameter learning rates with weight decay for regularization. The **learning rate** 3×10^{-4} controls the step size of each weight update, and the **batch size** of 4 means four email–label pairs are processed together before each gradient step. After every epoch we evaluate on the validation set and keep a copy (**checkpoint**) of the model weights; at the end of training we restore the checkpoint with the lowest validation loss, ensuring we use the model state that generalized best rather than the one from the final epoch. Sequences are dynamically padded per-batch (each batch is padded to its longest sequence, not to a fixed maximum) to avoid wasted computation on padding tokens. At evaluation time, the same log-likelihood scoring from Agent 1 is reused, but the LoRA-adapted weights now produce dramatically better predictions because the model has learned which department names are appropriate for which email content.

3.3 Agent 3: DistilBERT Classifier

A different paradigm: discriminative classification. Agents 1 and 2 use *generative* models (GPT-2): they predict the next token and we repurpose this ability for classification by scoring label names. Agent 3 takes a fundamentally different approach: it uses DistilBERT, a *discriminative* encoder model designed specifically for understanding text. While GPT-2 reads text left-to-right (each token can only attend to previous tokens), DistilBERT uses **bidirectional attention**—every token attends to every other token simultaneously, giving the model a richer understanding of the full email context.

How it works. DistilBERT (66 M parameters, 6 transformer layers) processes the email through its encoder and produces a contextualized representation for the special `[CLS]` token—a synthetic token prepended to every input that serves as a summary of the entire sequence after all attention layers have processed it. A linear **classification head** (a single fully-connected layer) maps this representation to **logits** (raw, unnormalized scores) over the 5 department labels. A **softmax** function then converts the logits into a probability distribution that sums to 1, and the label with the highest probability is selected. Unlike Agents 1–2, classification requires only a **single forward pass**—there is no need to score each label separately, making inference much faster.

How it is applied. The subject and body of each email are concatenated into a single string, tokenized, and truncated to 512 tokens (DistilBERT's maximum input length). All 66 M parameters are fine-tuned end-to-end with **cross-entropy loss**, which measures how far the model's predicted probability distribution is from the true label and drives the model to assign high probability to the correct department. We also apply **gradient clipping** (max norm = 1.0), which rescales gradients when their norm exceeds the threshold, preventing large, destabilizing parameter updates. Training runs for 3 epochs with AdamW (lr = 2×10^{-5}—a smaller learning rate than Agent 2 because all parameters are updated, so smaller steps are needed to avoid overshooting; batch size = 16—larger than Agent 2 because DistilBERT's single-pass inference uses less memory per sample). The best checkpoint is restored based on validation loss. We initially experimented with **class-weighted**

cross-entropy (weights inversely proportional to class frequency) to address the severe imbalance, but the weights were too aggressive—*General Inquiry* received 19× the loss of *Technical Support*—causing majority-class recall to drop from ∼85% to ∼63% and overall accuracy to fall from ∼77% to ∼68%. We therefore use standard (unweighted) cross-entropy in the final model.

4 Results

4.1 Overall Comparison

Table 1 summarizes the performance of all five model configurations on the test set.

Table 1: Test set results across all agents.

Agent	Model	Params	Trainable	Accuracy	Time (s)	GPU Mem (MB)
1 – Frozen	GPT-2 Prompting	124 M	0 (frozen)	22.7%	288	—
	DistilGPT-2 Prompting	82 M	0 (frozen)	28.4%	179	—
2 – LoRA	GPT-2 + LoRA	126 M	1.62 M (1.29%)	70.7%	422	—
	DistilGPT-2 + LoRA	83 M	0.81 M (0.98%)	70.4%	246	—
3 – Classifier	DistilBERT	67 M	67 M (100%)	76.6%	18	—

"—" = fill after re-running the notebook.

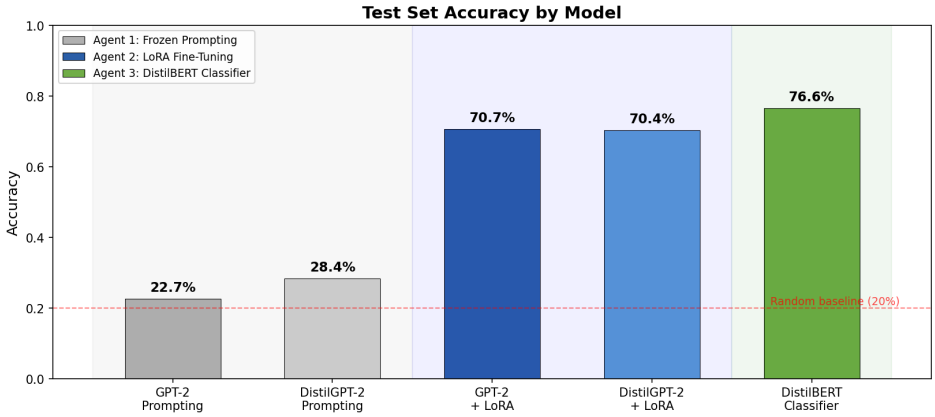

Figure 2: Test set accuracy by model and agent.

4.2 Prompt Sensitivity Analysis

We tested four prompt variants on both frozen models. Surprisingly, the minimal variant (B)—which contains only the email text followed by `"Department:"`—performs best (GPT-2: 45.0%, DistilGPT-2: 44.1%), substantially outperforming the instruction-based baseline (A) and all other variants. This is because GPT-2 is a base language model, not instruction-tuned: shorter prompts keep the email content close to the completion target, while longer prompts with instructions or examples confuse the model and waste context window space. The role-based variant (D) performs worst (10–12%), confirming that role-play framing is ineffective for non-instruction-tuned models. No variant approaches the fine-tuned models' performance.

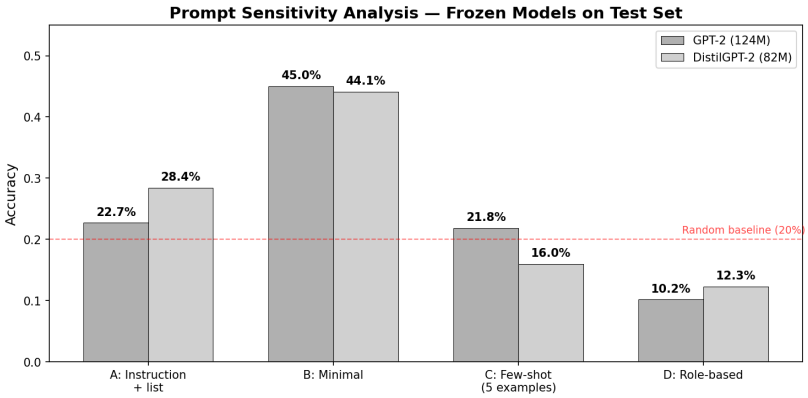

Figure 3: Prompt sensitivity analysis on frozen models.

4.3 Per-Class Performance

Fine-tuned models perform well on majority classes (*Technical Support*: up to 84% F1; *Billing and Payments*: up to 85% F1) but struggle with minority classes (*General Inquiry*: 29 training samples, up to 41% F1; *Sales and Pre-Sales*: 93 training samples, up to 39% F1). DistilBERT achieves the most balanced performance across the majority classes. The primary confusion pattern is between *Customer Service* and *Technical Support*, reflecting genuine semantic overlap in the email content.

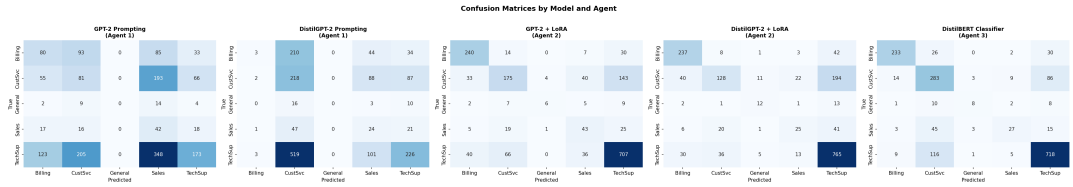

Figure 4: Confusion matrices for all five models.

5 Discussion and Conclusions

1. **Zero-shot prompting is insufficient** for domain-specific classification. Frozen GPT-2 models achieve near-random accuracy (22–28%), confirming that base language models lack task-specific knowledge without fine-tuning.
2. **LoRA delivers outsized returns.** Training just ∼1% of parameters yields +42 to +48 percentage point accuracy gains, making it an excellent option when compute is limited.
3. **Discriminative models outperform generative ones.** DistilBERT (76.6%) beats GPT-2+LoRA (70.7%) despite being smaller (67 M vs. 126 M parameters), thanks to bidirectional attention and a native classification objective.
4. **DistilBERT is both faster and more accurate,** processing ∼94 emails/s versus ∼4 for GPT-2+LoRA—a ∼24× speedup due to single-pass inference.
5. **Class imbalance remains the primary bottleneck.** Techniques such as oversampling or data augmentation could further improve minority-class performance.