machine learning & knowledge extraction

MDPI

Article

A Study on Text Classification in the Age of Large Language Models

Paul Trust *,† **and Rosane Minghim** †

School of Computer Science and Information Technology, University College Cork, T12 XF62 Cork, Ireland; r.minghim@cs.ucc.ie

* Correspondence: 120222601@umail.ucc.ie

† These authors contributed equally to this work.

Abstract: Large language models (LLMs) have recently made significant advances, excelling in tasks like question answering, summarization, and machine translation. However, their enormous size and hardware requirements make them less accessible to many in the machine learning community. To address this, techniques such as quantization, prefix tuning, weak supervision, low-rank adaptation, and prompting have been developed to customize these models for specific applications. While these methods have mainly improved text generation, their implications for the text classification task are not thoroughly studied. Our research intends to bridge this gap by investigating how variations like model size, pre-training objectives, quantization, low-rank adaptation, prompting, and various hyperparameters influence text classification tasks. Our overall conclusions show the following: 1—even with synthetic labels, fine-tuning works better than prompting techniques, and increasing model size does not always improve classification performance; 2—discriminatively trained models generally perform better than generatively pre-trained models; and 3—fine-tuning models at 16-bit precision works much better than using 8-bit or 4-bit models, but the performance drop from 8-bit to 4-bit is smaller than from 16-bit to 8-bit. In another scale of our study, we conducted experiments with different settings for low-rank adaptation (LoRA) and quantization, finding that increasing LoRA dropout negatively affects classification performance. We did not find a clear link between the LoRA attention dimension (rank) and performance, observing only small differences between standard LoRA and its variants like rank-stabilized LoRA and weight-decomposed LoRA. Additional observations to support model setup for classification tasks are presented in our analyses.

Keywords: text classification; language models; fine-tuning; prompting

Citation: Trust, P.; Minghim, R. A Study on Text Classification in the Age of Large Language Models. *Mach. Learn. Knowl. Extr.* **2024**, *6*, 2688–2721. https://doi.org/10.3390/make6040129

Academic Editor: Irena Spasić

Received: 10 August 2024
Revised: 3 November 2024
Accepted: 12 November 2024
Published: 21 November 2024

1. Introduction

Text classification is a core task in natural language processing (NLP) involving the assignment of specific categories to text sequences. The field has seen a surge in interest due to the exponential increase in digital content across social networks, blogs, news platforms, and online scholarly databases [1]. This surge has made text classification a valuable tool for a range of applications such as sentiment analysis [2], topic classification, news categorization [3], and question answering [4]. The automation of text classification through machine learning (ML) is crucial to ensure more objective, fast, and reliable results [5].

Traditionally, text classification has involved collecting training text collections along with their human-curated labels, followed by training classifiers from scratch using supervised learning tailored to specific tasks [6]. This approach has been instrumental across many ML endeavors [7]. Nevertheless, the limited availability of fully annotated datasets is a significant obstacle in the process of training high-quality models. Initial efforts to address these limitations in NLP focused on feature engineering, incorporating domain expertise to identify relevant features from raw data, thus providing models with the necessary inductive bias to learn effectively from limited data [8,9]. However, with the adoption of neural network techniques for text classification, the trend is gradually shifting from purely supervised learning to transfer learning approaches [10,11]. Transfer learning begins with

pre-training models on extensive raw text datasets using self-supervised approaches such as generative pre-training like next token prediction or discriminative pre-training like masked language modelling [4], allowing models to learn universal linguistic features. The pre-training phase is crucial for acquiring vast amounts of knowledge, which can be stored in their parameters. These pre-trained LLMs are more easily adapted to various downstream tasks by fine-tuning on task-specific datasets [4,12,13].

The growing size of LLMs [12,14,15] enables them to acquire extensive parametric knowledge during pre-training, allowing them to execute tasks directly from human prompts without requiring fine-tuning [16]. This is particularly useful in applications like text classification, where LLMs can generate labels from text inputs and task instructions without needing updates to their parameters. To enhance this process, there is a method called few-shot prompting, where demonstration examples are included in the prompts to guide the generation [17]. However, the performance of prompting depends on the model size and is still not as effective as fine-tuning approaches. Fine-tuning large LLMs, especially those with billions of parameters, is challenging due to the computational and data demands involved. Techniques like prefix tuning [18], prompt tuning [19], and low-rank adaptation (LoRA) [20] have been developed to reduce the number of parameters that need to be adjusted. Furthermore, techniques such as Qlora [21] enable fine-tuning models at significantly lower precision, such as 4 bits instead of the conventional 16 bits, thereby reducing hardware demands. Interestingly, studies have indicated that this approach does not result in a significant decrease in performance across various text generation tasks. Other approaches have utilized LLMs to generate training data often known as teacher LLMs used for fine-tuning other LLMs known as student LLMs, demonstrating that this method can be competitive with human-curated data, particularly in text generation tasks [22].

Most advancements in LLMs have been evaluated for text generation tasks such as question answering and summarization, with their impact on text classification requiring further exploration. This research investigates the trade-offs of different adaptation methodologies in text classification using LLMs, focusing in particular applications like news classification, sentiment analysis, and emotion categorization. Our experiments demonstrate that while LLMs with more parameters achieved the best performance, the trend was not linear and depended on the model family. For instance, early-generation discriminatively pre-trained models like BERT can be as competitive in text classification as generatively pre-trained models with ten times the number of parameters. For adaptation techniques, zero-shot prompting, which obtained an average F1 score of 63.90%, and few-shot prompting with 68.34% still lag behind fine-tuning approaches, which have the best average accuracy of 85.65%. Despite claims that quantization shows no performance loss for text generation tasks [21], we found a performance difference in fine-tuning models at different bit precisions for text classification, 16-bit with an average accuracy of 85.65%, 8-bit with 74.94%, and 4-bit with 71.11%. Additionally, we observed a performance drop from 81% to 77.80% when applying low-rank adaptation in text classification. For LoRA parameter configurations, we found that increasing LoRA dropout values leads to a decrease in classification performance. Additionally, the performance differences between LoRA variants, such as weight-decomposed LoRA and rank-stabilized LoRA, compared to standard LoRA were minimal.

2. Background and Related Work

Text classification is an application of machine learning (ML) used to categorize texts into predetermined categories widely utilized in tasks such as sentiment analysis, spam filtering, hate speech detection, and news classification, among others [23]. It is one application of language models (LMs), which are computational models capable of comprehending and generating human language [4,16]. LMs can predict the likelihood of word sequences or generate new text based on given input. Designing text classification models traditionally begins with preprocessing steps such as removing stopwords, stemming, lowercasing, and tokenization. The text is then converted into a numeric format using techniques like

Bag-Of-Words (BOW) [24], N-gram [25], Term Frequency–Inverse Document Frequency (TF-IDF) [26], Word2Vec [27], and contextualized word embeddings [4].

Building on these numerical representations, traditional ML models like Naive Bayes [28], K-Nearest Neighbor [29], Support Vector Machines [30,31], and decision trees [32] have been commonly used for text classification tasks. Examples of traditional ML applications include the performance analysis of supervised machine learning algorithms in text classification by Mishu et al. (2016) [33] and the application of the expectation maximization algorithm to text classification by Nigam et al. (2006) [34]. However, these traditional models often struggle to capture long-term dependencies in the text, which affects their downstream performance. Moreover, they require extensive feature engineering and text preprocessing. Deep learning approaches, especially recurrent neural networks (RNNs) and their variants [35,36], were proposed to address the limitations of traditional machine learning models. Examples of applications in text classification include Liu et al. (2016), who applied an RNN-based model for semantic analysis of extensive texts [37], and Wang et al. (2016), who introduced the Bilateral Multi-Perspective Matching (BiMPM) model, which uses a bidirectional long short-term memory (LSTM) for natural language inference [38]. Additionally, Convolutional Neural Networks (CNNs), initially designed for image classification, were adopted for text classification with models like TextCNN being proposed to identify key phrases through convolutional layers and max pooling [39]. The BLSTM-2DCNN model integrates RNNs and CNNs for text classification by combining a bidirectional LSTM with two-dimensional max pooling for enhanced context understanding [40].

The recurrent nature of RNN-based architectures limits the parallelization and scaling of language models, leading to the development of a new architecture called the transformer [41]. Transformers use an attention mechanism and eliminate recurrence, achieving wide success in many NLP tasks including text classification. With transformers, a new learning paradigm emerged in NLP: pre-train then prompt or fine-tune where models are first pre-trained on extensive text corpora using unsupervised learning objectives like next token prediction [16] or masked language modeling [4]. This pre-training allows models to learn rich linguistic features that can be saved and initialized during fine-tuning on specific tasks with less data and training time [4,16]. These pre-trained models have significantly improved text classification performance. Scaling models in size and dataset volume was found to correlate with performance [42], prompting the NLP community to train even larger language models on trillions of texts containing billions of parameters such as GPT-4 [43] and LLAMA2 [44].

These models, especially those with billions of parameters, can perform text classification from human-written instructions known as prompts without the need for fine-tuning. This is impressive because a single pre-trained model can be adapted to various text classification tasks. The performance of prompting can be further improved with in-context learning, where a few demonstration text examples are added to the prompt without any parameter updates. However, prompting and in-context learning are often surpassed by fine-tuning in most NLP tasks [16,45]. Fine-tuning large-scale LLMs demands considerable computational resources and large datasets, which results in high energy usage, financial costs, and environmental effects due to the carbon footprint associated with powering modern tensor processing hardware [46,47]. These costs limit accessibility for smaller organizations, highlighting the need for more ethical and environmentally friendly LLM training [48]. To address this, several parameter-efficient approaches have been proposed, including parallel architectures [49], low-rank adaptation (LoRA) [20], and model quantization [50]. Parallel architectures spread a model across multiple GPUs, depending on the available hardware [49]. LoRA reduces training parameters by freezing pre-trained weights and adding trainable low-rank matrices, decreasing GPU memory demands [20]. Model quantization lowers precision to 4 bits, minimizing hardware requirements with minimal accuracy loss [50]. With all these new architectures and adaptation

techniques, it becomes challenging to determine what works and what does not work for text classification.

Several studies have explored the use of LLMs for text classification. For example, Sun et al. (2023) [51] evaluated large language models for text classification but focused solely on prompt-based approaches. Similarly, Lepagnol et al. (2024) examined different LLMs for zero-shot classification using prompts and various scoring methods [52]. Chae et al. (2023) compared fine-tuning and prompting for LLM text classification but did not consider important factors like model size and quantization [53]. Yu et al. (2023) analyzed open and closed models for text classification but did not discuss parameter-efficient fine-tuning [54]. In contrast, our research offers a comprehensive evaluation of various adaptation strategies, including parameter-efficient techniques such as quantization, low-rank adaptation, prompting, few-shot learning, and weakly supervised learning, while also considering the effects of hyperparameters like model size, pre-training objectives, and adaptation methods on classification performance.

3. Methodology

In a text classification task, given a dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where each x_i is a text sample and y_i is the corresponding label, the objective is to develop a function $f : X \to Y$ that accurately predicts labels for any text sample. This function can be taught to the model through various methods: traditionally by learning from scratch, and more recently through fine-tuning, prompting, or in-context learning. Fine-tuning can be performed with full parameter updates using human-labeled data or through parameter-efficient strategies such as quantization [21], adapters [20], prompt tuning [17], and weak supervision. This research experiments with and evaluates various adaptation techniques for pre-trained LLMs, examining their benefits and trade-offs in text classification, as detailed in the next sections.

3.1. Discrete Prompting

We start by discussing prompting, also known as discrete prompting, a zero-shot method where large language models (LLMs) perform new tasks during inference by following instructions, without needing any parameter updates or extra training [16]. This involves designing an instruction in human language that describes the task of interest (prompts). Table 1 shows all the different prompts used in this research for various classification tasks. In addition to task instructions, prompts can include a set of k demonstration examples, which are either similar to the text we are interested in predicting or randomly selected from the dataset. Despite the inclusion of these examples, no parameter updates occur, a method known as in-context learning or few-shot prompting. A sample prompt for topic classification with demonstration examples is shown in Figure 1.

More formally, given a query input text x and a set of candidate answers $Y = \{y_1, \ldots, y_m\}$, where Y represents labels in text classification, a pre-trained language model M selects the candidate answer with the highest probability score conditioned on a prompt C. The prompt C comprises a task instruction I and optionally k demonstration examples $C = \{I, s(x_1, y_1), \ldots, s(x_k, y_k)\}$, where $s(x_k, y_k)$ represents an example written in natural language relevant to the task. The likelihood of a candidate answer y_j is determined by a scoring function f on the entire input sequence as shown in Equation (1) [55]:

$$P(y_j \mid x) = f_{\mathcal{M}}(y_j, C, x) \tag{1}$$

The final predicted label $\hat{y}$ is the candidate answer with the highest probability: $\hat{y} = \arg\max_{y_j \in Y} P(y_j \mid x)$.

Table 1. Overview of datasets and their classification prompts.

Dataset Name	Classification Prompt
IMDB [56]	Determine the sentiment of the text provided. The sentiment can be one of the following categories: - positive - negative Text: {text} Sentiment: The sentiment must one of the following: ['positive', 'negative']
Agnews [57]	Categorize the provided text into one of the specified categories: - World - Sports - Business - Science Text: {text} Category: The category must one of the following: ['World', 'Sports', 'Business', 'Science']
DailDialog [58]	What is the emotion of the given text: - none - anger - disgust - fear - happiness - sadness - surprise Text: {text} Emotion: Your answer must be one of the following: ['none', 'anger', 'disgust', 'fear', 'happiness', 'sadness', 'surprise']
Emotion [59]	What is the emotion of the given text: - anger - fear - joy - love - sadness - surprise Text: {text} Emotion: Your answer must be one of the following: ['anger', 'fear', 'joy', 'love', 'sadness', 'surprise']
Financial Sentiment [60]	Determine the financial sentiment of the given text: The sentiment can be one of following - Bearish - Bullish - Neutral Text: {text} Sentiment: Sentiment must be one of the following; ['Bearish', 'Bullish', 'Neutral']
Political News [61]	Does an event sentence contain any cause–effect meaning? Sentences are labeled as Causal or Non-causal, where the presence of causality indicates that one argument provides the reason, explanation, or justification for the situation described by the other. Text: text Category: The category must one of the following: ['Causal', 'Non-causal']

In-context Learning Prompt for Topic Classification

```
What is the topic for a given news article?

Choose from these predefined topics:

- World
- Sports
- Business
- Science

Article: Thursday's Baseball Roundup (CP) - Somehow, in the
    span of half an hour, the Detroit Tigers' pitching went
    from brutal to brilliant. Shortly after being on the
    wrong end of several records in a 26-5 thrashing from the
     Kansas City
Topic: Sports

Article: City Series-ly wounded With the Yanks out of the
    World Series, the city's economy loses out on at least
    $40 million, according to studies by the controller's
    office and other city agencies.
Topic:
```

Figure 1. In-context learning prompt for topic classification.

3.2. Fine-Tuning

In this section, we describe various parameter-efficient fine-tuning strategies for pre-trained LLMs. These strategies are compared with the full parameter update fine-tuning method, which updates all parameters of the pre-trained LLMs without freezing any and performs updates in full precision.

Continuous Prompt Tuning

When we discussed prompting, we referred to discrete prompts written in human language. For example, to perform sentiment classification with discrete prompts, you prompt a model with the following instruction: *Determine the sentiment of the provided text*. In contrast, continuous prompt tuning uses "soft prompts" that operate directly within the model's embedding space, avoiding the constraints of natural language prompts. These prompts are flexible, allowing for customization beyond the pre-trained model parameters, and are specifically tuned for individual downstream tasks. We will experiment and evaluate two continuous prompt tuning strategies: prefix tuning [18] and prompt tuning [19].

Prefix tuning [18] is a method where a learned sequence, known as a prefix, is added to the beginning of an autoregressive language model input, resulting in $z = [\text{PREFIX}; x]$, where z represents the combined sequence of the prefix and the original input x. The purpose of the prefix is to guide the model's behavior by altering its hidden representations. Let P_θ represent the trainable matrix of prefix parameters, with the prefix length denoted as $|\mathrm{P}_{\mathrm{idx}}|$. The hidden state h_i is calculated as shown in the equation below.

$$h_i = \begin{cases} P_\theta[i,:], & \text{if } i \in \mathrm{P}_{\mathrm{idx}} \\ \mathrm{LM}_\theta(z_i, h_{<i}), & \text{otherwise} \end{cases}$$

In this context, P_{idx} refers to the indices of the prefix, while LM_θ represents the language model's output for input z_i, conditioned on the preceding hidden states $h_{<i}$. The

training objective remains similar to standard language modeling, but only the prefix parameters P_θ are optimized. This means that instead of fine-tuning the entire language model, only the prefix affects the hidden activations during training. To improve the expressiveness of the prefix and allow it to capture more task-specific details, the prefix parameters $\mathbf{P}$ are often processed through a Multi-Layer Perceptron (MLP), represented as $\mathbf{P}_\theta = \text{MLP}(\mathbf{P}_0)$. In this case, $\mathbf{P}_0$ refers to the initial prefix, and $\text{MLP}(\cdot)$ is a feed-forward neural network that refines $\mathbf{P}_0$ into the updated representation $\mathbf{P}_\theta$. This transformation makes the prefix more adaptable to different tasks, while keeping the rest of the model architecture unchanged [18].

Prompt tuning [19] is a simplified version of the prefix tuning method [18] which enhances language models by adding a layer of soft prompt tokens P, refining output generation Y without altering the core parameters θ of the model. This method introduces a customizable sequence of continuous vectors $P = \{p_1, p_2, \ldots, p_n\}$ with their own tunable parameters θ_P, aiming to optimize the conditional probability $\Pr_{\theta;\theta_P}(Y \mid [P; X])$, where X is the input. Unlike traditional prompting, which relies on discrete tokens from a fixed vocabulary, prompt tuning integrates these vectors with the input embeddings X_e, forming an enhanced input matrix $[P_e; X_e] \in \mathbb{R}^{(p+n)\times e}$. The goal is to maximize the likelihood of correctly generating Y, focusing adjustments on the soft prompts P_e while keeping the original model parameters θ unchanged. This approach diverges from conventional prompt design by dynamically tuning the embeddings of a predefined set of tokens, offering an efficient way to adapt models to new tasks without expanding the neural network's tunable parameters.

3.3. Low-Rank Adaptation

Low-rank adaptation (LoRA) is a technique that efficiently fine-tunes large language models (LLMs) by updating only a subset of parameters, thus reducing computational and memory requirements. Consider a pre-trained autoregressive language model $P_\Phi(y|x)$ parameterized by Φ. For a downstream task with a dataset $Z = \{x, y\}^N$, where each x is text and y is the corresponding class label, full fine-tuning starts with the pre-trained weights Φ_0 and updates them to $\Phi_0 + \Delta\Phi$ by repeatedly following the gradient to maximize the conditional language modeling objective shown in Equation (2). Here, t represents the time step in a sequence of tokens when modeling language and $y_{<t}$ represents the sequence of tokens before the token at t.

$$\max_{\Phi} \sum_{(x,y)\in Z} \sum_{t=1}^{|y|} \log(P_\Phi(y_t \mid x, y_{<t})) \tag{2}$$

A major drawback of full fine-tuning is that it requires learning a separate set of parameters $\Delta\Phi$ for each downstream task, where the size of $\Delta\Phi$ is the same as the original pre-trained LLM parameters $|\Phi_0|$. For large pre-trained LLMs with billions of parameters, storing and updating these initial parameters $|\Phi_0|$ becomes technically complex and computationally expensive. LoRA provides a more efficient solution by representing the task-specific parameter update $\Delta\Phi = \Delta\Phi(\Theta)$ with a smaller set of parameters Θ, where $|\Theta| \ll |\Phi_0|$. The task then focuses on optimizing Θ, as shown in Equation (3).

$$\max_{\Theta} \sum_{(x,y)\in Z} \sum_{t=1}^{|y|} \log\Big(P_{\Phi_0+\Delta\Phi(\Theta)}(y_t \mid x, y_{<t})\Big) \tag{3}$$

LoRA uses a low-rank representation to encode $\Delta\Phi$, improving both computational and memory efficiency. This technique enables LLMs with billions of parameters to have a set of trainable parameters in Θ as small as 0.01% of the size of $|\Phi_0|$. LoRA achieves this by keeping the pre-trained model weights fixed and introducing trainable low-rank matrices $A \in \mathbb{R}^{d\times r}$ and $B \in \mathbb{R}^{r\times k}$ to each layer of the transformer architecture. This reduces the number of trainable parameters needed to update a pre-trained weight matrix $W_0 \in \mathbb{R}^{d\times k}$,

expressed as $W_0 + \Delta W = W_0 + BA$. Here, r, the rank of the adaptation, is much smaller than d or k, resulting in a significant reduction in trainable parameters.

During training, W_0 stays fixed, while A and B are updated using optimization methods like stochastic gradient descent. This selective updating enables the model to learn task-specific changes (ΔW) as the product of A and B. For the output h, given an input x, the modified forward pass is described in Equation (4).

$$h = W_0 x + \Delta W x = W_0 x + BAx \tag{4}$$

3.4. Model Quantization

LLMs are increasingly demanding significant GPU memory for both training and inference. In transformer models with 6.7 billion parameters or more, the feed-forward and attention projection layers along with their matrix multiplications account for about 95% of the parameters and $65 - 85\%$ of the total computations [62]. Many quantization techniques, which use low-bit precision matrix multiplication, can reduce the memory required to train these LLMs, but they often result in decreased model performance. This discussion will evaluate two quantization methods: one reducing LLMs to 8 bits [63] and another to 4 bits [21], both decreasing memory requirements without compromising performance.

8-bit Quantization: Before exploring more robust quantization methods that reduce degradation when converting LLMs to 8 bits, let us first discuss two of the most common techniques: absmax and zero-point quantization. Absmax quantization scales inputs into the 8-bit range $[-127, 127]$ by multiplying each element X_{16} by $s_{x_{16}}$, where $s_{x_{16}}$ is 127 divided by the absolute maximum value of the tensor. This process is equivalent to dividing by the infinity norm and then multiplying by 127. Mathematically, for an FP16 input matrix X_{16} in $\mathbb{R}^{m \times n}$, the absmax quantization is given by Equation (5).

$$X_8 = \left\lfloor \frac{127 \cdot X_{16}}{\max(|X_{16}|_\infty)} \right\rceil = \left\lfloor \frac{127}{\|X_{16}\|_\infty} X_{16} \right\rceil = \lfloor s_{x_{16}} X_{16} \rceil \tag{5}$$

Here, $\lfloor \cdot \rceil$ indicates rounding to the nearest integer. This method ensures that the tensor values are scaled to fit within the 8-bit integer range, preserving as much information as possible within the limited bit width.

Zero-point quantization shifts the input distribution to fit within the 8-bit range $[-127, 127]$ by scaling with the normalized dynamic range $nd_{x_{16}}$ and shifting by the zero-point $zp_{x_{16}}$. The process involves calculating the normalized dynamic range shown in Equation (6)

$$nd_{x_{16}} = \frac{\max(y) X_{16}^j - \min(y) X_{16}^j}{2 \cdot 127} \tag{6}$$

determining the zero-point $zp_{x_{16}} = \left\lfloor X_{16} - \min(y) X_{16}^j \right\rceil$, and then quantizing the tensor $X_8 = \lfloor nd_{x_{16}} X_{16} \rfloor$. In operations, both X_8 and $zp_{x_{16}}$ are used, adding $zp_{x_{16}}$ to each element of X_8 before a 16-bit integer operation.

The primary challenge with these quantization methods is that they use a single scaling constant per tensor, which is problematic, especially in the presence of outliers. Outliers can significantly reduce the quantization precision of all other values. To address this, 8-bit quantization degradation-free methods employ mixed-precision decomposition. In this approach, a small number of large-magnitude feature dimensions (about 0.1%) are represented in 16-bit, while the remaining 99.9% of the values are processed using 8-bit multiplication. For a given input $X_{f16} \in \mathbb{R}^{s \times h}$, outliers occur systematically across almost all sequence dimensions s but are limited to specific feature/hidden dimensions h. The Int8 degradation-free method employs mixed-precision decomposition for matrix multiplication by separating outlier feature dimensions into a set $O = \{i \mid i \in \mathbb{Z}, 0 \leq i \leq h\}$. This set O contains all dimensions of h that have at least one outlier with a magnitude larger than the threshold α, which was set to 0.6 in their research [63]. The mixed-precision approach combines vector-wise quantization and mixed-precision decomposition using

16-bit precision for outliers and 8-bit precision for general data efficiently as shown in Equation (7).

$$\begin{aligned} C_{f16} &\approx \sum_{h \in O} X^h_{f16} W^h_{f16} + S_{f16} \sum_{h \notin O} X^h_{i8} W^h_{i8} \\ &\approx X_{hf16} W_{fh16} + S_{f16} \cdot X_{hi8} W_{ih8}, \quad h \in O, h \notin O \end{aligned} \tag{7}$$

This separation into 8-bit and 16-bit precision allows for high-precision multiplication of outliers while performing memory-efficient matrix multiplication with 8-bit weights for more than 99.9% of the values.

4-bit Quantization (QLoRA): QLoRA [21] is an efficient fine-tuning method that reduces the memory usage of LLMs, enabling training in 4-bit precision without performance degradation. QLoRA employs 4-bit NormalFloat (NF4) quantization, a data type optimized for normally distributed weights. It also utilizes Double Quantization, which further reduces memory usage by quantizing the quantization constants, and Paged Optimizers, which effectively manage memory spikes. Paged Optimizers use the NVIDIA unified memory feature to automatically perform page-to-page transfers between the CPU and GPU. This ensures error-free GPU processing when the GPU occasionally runs out of memory, functioning similarly to regular memory paging between CPU RAM and disk. The feature allocates paged memory for optimizer states, automatically moving them to CPU RAM when the GPU runs out of memory and paging them back into GPU memory when needed during the optimizer update step.

4-bit NormalFloat (NF4) quantization is a data type optimized for normally distributed weights that uses an optimal data type for normally distributed weights by assigning an equal number of values to the input range. This method estimates the quantiles of the input tensor through empirical cumulative distribution functions. The main challenge is the expensive process of computing quantiles and addressing approximation errors for critical outliers. To address this, NF4 quantization leverages specific quantization constants derived from the standard normal distribution $N(0,1)$ to estimate a k-bit quantization data type. This involves normalizing the values into the $[-1,1]$ range and then rescaling. For an input tensor with arbitrary standard deviations, the steps are as follows: estimating the 2^k+1 quantiles of the $N(0,1)$ distribution, normalizing these quantiles into the $[-1,1]$ range, and quantizing the input tensor by rescaling it to the $[-1,1]$ range. The k-bit quantiles q_i are computed as shown in Equation (8)

$$q_i = \frac{1}{2}\left(Q_X\left(\frac{i}{2^k+1}\right) + Q_X\left(\frac{i+1}{2^k+1}\right)\right) \tag{8}$$

where $Q_X(x)$ is the quantile function of the standard normal distribution $N(0,1)$. This ensures that the distribution of weights aligns with the quantized data type, making the process efficient and effective. Double Quantization (DQ) quantizes the quantization constants themselves to achieve further memory savings. By treating the quantization constants q^{FP32} from the first quantization as inputs to a second quantization, DQ generates new quantized constants q^{FP8}. Using 8-bit floats with a block size of 256, this approach maintains performance without degradation. The process involves centering the values around zero by subtracting the mean before quantization, enabling symmetric quantization. For a block size of 64, this method reduces the memory footprint per parameter from 0.5 bits to 0.127 bits, saving 0.373 bits per parameter. QLoRA is applied to a single linear layer in the quantized base model utilizing the described components. The operation for a single LoRA adapter is represented by Equation (9)

$$Y^{BF16} = X^{BF16} \cdot \text{doubleDequant}(q^{FP32}_{c_1,c_2}, \text{W}^{NF4}) + X^{BF16} \cdot L_2^{BF16} \cdot L_1^{BF16} \tag{9}$$

where the doubleDequant function is defined by the following Equation:

$$\text{doubleDequant}(q^{FP32}_{c_1,c_2}, \text{W}^{k\text{-bit}}) = \text{dequant}(\text{dequant}(q^{FP32}_{c_1,c_2}), \text{W}^{4\text{-bit}}) = \text{W}^{BF16} \tag{10}$$

QLoRA uses NF4 for weights and FP8 for quantization, with a block size of 64 for weights to ensure high precision and a block size of 256 for constants c_2 to conserve memory. For parameter updates, only the gradient with respect to the error for the adapter weights $\frac{\partial E}{\partial L}$ is needed, not for the 4-bit weights $\frac{\partial E}{\partial W}$. The calculation of $\frac{\partial E}{\partial L}$ involves computing $\frac{\partial E}{\partial W}$ using double dequantization from storage W^{NF4} to computation data type W^{BF16}, and then calculating the derivative $\frac{\partial E}{\partial X}$ in BFloat16 precision.

3.5. Weakly Supervised Text Classification

In this subsection, we describe the method of fine-tuning or training LLMs with synthetic labels generated by other LLMs instead of using human-provided labels. We consider a dataset D with target labels that can belong to any of the m distinct classes $C = \{c_1, \ldots, c_m\}$. The goal is to assign each data point $x_i \in D$ to one of the classes $c_j \in C$. Traditional supervised text classification methods rely on large volumes of high-quality labeled data, typically curated by human annotators. In contrast, we investigate a weakly supervised framework that leverages noisy labels for training classifiers instead of relying solely on human-annotated data. For evaluation purposes, we assume access to a minimally labeled test dataset $D_{test} = \{x_{test}, y_{test}\}$.

We describe three key stages involved in this approach: The first stage involves using several large-scale LLMs referred to as teacher models to generate synthetic labels. The second stage aggregates outputs from different models to derive an ensemble label. The third stage trains or fine-tunes another LLM using the ensemble label. In the first stage, we use models capable of generating labels through prompting, using prompts similar to those described in the zero-shot learning Section 3.1 of the methods. With these prompts, we generate noisy labels using k LLMs, denoted as $\lambda = \{\lambda_1, \ldots, \lambda_k\}$. Each language model λ_j labels each x_i with $\lambda_j(x) \in \{c_1, \ldots, c_m\}$, corresponding to the classes in C.

In the second stage, we aim to derive a single, less noisy label by aggregating k synthetic labels generated by k LLMs. To achieve this, we utilize various ensemble methods which are described below.

- **Majority Vote**: Majority vote is a straightforward yet effective ensemble learning technique. With k classifiers (in this case, outputs from k LLMs) for the same input text, each provides a vote for one of the possible classes $c \in C$. The majority vote is determined by aggregating these votes and selecting the class with the highest number of votes. Formally, let v_{ic} be a binary indicator, where $v_{ic} = 1$ if classifier λ_i votes for class c, and $v_{ic} = 0$ otherwise. The total votes for class c are given by $V_c = \sum_{i=1}^{N} v_{ic}$. The predicted class $\hat{y}$ is the one with the maximum V_c, formalized as $\hat{y} = \arg\max_c V_c$. This method relies on the principle of "wisdom of the crowd", where the collective decision of multiple classifiers is more reliable than individual decisions, especially when classifiers have diverse and complementary strengths [64].
- **Worker Agreement with Aggregate (WAWA)**: The WAWA algorithm aggregates classification by performing three main steps. First, it calculates the majority vote label for each task. Second, it estimates each worker's (classifier's) skill based on the proportion of their responses matching the majority vote. Third, it computes a weighted majority vote, where the weights are the skill estimates from the previous step. This method assumes that tasks with a majority label are more likely correct and that a worker's skill can be measured by their agreement with the majority [65].
- **Dawid–Skene Aggregation Model**: The Dawid–Skene aggregation model is a probabilistic approach that estimates the expertise level of workers (LLMs) in a crowdsourcing environment using confusion matrices. In this model, the worker confusion matrix denoted as e_w is a $C \times C$ matrix for a C-class classification problem. The model also considers a vector p of prior class probabilities and the true label z_j of a task. The relationship between these parameters is captured by a latent label model. The probability of a true label being c is given as $\Pr(z_j = c) = p[c]$, and the distribution of worker responses with the true label c is given by the corresponding column of the

error matrix: $\Pr(y_{jw} = k \mid z_j = c) = e_w[k, c]$. The parameters p and e_w and the latent variables z are optimized using the Expectation–Maximization algorithm [66].

- **OneCoinDawidSkene**: The OneCoinDawidSkene model [67] is an extension of the Dawid–Skene aggregation model, specifically designed to simplify the calculation of worker errors during the M-step of the Expectation–Maximization (EM) algorithm. In the OneCoinDawidSkene model, a worker's confusion (error) matrix is parameterized by a single parameter s_w, which represents the worker's skill (accuracy). The error matrix e^w_{j,z_j} is defined as in Equation (11).

$$e^w_{j,z_j} = \begin{cases} s_w & \text{if } y^w_j = z_j \\ \frac{1-s_w}{K-1} & \text{if } y^w_j \neq z_j \end{cases} \tag{11}$$

where e^w is the worker confusion (error) matrix of size $K \times K$ for K-class classification, z_j is the true task label, y^w_j is the worker's response to task j, and s_w is the probability that the worker assigns the correct label. In essence, the worker uses a single coin flip to decide their assignment, with s_w being the probability of assigning the correct label and $\frac{1-s_w}{K-1}$ being the probability of assigning an incorrect label.
The OneCoinDawidSkene model simplifies parameter estimation by only needing to estimate s_w for each worker and y^w_j for each task. This model is optimized using the EM algorithm, where the E-step estimates the true task label probabilities using the specified worker responses, prior label probabilities, and workers' error probability matrix. The M-step calculates each worker's skill according to the label probability and then estimates the workers' error probability matrix by assigning user skills to the error matrix row by row. This simplicity enhances the model's convergence properties and makes it easier to estimate [67].
- **Multi-Annotator Competence Estimation (MACE)**: The Multi-Annotator Competence Estimation (MACE) method is a probabilistic model designed to estimate the competence of workers, particularly in crowdsourced settings where multiple annotators contribute to task solutions. MACE assigns each worker a label probability distribution that reflects their likelihood of correctly labeling tasks. In scenarios where a worker might be providing spam responses, MACE considers that for each task, there is a Bernoulli-distributed probability $1 - \theta_j$ that the worker is spamming. Here, S_{ij} represents whether worker j is spamming on instance i. If $S_{ij} = 0$, the worker is assumed to be providing the correct label $A_{ij} = T_i$. Conversely, if the worker is spamming ($S_{ij} = 1$), their response A_{ij} is drawn from a multinomial distribution with the parameter vector ξ_j. The model is further refined by incorporating a Beta prior on θ_j and a Dirichlet prior on ξ_j. The optimization of MACE is performed using the Expectation–Maximization (EM) algorithm. This method provides a robust framework for aggregating annotations from multiple sources while accounting for the varying quality of annotator inputs [68].
- **GLAD (Generative model of Labels, Abilities, and Difficulties)**: GLAD [69] is a probabilistic framework designed to infer true task labels from noisy annotations provided by multiple workers. It achieves this by simultaneously estimating the abilities of the workers and the difficulties of the tasks. In a K-class classification scenario, the model involves several key components: a vector of prior class probabilities $\mathbf{p}$, worker ability parameters α_i, inverse task difficulty parameters β_j, latent variables z_j representing the true task labels, and observed worker responses y^j_{ij}. The model assumes that the prior probability of a latent variable z_j being equal to a class c is given by $\Pr(z_j = c) = p[c]$. The probability distribution of the worker responses, assuming the true label is c, follows a single-coin Dawid–Skene model. In this model, the probability of the true label is a sigmoid function of the product of the worker's ability and the inverse task difficulty:

$$\Pr(y_{ij}^{j} = k \mid z_j = c) = \begin{cases} \frac{a(i,j)}{a(i,j)+(K-1)}, & k = c \\ \frac{1}{a(i,j)+(K-1)}, & k \neq c \end{cases}$$

where $a(i,j) = \exp(\alpha_i \beta_j)$. The parameters $\mathbf{p}$, α, and β, along with the latent variables z, are optimized using the Expectation–Maximization (EM) algorithm. During the E-step, the algorithm estimates the true task label probabilities by considering the workers' ability parameters, prior label probabilities, and task difficulty parameters. In the M-step, it optimizes the worker's abilities and task difficulties using the conjugate gradient method. This iterative process allows the model to effectively integrate noisy annotations and produce more accurate estimates of the true labels by accounting for both the reliability of individual workers and the inherent difficulty of the tasks.

After aggregation, the final step involves fine-tuning a smaller LLM using our curated dataset, which includes aggregate labels (assuming noise reduction through the ensembling process) derived from various noisy LLM-generated labels, represented as $\tilde{D} = \{x, \tilde{y}\}$. Instead of using human-annotated labels, we fine-tune the pre-trained LLM with the aggregated labels generated by other LLMs. A feed-forward neural network is added to the final layers to adapt the model for our downstream text classification task. The model is trained by minimizing the standard cross-entropy loss, as shown in Equation (12).

$$\theta = \arg\min_{\theta} \frac{1}{k} \sum_{i=1}^{k} E_{\bar{y} \sim \bar{Y}}[L(x_i, \bar{y}_i)] \tag{12}$$

where E denotes expectation, L denotes the loss function, and k denotes the number of training examples. The objective function used is similar to a standard supervised learning loss, except that we minimize the expected value with respect to the noisy probabilistic labels $\bar{Y}$ generated by other LLMs rather than human-generated labels.

3.6. Data

We utilized the following text classification datasets to evaluate different adaptation techniques for LLMs: Agnews for topic classification, Political News for causality classification, Twitter Financial News for financial sentiment analysis, IMDB for sentiment analysis, and DailyDialog and Emotion for emotion classification.

Agnews: The Agnews dataset [57] is a collection of news articles gathered from the AG corpus of news websites, serving as a fundamental resource for text classification tasks in the field of natural language processing (NLP). It includes 120,000 training samples and 7600 test samples, categorized into four main classes: World, Sports, Business, and Science/Technology.

Political News: The Political News dataset introduced by Tan et al. (2022) [61] is designed to analyze causal relationships in news texts. It features annotated news stories highlighting cause-and-effect links, making it valuable for natural language understanding and information extraction. The dataset focuses on journalistic content and real-world events with sentences classified as either Causal or Non-causal. This classification aids in examining narrative frameworks and the representation of causality in news.

Twitter Financial News dataset: The Twitter Financial News dataset is an English-language collection of finance-related tweets annotated for sentiment classification. It consists of 11,932 documents each labeled with one of three sentiment categories: Bearish, Bullish, or Neutral [60].

IMDB: The IMDB dataset [56] is a well-known collection of movie reviews widely used in sentiment analysis research within the field of NLP. It consists of 50,000 reviews evenly divided into training and test sets, each containing an equal number of positive and negative reviews. This dataset serves as a benchmark for evaluating the performance of various text classification algorithms, ranging from traditional machine learning models to advanced deep learning approaches.

DailDialog: The DailyDialog dataset [58] is a high-quality collection of conversations, notable for its human-written content that closely resembles real-life daily communication with minimal errors. Covering a wide range of everyday topics, it is particularly useful for NLP and dialogue system research. The dataset includes manual annotations for communication intentions and emotional information, providing valuable insights into human interaction. The emotions are categorized into seven types: no emotion, anger, disgust, fear, happiness, sadness, and surprise. The training set comprises 11,118 examples, while both the validation and test sets contain 1000 examples each.

Emotion: Emotion [59] is a dataset that was collected using noisy labels and annotated through distant supervision. This dataset is specifically designed for the recognition of six emotions: anger, fear, joy, love, sadness, and surprise. It was introduced to aid in the development and evaluation of models for emotion recognition tasks, focusing on understanding and modeling the nuanced ways emotions are expressed through text. The dataset supports the advancement of natural NLP technologies by providing a resource for training models to recognize emotional states from textual data, thereby contributing to the broader goal of enhancing machine understanding of human emotions. The dataset consists of 16,000 training, 2000 validation, and 2000 test examples.

3.7. Experimental Setup

Here, we summarize the building blocks of our experiments in regard to models, training, and evaluation.

3.7.1. Models

We used the following large language models (LLMs) for the experiments, which are representative of BERT, RoBERTa, OPT, LLAMA, Mistral, Zephyr, and Falcon.

BERT: BERT [4] is a transformer model pre-trained in a bi-directional context using masked language modeling and next sentence prediction. It was trained on BookCorpus (800 million words) and English Wikipedia (2.5 billion words). The model can be fine-tuned for downstream tasks, with word embeddings extracted from its last layers.

RoBERTa: RoBERTa (Robustly Optimized BERT Pre-training Approach) [70] is a transformer-based neural model that enhances BERT by training longer with larger batches, using more data (160 GB of uncompressed text), and removing the next sentence prediction objective.

OPT: OPT [71] is an open-source suite of decoder-only transformers, ranging from 125 million to 175 billion parameters. Pre-trained on diverse datasets, OPT is competitive with GPT-3 and significantly more energy-efficient, requiring only one-seventh of the carbon footprint.

Falcon: Falcon is an LLM with different parameters sizes, 7B, 40B, and 180B, designed as causal decoder-only models trained on diverse, high-quality corpora primarily sourced from web data. The largest model, Falcon-180B, has been trained on over 3.5 trillion tokens. Falcon-180B outperforms models like PaLM and Chinchilla and improves upon models such as LLAMA2 and Inflection-1. It approaches the performance of PaLM-2-Large at reduced pre-training and inference costs, making it one of the top three language models globally, alongside GPT-4 and PaLM-2-Large. The Falcon models are pre-trained using a custom distributed training codebase on up to 4,096 A100 GPUs on AWS infrastructure. A 600B token extract of the web dataset used for pre-training is released under a permissive license to promote open science and the development of an open ecosystem for large language models.

LLAMA2: LLAMA2 [44] encompasses a series of pre-trained and fine-tuned large language models (LLMs) with parameters ranging from 7 billion to 70 billion. Among these is LLAMA-CHAT, specifically optimized for dialogues. LLAMA2 utilizes an optimized autoregressive transformer, pre-trained on publicly accessible data spanning over 2 trillion tokens. The model's fine-tuning incorporates both instruction-based tuning and Reinforcement Learning from Human Feedback (RLHF).

Mistral: Mistral provides two types of large language models (LLMs): open-weight models (Mistral 7B, Mixtral 8x7B, and Mixtral 8x22B) and optimized commercial models (Mistral Small, Medium, Large, and Embeddings). The open-weight models are available under the Apache 2 license and can be fine-tuned for downstream tasks. Mistral 7B excels in performance and efficiency, surpassing the 13-billion-parameter LLAMA2 in all benchmarks and outperforming the 34-billion-parameter LLAMA1 in reasoning, mathematics, and code generation tasks. It features Grouped-Query Attention (GQA) for faster inference and Sliding Window Attention (SWA) for efficient handling of varying sequence lengths, thereby reducing inference costs [72].

Zephyr: Zephyr is a series of language models designed to serve as effective assistants, representing an advancement over Mistral-7B-v0.1 through additional fine-tuning and intent alignment. It was trained using both publicly available and synthetic datasets, with the development process incorporating Direct Preference Optimization (DPO). Zephyr-7B sets a new benchmark for chat model performance among models with 7 billion parameters, significantly outperforming LLAMA2-70B, which was previously the leading open-access model trained using Reinforcement Learning from Human Feedback (RLHF). Notably, Zephyr achieves this superior performance without relying on human annotation [22].

3.7.2. Training and Evaluation Setup

We applied fine-tuning techniques using Huggingface Transformers (https://huggingface.co/) combined with BitsAndBytes (https://github.com/bitsandbytes-foundation/bitsandbytes, accessed on 8 August 2024) for quantization. Parameter-efficient tuning was managed through the PEFT library (https://huggingface.co/docs/peft/en/index, accessed on 8 August 2024), also integrated with transformers. Prompting and in-context learning were performed using both transformers and Langchain (https://www.langchain.com/). Ensembling was implemented using the crowd-kit library (https://github.com/Toloka/crowd-kit, accessed on 8 August 2024), and evaluation metrics were calculated with the Scikit-Learn library (https://scikit-learn.org/). We utilized an NVIDIA A40 GPU for all fine-tuning and prompting tasks.

Experiments with LLMs were conducted using the Adam optimizer [73], with an initial learning rate of 2×10^{-5}, a batch size of 64, and a maximum sequence length of 200 tokens since most of our text were short, where sequence text beyond this was truncated to this sequence length. The models were trained for five epochs, with a weight decay of 0 and a maximum gradient norm of 0.0. For quantization, no bias was applied; we used a LoRA rank of 6, a 4-bit data type (fp4), an Int8 threshold of 6, and a LoRA dropout of 0.0. Training and validation losses were monitored at each epoch, and models with the best validation accuracy were saved for comparison with the test set. Detailed code and data used for these experiments can be accessed on GitHub via this link https://github.com/TrustPaul/Text-Classification-Exploration.git, accessed on 8 August 2024.

We compared the different models and parameter configurations using F1 score, precision, and accuracy (Equations (13)–(15)), along with the receiver operating characteristic (ROC) curve for more detailed error prediction analysis.

$$Precision = \frac{TP}{TP + FP} \tag{13}$$

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{14}$$

$$F1 = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{15}$$

where TP are the true positives, FP are the false positives, FN are the False Negatives, and TN are the True Negatives.

To assess the statistical significance of the results, we applied two statistical tests: the Wilcoxon signed-rank test for comparing two groups across multiple datasets, and the Fried-

man test for comparing multiple groups across multiple datasets [74]. The Wilcoxon signed-rank test, a non-parametric alternative to the paired t-test, compares two classifiers by ranking the differences in their performance across datasets, focusing on the ranks of positive and negative differences rather than the actual values. This method is more robust than the t-test as it does not assume a normal distribution and is less influenced by outliers [74]. The Friedman test, a non-parametric equivalent of repeated-measures Analysis of Variance (ANOVA), is used to compare multiple classifiers by ranking them for each dataset and determining if the average ranks significantly differ [74].

4. Experimental Results and Discussions

In this section, we provide a detailed presentation and discussion of the results and findings from our experiments.

Performance across Datasets: We begin our discussion of the results with experiments to determine if there is any performance difference among all models across different datasets. Figure 2 presents a box plot showing the average F1 score across various datasets: Agnews, Dailydialog, Emotion, Financial Sentiment, IMDB, and Political News. The models perform best on topic classification with the Agnews dataset, achieving an average F1 score of 90%, and on sentiment classification with the IMDB dataset, with an average F1 score of 77.5%. Performance on financial sentiment analysis is moderate, with an average F1 score of 72.5%. Performance on emotion classification varies, with the Dailydialog dataset having an average F1 score of 45%, and the Emotion dataset showing an average F1 score of 70%. These results indicate that LLMs tend to perform better on standard classification benchmark datasets such as Agnews and IMDB, which are publicly available and likely included in the pre-training data of most LLMs. In contrast, their performance declines on less conventional datasets and is the worst on non-public datasets like the Political News dataset. Additionally, the performance of LLMs decreases on datasets with a higher number of classes, such as Dailydialog and Emotion, which have 7 and 6 classes, respectively.

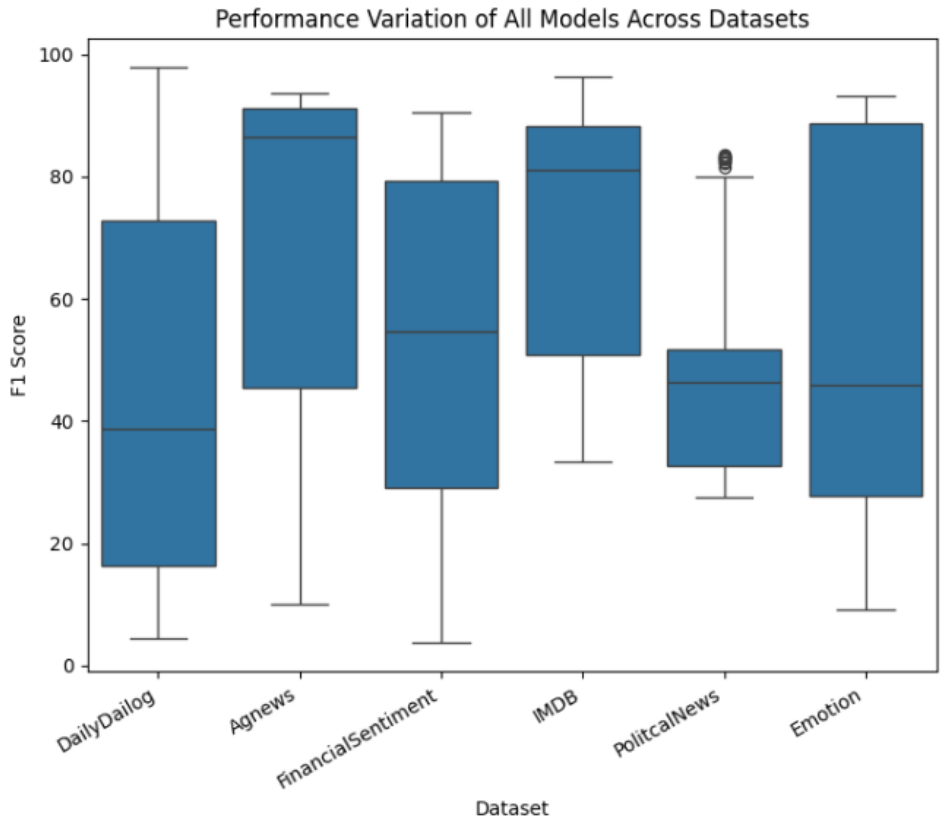


Figure 2. Box plot showing the performance variations of all models across different datasets.

Additionally to the overall results in Figure 2, we conducted an error analysis for each dataset using receiver operating characteristic (ROC) curves to evaluate different models.

Analyzing performance of different models on the Emotion classification task: In the emotion classification task, we examined the prediction errors of various models using the ROC curves in Figure 3, with the areas under the curve (AUCs) ranging from 0.67 to 0.99. The Mistral model achieved the highest performance with an AUC of 0.99, followed by LLAMA at 0.98, demonstrating high classification accuracy and minimal trade-off between true positive and false positive rates. Performance amongst the OPT models varied: OPT-125M had the lowest AUC of 0.67, indicating weaker classification, while the larger models, OPT-6.7B and OPT-13B, reached AUCs of 0.95 and 0.85, respectively. This trend suggests that larger models generally performed better on this dataset, though this did not apply to other datasets.

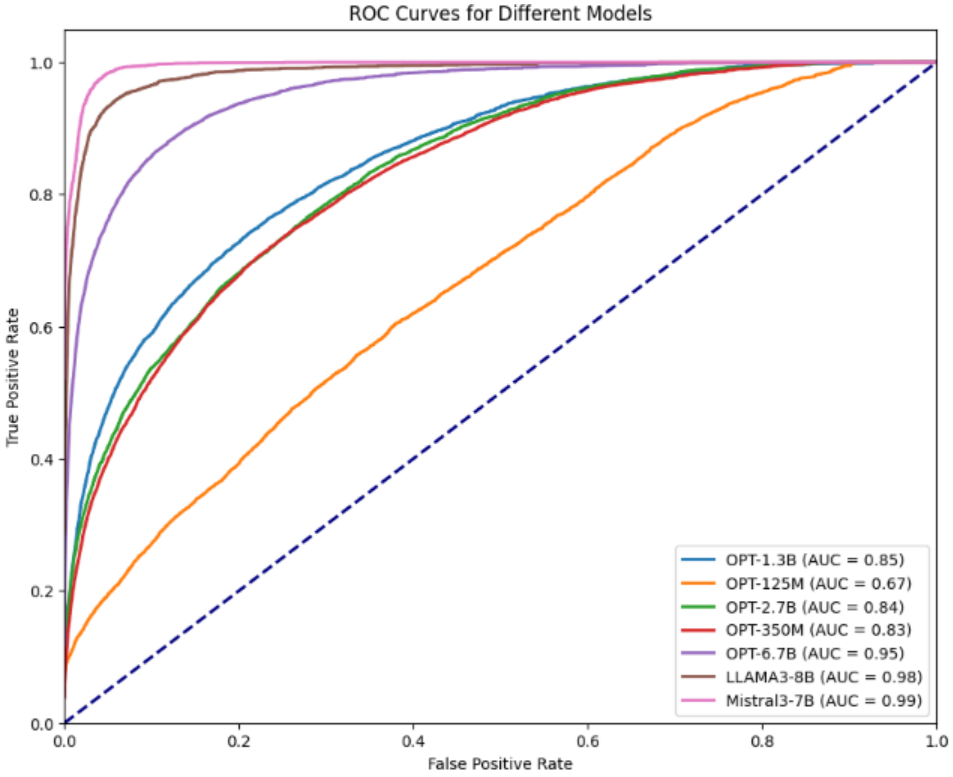


Figure 3. Receiver operating curve of different models on Emotion classification dataset.

Analyzing performance of different models on the sentiment classification task on Financial Sentiment classification dataset: We evaluate model performance on the sentiment classification task using the Financial Sentiment dataset as shown in Figure 4. Mistral achieved the highest AUC of 0.89, followed by OPT-6.7B with 0.86, while OPT-125M had the lowest AUC of 0.65. Other models, including OPT-1.3B, OPT-2.7B, OPT-350M, and LLAMA, consistently achieved AUC values between 0.80 and 0.81.

Analyzing performance of different models on the Dailydialog dataset: We conducted a detailed prediction error analysis on another emotion classification task using the Dailydialog dataset. The ROC curves in Figure 5 show the performance of various models, with AUC values ranging from 0.69 to 0.86. Among the models, both OPT-6.7B and OPT-13B achieve the highest AUC scores of 0.86, followed closely by the OPT-350M model with an AUC of 0.85. The OPT-125M model performs the worst, with the lowest AUC of 0.69. Other models, such as OPT-1.3B and OPT-2.7B, fall in the mid-range, with AUC values of 0.78 and 0.79, indicating moderate classification performance.

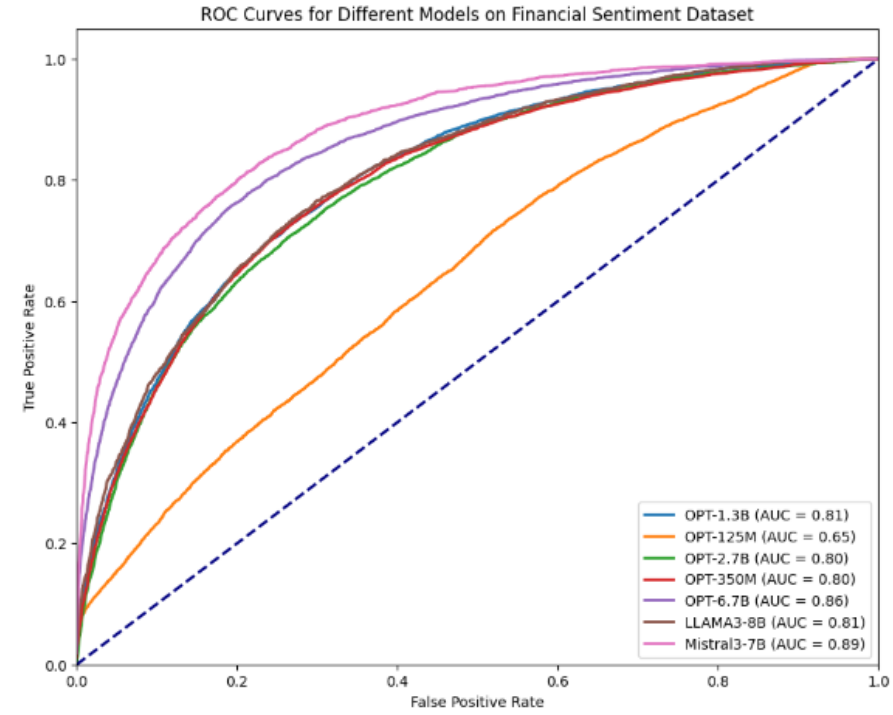


Figure 4. Receiver operating curve of different models on Financial Sentiment classification dataset.

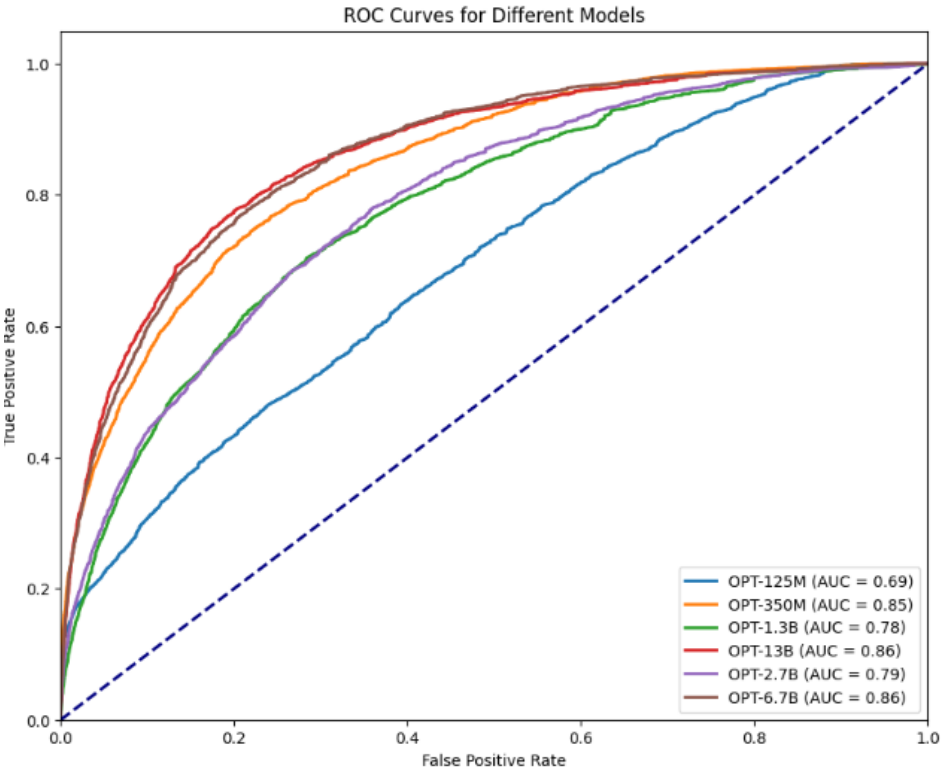


Figure 5. Receiver operating curve of different models on Dailydialog classification dataset.

Analyzing performance of different models on the Political News dataset: We analyzed the performance of different models on the causal prediction task using the political news dataset. The ROC curves in Figure 6 show the classification performance, with AUC values lower than those from other datasets, ranging from 0.61 to 0.70. The

Mistral model achieves the highest AUC of 0.70, indicating relatively better accuracy compared to other models. OPT-6.7B and OPT-13B follow closely, both with an AUC of 0.68. The OPT-350M and OPT-2.7B models score AUCs of 0.67, placing them in the middle. The lowest-performing models are LLAMA, OPT-1.3B, and OPT-125M, with AUC values of 0.61, 0.65, and 0.62, respectively.

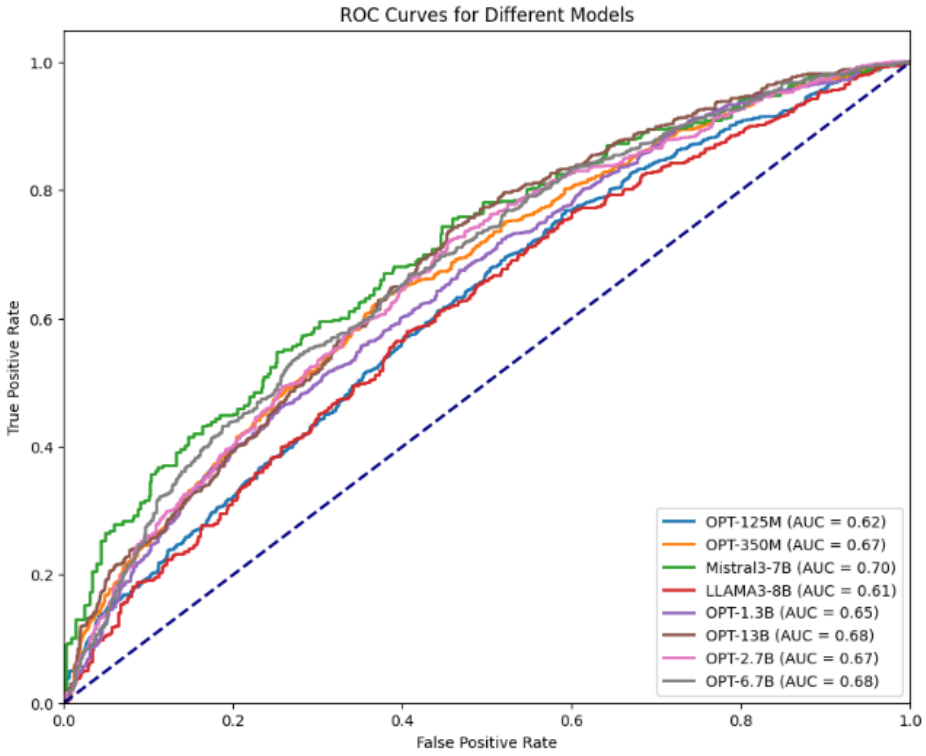


Figure 6. Receiver operating curve (ROC) for different models on the Political News dataset.

Analyzing performance of different models on the Agnews News dataset: Finally, we analyze the prediction errors in the topic classification task using the Agnews dataset. The ROC curves in Figure 7 show the performance of various models on this dataset, with AUC values ranging from 0.96 to 0.98. The OPT-2.7B and OPT-6.7B models achieve the highest performance with an AUC of 0.98, indicating their strong ability to distinguish between classes with minimal false positives. Close behind are the OPT-350M and OPT-1.3B models, each scoring an AUC of 0.97, showing slightly lower but still high classification accuracy. The smallest model, OPT-125M, attains an AUC of 0.96, which, while slightly lower, still reflects strong performance. Overall, these results indicate that all models perform exceptionally well on the Agnews dataset, with minimal variation in AUC values across different model sizes, suggesting that this classification task is relatively easier for all models.

Performance variations with different training set sizes for different datasets: The next set of experiments evaluates how model performance varies with different training set sizes. Figure 8 shows the F1 score for various percentages of training data. Topic classification on the Agnews dataset is the simplest task, reaching optimal performance with less data. Emotion classification starts with poor performance on small datasets but improves significantly as the dataset grows, leveling off around 50%. The Financial Sentiment dataset shows steady improvement with data size, achieving high F1 scores with larger training sets. DailyDialog, however, exhibits the slowest performance growth, with gradual F1 score increases as the training size expands. Overall, model performance improves as the training set size increases up to about 60% of the training set, after which further gains become minimal.

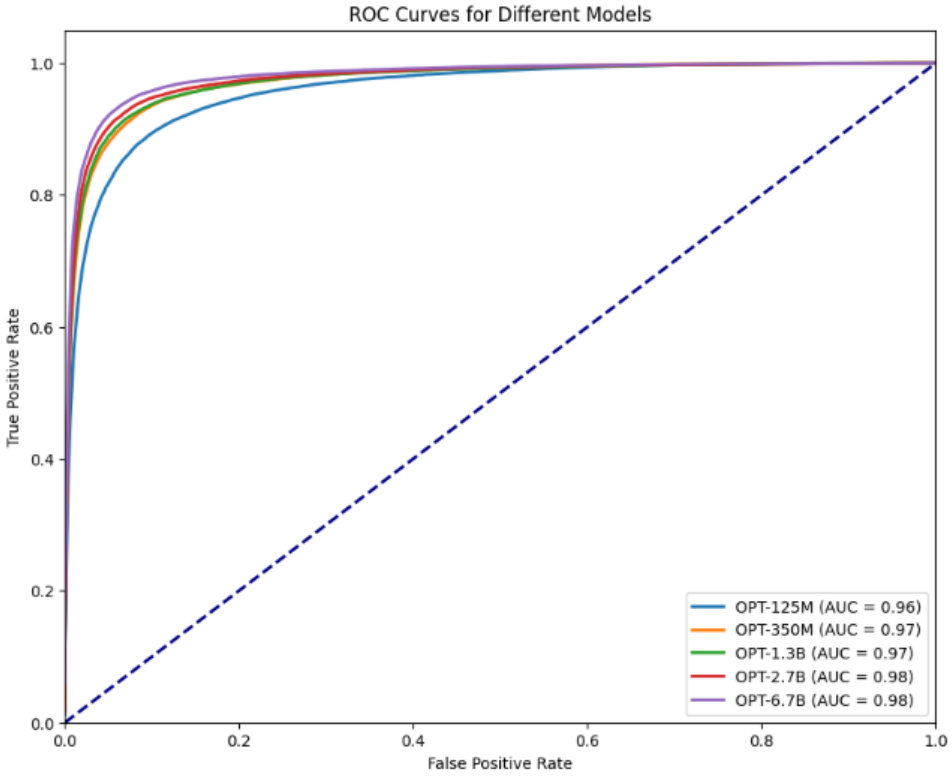


Figure 7. Receiver operating curve (ROC) for different models on the Agnews News dataset.

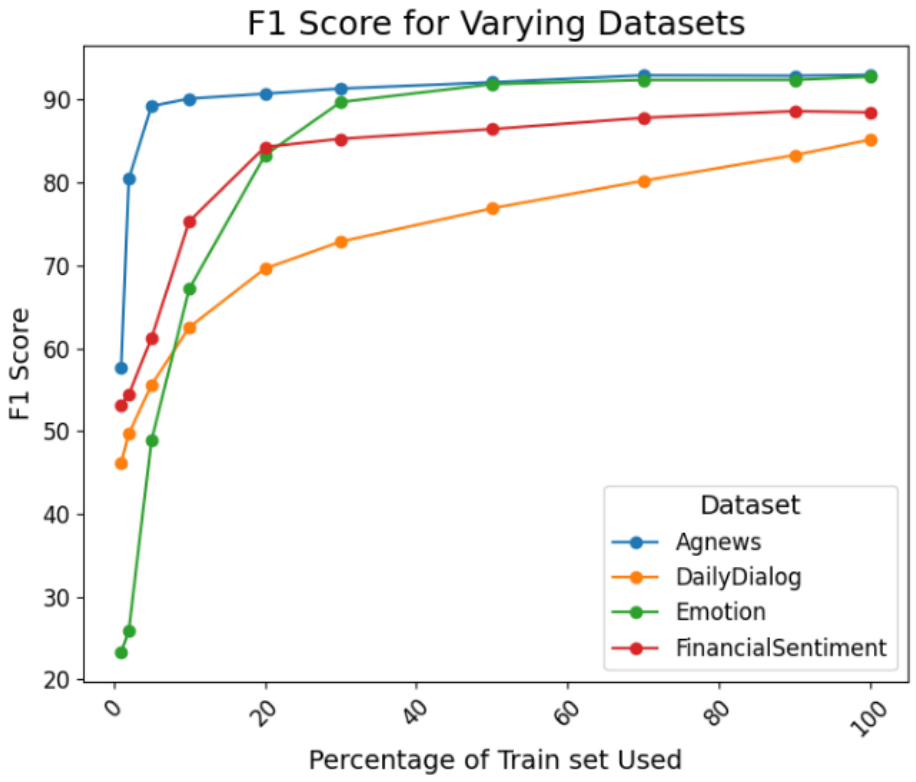


Figure 8. Performance variations for different train sizes on different datasets.

Performance variations with different training set sizes for different models: We conducted experiments to compare the performance of different models across all datasets with varying training set sizes. Figure 9 illustrates how model performance changes with

different training set sizes, similar to the patterns seen in Figure 8. Initially, all models show low performance with small training sizes, but there is a substantial improvement up to about 50% of the training set, after which further increases in training set size have a smaller effect. BERT and RoBERTa show rapid early improvement, with BERT achieving high F1 scores quickly, while RoBERTa steadily improves. Both plateau around a 90% F1 score after 60% of the training set. DistilRoBERTa starts slower but levels with others after 20% of the training set, though it remains slightly behind BERT and RoBERTa. OPT-125M shows slower initial gains but improves significantly after 30% of the training set, while OPT-350M improves consistently across all training sizes. DistilBERT shows more gradual and linear improvement, plateauing earlier than the others.

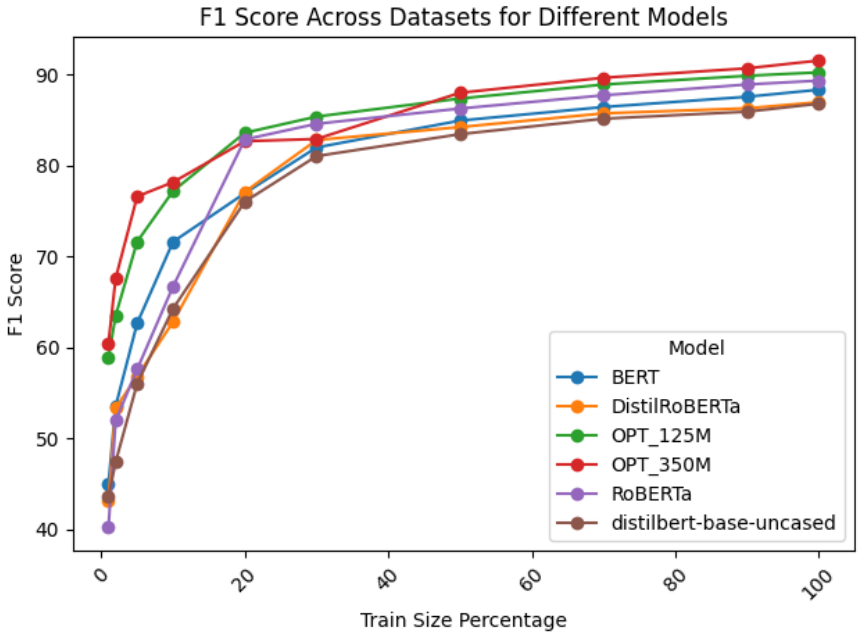


Figure 9. Performance variations for different models for increasing training size.

Adaptation Method: We conduct further experiments to assess the impact of an adaptation method on classification performance and detailed our findings in Table 2. We compare the full fine-tuning approach which involves fine-tuning in 16-bit precision with other parameter-efficient adaptation methods such as fine-tuning with quantized weights in 4-bit and 8-bit precision, prompt tuning, prefix tuning, and zero-shot and few-shot prompting strategies. Our findings show that fine-tuning an LLM with the full fine-tuning approach using 16-bit precision achieves the best results with an average accuracy of 85.65% and an F1 score of 85.93%. This indicates that maintaining high-bit precision during adaptation remains the most effective method for adapting an LLM for text classification. However, it is important to note that high-precision fine-tuning requires considerable computational resources which may not be readily accessible, especially in low-resource settings. Additionally, training LLMs in full precision takes significantly longer.

Table 2. Average performance of adaptation methods.

Dataset	Accuracy	F1 Score	Precision	Recall
Zero-Shot	63.91	59.61	63.17	64.63
Few-Shot	68.34	61.52	65.37	64.39
Prompt Tuning	73.77	77.55	74.10	77.07
Fine-tune-4bits	71.11	74.68	71.55	80.08
Fine-tune-8bits	74.94	76.58	75.40	82.97
Prefix Tuning	82.53	84.44	81.80	83.34
Fine-tune-16bits	**85.65**	**85.93**	**85.64**	**88.62**

Our next experiments involve fine-tuning LLMs with quantized weights in both 8-bit and 4-bit precision. We tested two methods [21,63] that are theoretically shown to have no performance degradation in text generation tasks. The results showed that reducing the bit precision to 8 bits during fine-tuning resulted in a moderate accuracy of 74.94% and an F1 score of 76.58%. Further reducing the precision to 4 bits led to an accuracy of 71.11% and an F1 score of 74.67%. These findings challenge the notion that quantization methods do not impact language model performance, as there was nearly a 10-point drop in accuracy when the fine-tuning precision was halved from 16 to 8 bits. However, the performance decline was less pronounced when decreasing the precision from 8 to 4 bits with only a three-point dip in accuracy.

We experimented with prompting strategies considering both discrete and continuous prompts. Among the continuous prompt tuning approaches, prefix tuning achieved better performance compared to prompt tuning with an accuracy of 82.53% versus 73.76% and an F1 score of 84.44% versus 77.55%. However, both continuous prompt tuning strategies are still outperformed by full 16-bit fine-tuning, though not by large margins. The worst performance among the adaptation techniques was observed with discrete prompts, particularly in the zero-shot setting, which had an average accuracy of 63.90% and an F1 score of 59.61%. The performance of discrete prompting improved with the addition of a few demonstration examples, increasing the accuracy to 68.33%. Despite this improvement, the accuracy remains significantly lower than that achieved by full fine-tuning and continuous prompt tuning. The trade-off with continuous prompting and fine-tuning approaches is that, while discrete prompts perform worse than continuous prompts, they are more interpretable because they are written in human language. Additionally, discrete prompts require no training data or parameter updates in a zero-shot setting where a task description written in human language is sufficient.

In-context Learning and Prompting: In addition to fine-tuning, we evaluated in-context learning using both zero-shot and few-shot prompting techniques. Figure 10 shows a box plot that illustrates performance variations across different methods of retrieving demonstration examples, as well as zero-shot prompting where no examples are provided. The best performance in in-context learning was achieved with few-shot prompting when the selected demonstration examples were similar to the test example in the embedding space. Randomly selecting examples performed almost as poorly as zero-shot prompting. A Friedman test confirmed that these differences were statistically significant, with a test statistic of 226.89 and a p-value of 5.39×10^{-50}.

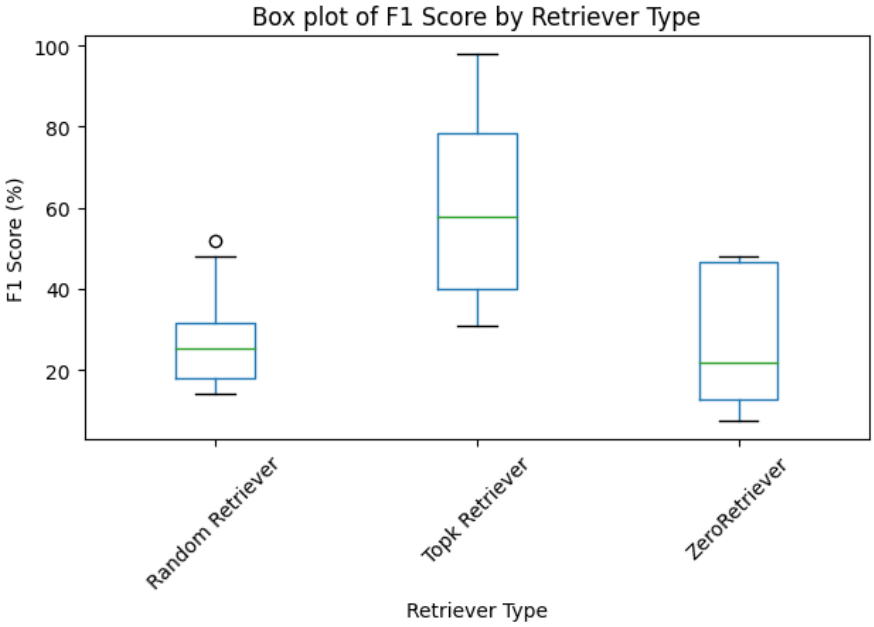


Figure 10. Choice of retrieval type in in-context learning.

Figure 11 illustrates how the number of demonstration examples affects the F1 score. The key observation is that performance improves slightly as the number of demonstrations increases, peaking around seven examples, after which performance starts to decline. We hypothesize that adding too many demonstration examples may confuse the model, making it harder to determine the correct class label for the text.

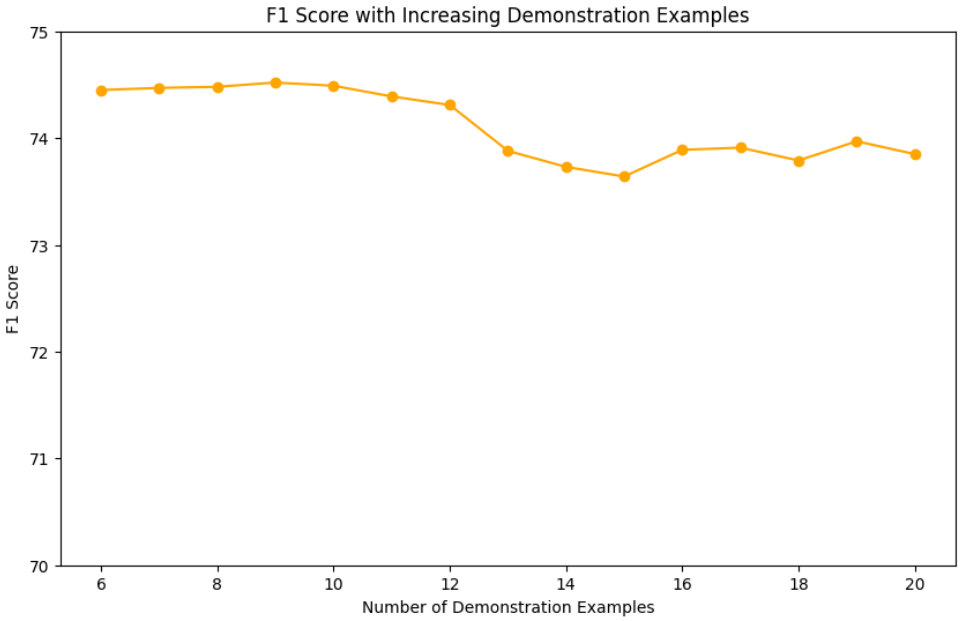


Figure 11. Impact of number of demonstration examples.

Discriminative Versus Generative Models: Figure 12 provides a comparative evaluation of discriminative and generative pre-trained LLMs across several datasets, using F1 score as the metric. Discriminative models generally outperform generative models, with discriminative models achieving an average F1 score of 76.79%, compared to 73.05% for generative models. The performance gap between the two is minimal on datasets such as Agnews, Financial Sentiment, IMDB, and Emotion, with differences typically within 3 percentage points. For example, on the Agnews dataset, discriminative models achieve an F1 score of 92.19% compared to 90.44% for generative models; on IMDB, the scores are 88.25% versus 87.34%; and on the Emotion dataset, discriminative models score 69.77% compared to 69.55% for generative models. However, larger performance gaps are observed on the DailyDialog and Political News datasets. On DailyDialog, discriminative models achieve an average F1 score of 60.41%, compared to 59.81% for generative models, and on Political News, discriminative models achieve 72.38% compared to 61.57% for generative models. Despite these differences, a Wilcoxon signed-ranks test revealed that the variations in performance across datasets were not statistically significant at the 5% level (test statistic = 662.0, p-value = 0.092).

Number of Parameters: Figure 13 displays two line graphs illustrating the effect of model parameters on classification performance, measured by the F1 score. In Figure 13a, which compares various model families, there is a clear fluctuation in performance as the number of parameters increases. While models with up to 1 billion parameters generally show improved performance, results become inconsistent beyond this point, with some models performing worse despite larger parameter sizes. For example, models with 2.7 billion parameters perform worse than some smaller models, whereas those with 6.7 billion parameters show a significant improvement. In Figure 13b, which focuses on the OPT model family, the trend is more pronounced. Performance increases initially up to 1 billion parameters, and then sharply drops at 2.7 billion, followed by a strong recovery as parameters approach 6.7 billion. This indicates that although larger models often perform better, the relationship between model size and classification performance is not strictly

linear. Factors like architecture and optimization methods influence performance. We should, however, note that there may be bias due to the precision used to train bigger models, since models that could not be tuned in 16-bit were either tuned in 8-bit or 4-bit.

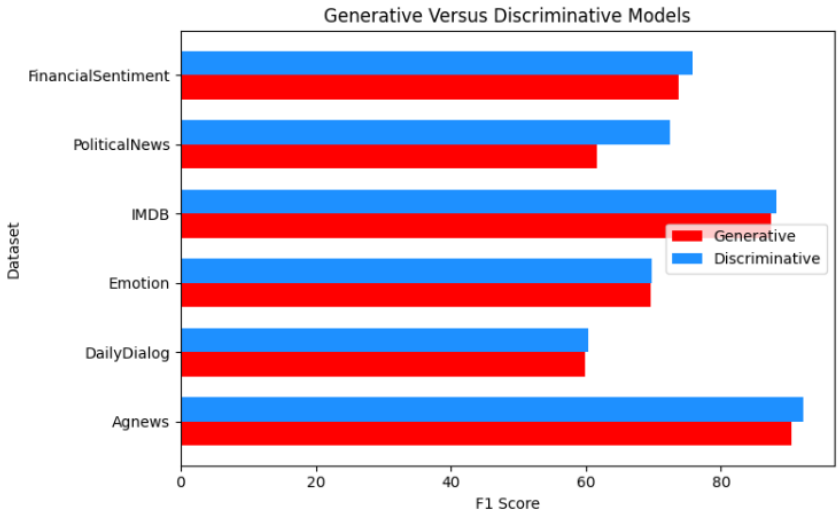
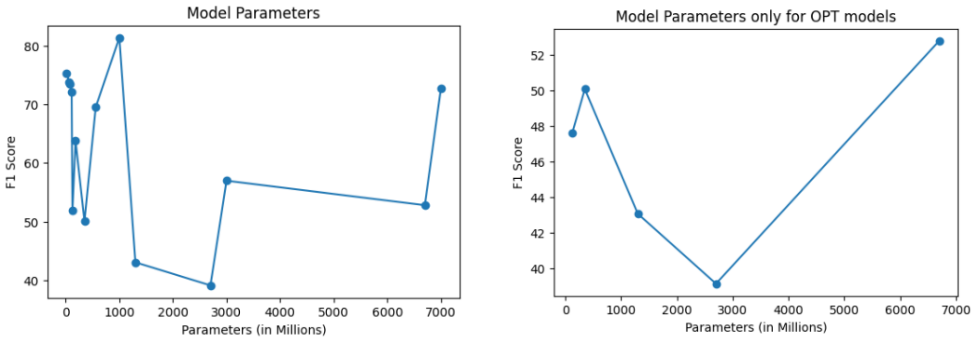


Figure 12. Average performance of generative models versus discriminative models.



(**a**) (**b**)

Figure 13. Comparison of performance variations with varying number of parameters for all models and same family of models (OPT). (**a**) How model parameters vary with classification performance for all datasets; (**b**) how model parameters vary with classification performance for for same family of models (OPT).

Model Choice: We investigated how different models affect text classification performance, as shown in Figure 14. The box plot illustrates the distribution of F1 scores for various models, with Mistral-7B and Zephyr-7B emerging as the top performers. These models consistently had high F1 scores with low variability, likely due to additional instruction tuning and preference alignment during pre-training. Other models, such as ALBERT, BERT, and RoBERTa, also showed strong performance, with BERT and ALBERT achieving high median F1 scores. Interestingly, smaller models like BERT outperformed larger ones like Falcon-7B and BLOOM-7B, suggesting that pre-training objectives and model architecture may be more important than size. Lower-performing models, including OPT-125M, LLAMA-7B, and DistilBERT, showed greater variability and lower median F1 scores. The Friedman test confirmed that these performance differences are statistically significant at the 5% level, with a test statistic of 28.72 and a p-value of 0.012.

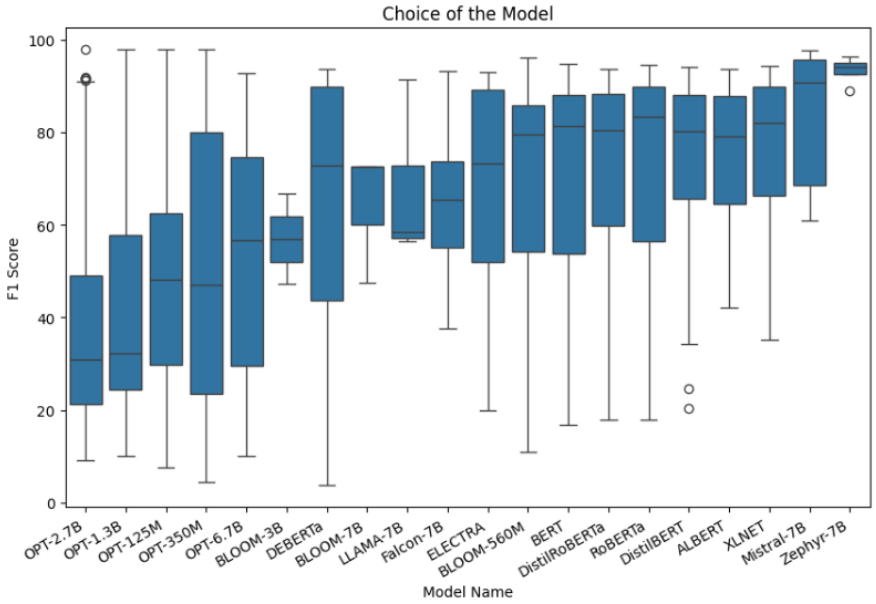


Figure 14. Performance variations of different models.

Does Low-Rank Adaptation (LoRA) Affect Classification Performance?: LoRA effectively fine-tunes LLMs by adjusting low-rank weight matrices instead of the original pre-trained parameters, significantly reducing the number of trainable parameters. We conducted experiments to assess its impact on text classification performance. Figure 15 is a bar graph illustrating the impact of LoRA on classification F1 scores by comparing datasets where adaptation was applied ("yes") versus those where it was not ("no").

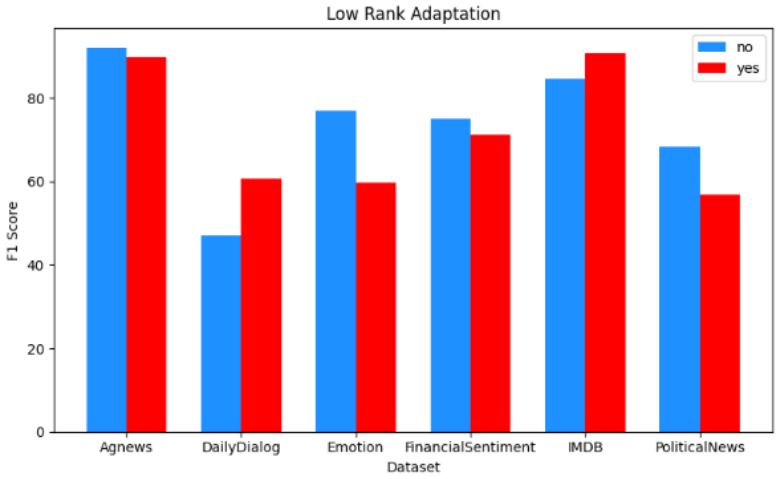
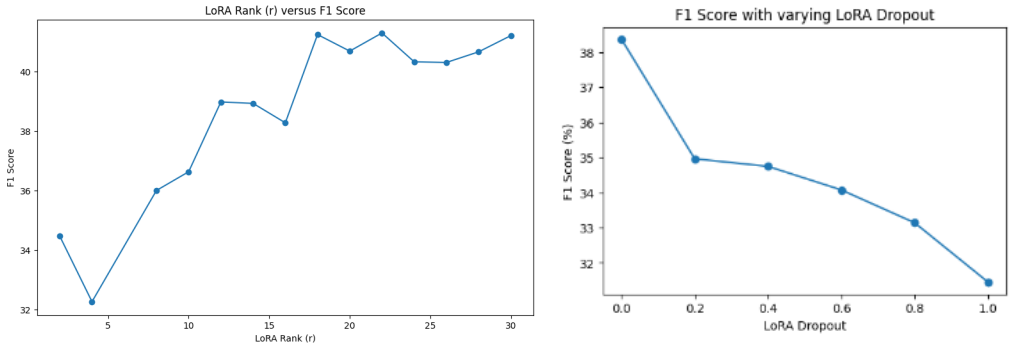


Figure 15. Performance variations of different datasets for LoRA.

The experimental results show mixed effects. For instance, on Agnews, the F1 score declines from 92.35% to 88.96%; on DailyDialog, it falls from 70.35% to 67.87%; on the Emotion dataset, the F1 score drops significantly from 85.66% to 72.00%; on the Financial Sentiment dataset, the F1 score decreases from 87.07% to 81.72%; and on the Political News dataset, the F1 score drops markedly from 67.80% to 59.75%. However, on IMDB, the F1 score improves from 82.08% to 88.58% with LoRA. Overall, LoRA results in a performance decline in average F1 score from 81.00% to 77.80%, and the performance differences are statistically significant with the Wilcoxon signed-rank test for a test statistic of 0.0 and p-values of 2.90×10^{-33}. This suggests that while LoRA can reduce parameter count and benefit certain datasets, its effectiveness is not universally guaranteed and depends on dataset-specific characteristics in text classification problems.

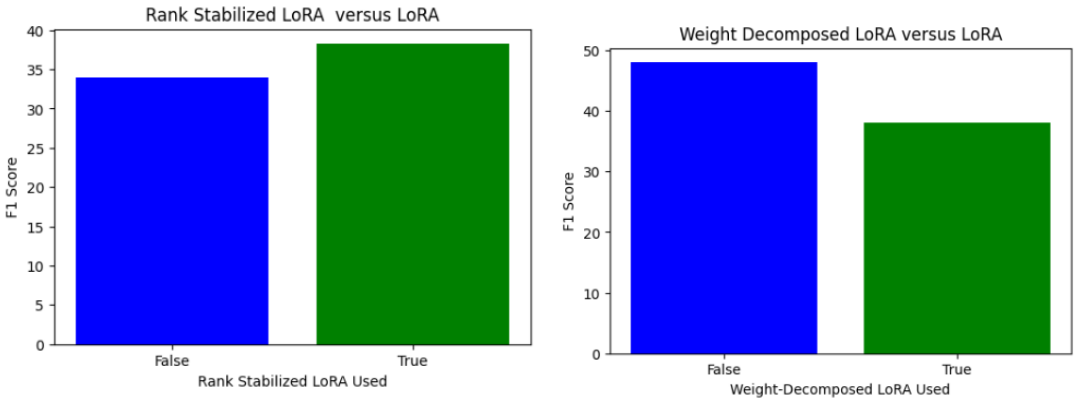
We conducted further experiments to investigate the effects of varying LoRA dropout values (dropout probabilities for LoRA layers) and the LoRA attention dimension (rank *r*) on model performance, as displayed in Figure 16. Figure 16a shows how the F1 score changes with different LoRA rank values, steadily increasing and peaking at rank 20 with an F1 score of approximately 48. Beyond this point, the score remains stable with minor fluctuations at higher ranks. Figure 16b demonstrates the effect of increasing dropout rates on the F1 score. As the dropout rate increases, the F1 score consistently decreases, starting from around 38 at a 0.0 dropout rate and falling sharply to 32 at a dropout rate of 1.0, indicating that higher dropout rates negatively impact performance.



(**a**) (**b**)

Figure 16. Comparison of performance variations between rank-stabilized LoRA and weight-decomposed LoRA values. (**a**) Performance variations of LoRA dropout values; (**b**) performance variations of LoRA rank (r).

We further examined two approaches: rank-stabilized LoRA [75], which adjusts the adapter scaling factor based on the square root of the rank, and weight-decomposed LoRA (DoRA) [76], which splits LoRA weight updates into two components: magnitude and direction. In DoRA, the direction is controlled by standard LoRA, while the magnitude is handled by a separate learnable parameter. The results in Figure 17 show that rank-stabilized LoRA leads to a slight improvement in F1 score, as illustrated in Figure 17a, where the F1 score increases from 35 to 38 when rank-stabilized LoRA is applied. In contrast, Figure 17b shows that weight-decomposed LoRA decreases the F1 score from 45 to 35. In these figures, "True" indicates that the method was applied, while "False" refers to using standard LoRA.



(**a**) (**b**)

Figure 17. Comparison of performance variations between rank-stabilized LoRA and weight-decomposed LoRA values. (**a**) Performance variations of rank-stabilized LoRA values; (**b**) performance variations of weight-decomposed LoRA values.

Effect of quantization: We examine the impact of quantization by assessing the performance of various models within the OPT family, ranging from 125 million to 6.7 billion parameters, using three precision levels: 4-bit, 8-bit, and full 16-bit fine-tuning. For smaller models, all three quantization techniques were used, while larger models required quantization due to hardware limitations, preventing full 16-bit fine-tuning. As shown in Figure 18, 16-bit precision consistently yielded the best performance across all datasets. However, performance decreased with lower precision, with the most significant F1 score drop occurring at 4-bit quantization. In most cases, 8-bit precision performed better than 4-bit, except for the OPT-2.7B model. The differences in performance across bit precision levels were statistically significant at the 5% level, confirmed by the Friedman test with a test statistic of 27.95 and a p-value of 8.51×10^{-7}.

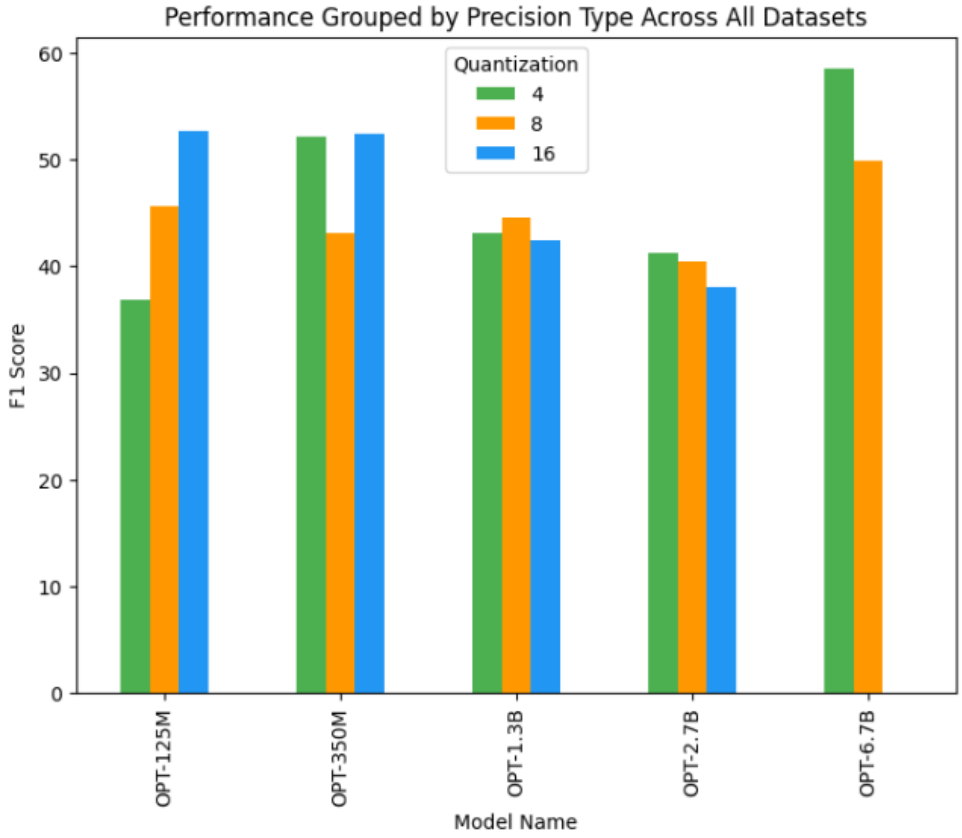
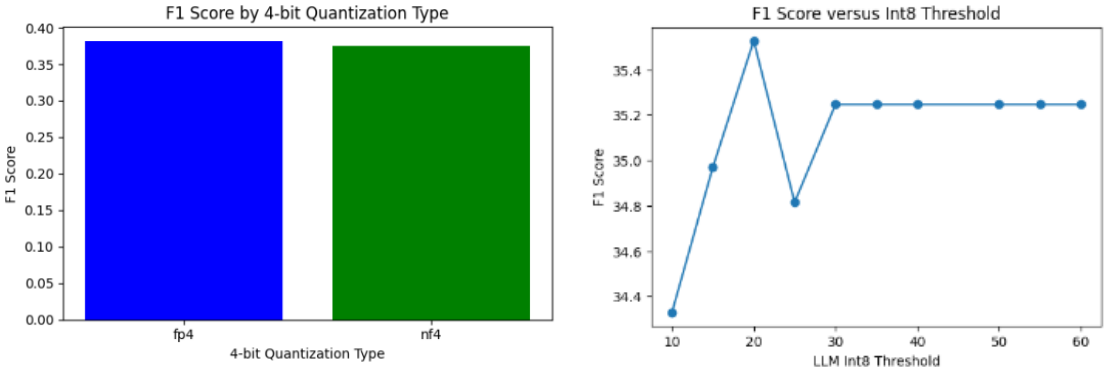


Figure 18. Effect of quantization for the OPT model family across all datasets.

Figure 19 illustrates the performance variations based on different 4-bit quantization types and the Int8 threshold. The Int8 threshold is used for outlier detection, where hidden state values exceeding this limit are classified as outliers and processed in 16 bits. The 4-bit quantization type refers to the specific data format used for 4-bit quantization. Figure 19a compares different quantization types, showing that using "nf4" slightly outperforms "fp4", achieving a higher F1 score. Figure 19b demonstrates performance variations across different Int8 thresholds. The F1 score increases sharply at a threshold of 20, reaching a peak around 35.4, and then remains stable as the threshold increases further. This suggests that while adjusting the Int8 threshold can improve performance, the improvements plateau beyond a certain point.



(**a**) (**b**)

Figure 19. Graphs of effect of variations in 4-bit quantization type and Int8 threshold on classification performance. (**a**) Average performance of different 4-bit quantization types; (**b**) performance variation across Int8 thresholds.

Table 3 provides a detailed analysis of various models and ensemble methods based on accuracy. Human performance is the highest across all metrics, with values around 93.32, showcasing superior performance. Among the models, Mistral-45B-DPO achieves the best results with an accuracy of 81.00, precision of 82.15, recall of 81.00, and an F1 score of 80.58. Other models like LLAMA2-70B and LLAMA3-8B have similar but slightly lower performance, while mistral-45B performs better than these with an accuracy of 77.51. Ensemble methods such as DawidSkene and MACE show competitive results with accuracy and F1 scores indicating their effectiveness in combining multiple models. In contrast, MajorityVote and WAWA ensembles perform poorly, with significantly lower accuracy and other metrics. This comparison highlights the varying performance levels of different approaches, emphasizing the challenge for automated methods to match human accuracy in this context.

Table 3. Average performance across datasets for different data generators (teacher models) and human-labeled data.

Data Generator	AG	Financial	Emotion	DialyDialog
MajorityVote	23.19	49.99	27.40	37.75
WAWA	23.50	51.40	27.20	29.46
GLAD	23.50	52.44	27.20	29.46
LLAMA3-8B	74.70	81.08	57.99	52.96
LLAMA2-70B	75.46	77.74	58.62	60.06
MACE	76.34	**83.46**	59.37	61.37
DawidSkene(DS)	77.92	82.01	**60.26**	52.96
OneCoinDS	76.27	76.00	58.78	52.91
Mistral-45B	77.51	81.59	59.63	52.91
Mistral-45B-DPO	**81.00**	82.68	58.45	58.34
Human	**93.32**	**90.34**	**90.25**	**90.35**

Weakly Supervised Text Classification: In addition to experimenting with the efficient adaptation techniques previously discussed, we also explored weakly supervised learning for text classification. This involves fine-tuning LLMs with synthetic labels generated by other LLMs, which eliminates the need for a labor-intensive and costly labeling process.

Performance of Teacher Models: Table 4 presents the zero-shot accuracy results for teacher models (models generating the training data) across various classification datasets: Agnews, DailyDialog, Financial Sentiment, Emotion, and DailyDialog. The data generator column represent the model used to create the training dataset for training other LLMs. In this context, "Human" indicates that the LLM was trained on human-labeled data, but it does not reflect human performance in these tasks. The model Mistral-45B-DPO performed best, achieving an accuracy of 81% on the Agnews dataset, 81.29% on the DailyDialog dataset, and 82.68% on the Financial Sentiment dataset. For the Emotion dataset, models

achieved similar accuracies around 58%, with LLAMA2 (70 billion parameters) achieving the highest at 58.62%, and 60.06% on the DailyDialog dataset. Ensembling methods sometimes improved results but other times worsened them. Among these methods, the Dawid–Skene ensemble consistently outperformed other ensemble techniques on most tested datasets.

Table 4. Average performance across datasets for different student models.

Model	AG	Financial	Emotion	DailyDialog
GLAD	23.91	62.02	27.50	18.77
WAWA	23.72	60.76	27.50	19.24
MajorityVote	23.30	59.25	26.85	60.82
LLAMA3-8B	74.26	81.79	58.55	43.94
LLAMA2-70B	75.12	77.39	58.30	65.71
MACE	75.91	**84.38**	60.60	65.88
DawidSkene(DS)	77.45	80.32	60.60	43.59
OneCoinDS	75.93	83.71	59.00	44.24
Mistral-45B	76.74	82.12	60.95	43.53
Mistral-45B-DPO	**80.49**	83.92	59.70	**69.94**
Human	75.46	**90.34**	**90.25**	**90.35**

Performance of Student Models: We used the weak labels generated by teacher models, whose performance is illustrated in Table 3, to fine-tune other student models. Specifically, we fine-tuned the RoBERTa model, which is significantly smaller than the teacher models used to generate the data (125 million parameters versus 8 billion parameters for the smallest teacher model and 70 billion parameters for the largest). Overall, fine-tuning with human-labeled data consistently achieved the highest accuracy across all datasets, with results ranging from 86.24% to 93.32%. The performance of student models is highly correlated with that of teacher models despite the student model RoBERTa having significantly fewer parameters than the teacher models, with a Pearson correlation coefficient of 0.986 and a statistically significant p-value of 2.64×10^{-8}. The best performance is achieved by fine-tuning RoBERTa on data generated by Mistral-45B-DPO, with an accuracy of 80.49% on Agnews, 83.93% on Financial Sentiment, and 69.94% on DailyDialog. For the Emotion classification dataset, Mistral-45B achieved the highest accuracy of 60.95%.

How weight decay influences model performance: Figure 20 shows the effect of the weight decay hyperparameter on classification performance. The results indicate that most weight decay values result in a similar area under the curve (AUC), typically around 0.87 or 0.88. However, slight differences are observed: weight decay values of 0.03, 0.05, and 0.1 achieve the highest AUC of 0.88, while other values, including 0.0, 0.01, and 0.02, and higher values like 0.3, 0.5, and 1.0 maintain an AUC of 0.87. This suggests that moderate weight decay (between 0.03 and 0.1) slightly improves performance, but further increases in weight decay do not provide additional benefit, with the AUC remaining stable at 0.87.

How number of epochs varies with classification performance: Figure 21 illustrates how the F1 score varies with the number of epochs. The F1 score initially increases from 71.70 at 5 epochs to 80.66 at 20 epochs. At 25 epochs, it drops to 72.49, indicating potential overfitting. The peak F1 score of 80.66 occurs at 20 epochs, marking optimal model performance. Beyond this point, the F1 score fluctuates between 73 and 78 from 30 to 50 epochs, suggesting diminishing returns with additional training. This indicates that while increasing epochs can improve the F1 score up to a certain point, excessive training may lead to overfitting or performance stagnation.

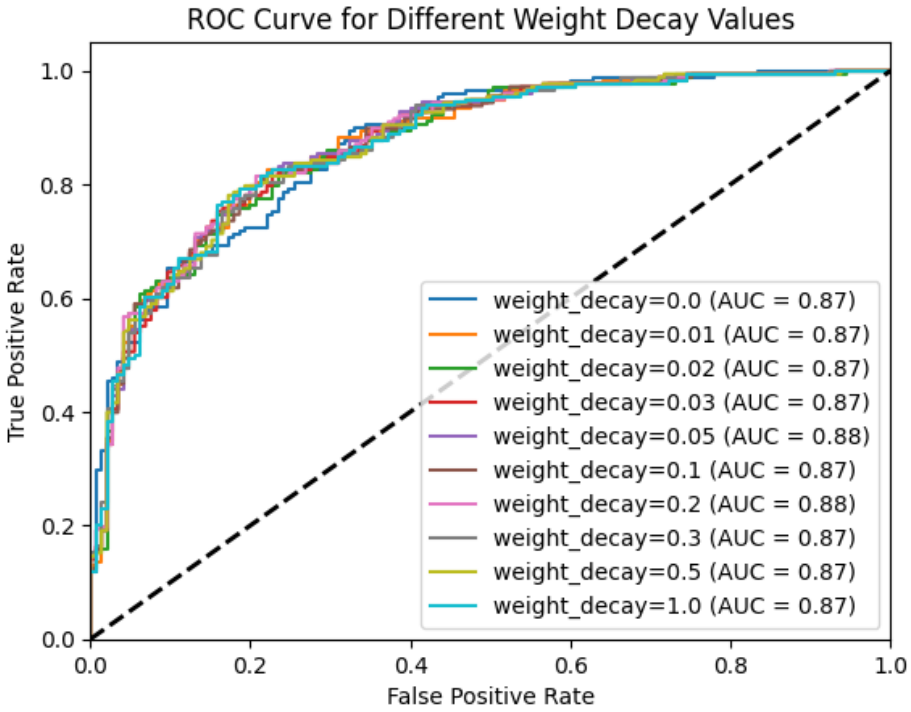


Figure 20. Performance variation across different weight decay values.

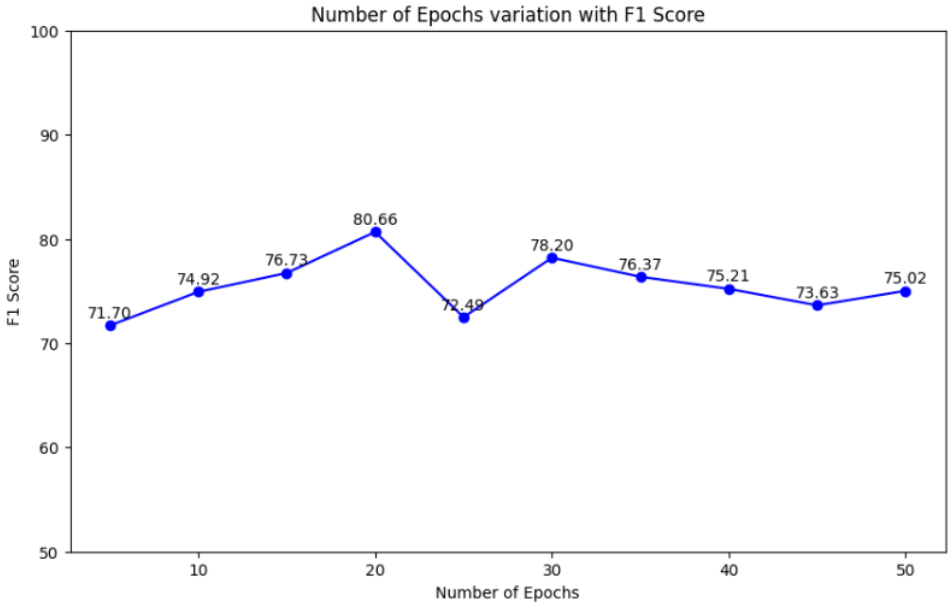


Figure 21. Performance variation across different number of epochs.

Influence of maximum gradient norm varies on performance: Figure 22 shows the impact of varying the maximum gradient norm on the F1 score. Initially, the F1 score is 72.21 at a gradient norm of 0, but it slightly decreases to 71.58 at a norm of 2. As the gradient norm increases, the F1 score improves, reaching 77.09 at norms of 4 and 6, where it stabilizes. The highest F1 score of 78.41 occurs at a gradient norm of 8. However, beyond this point, the F1 score decreases slightly to 74.83 at a norm of 10. Overall, the graph suggests that adjusting the maximum gradient norm can enhance the F1 score, but increasing it beyond a certain level may result in diminishing returns or slight performance drops.

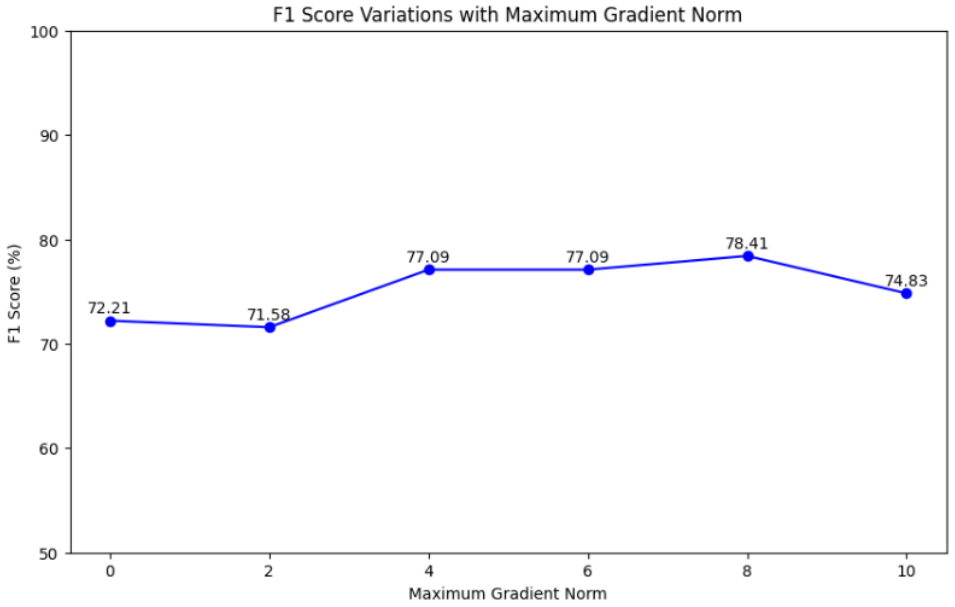


Figure 22. Performance variation across maximum gradient norms.

How learning rate varies with classification performance: The line graph in Figure 23 illustrates the relationship between learning rate and classification performance measured by F1 score. The results show that the F1 score fluctuates significantly as the learning rate changes. The model achieves its highest F1 score, peaking at about 80, when the learning rate is near 0.0001. However, beyond this point, the F1 score declines sharply, showing instability between learning rates of 0.002 and 0.004, where the F1 score fluctuates between 40 and 60. After 0.004, the F1 score stabilizes around 40, with no significant improvement as the learning rate increases further.

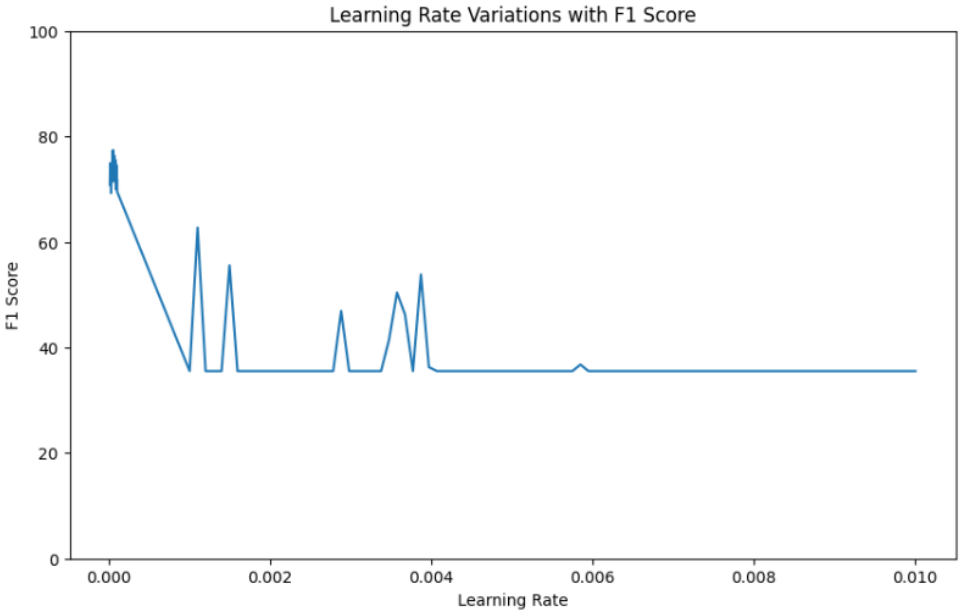


Figure 23. Performance variation across learning rates.

5. Conclusions

In this research, we investigated and evaluated various adaptation strategies for LLMs and their impact in classification performance against datasets, including prompting and parameter-efficient fine-tuning methods such as quantization, low-rank adaptation, prompt

tuning, weak supervision and prefix tuning, along with their associated hyperparameters. We also examined the impact of pre-training objectives on text classification performance. Our comparative analysis has shown that discriminatively pre-trained models outperform generatively pre-trained models in classification tasks. Among the adaptation methods, fine-tuning at full 16-bit precision yielded the best results, with an average accuracy of 85.65% and an average F1 score of 85.93%, while discrete prompting performed the worst, with an average accuracy of 63.91% and F1 score of 56.61%. Additionally, we found that the relationship between model size and classification performance is not strictly linear: larger LLMs do not always outperform smaller ones, as performance is influenced by factors such as model family, pre-training objectives, and adaptation techniques. For fine-tuning with quantization, 8-bit fine-tuning generally outperformed 4-bit fine-tuning (76.58% versus 74.68% in terms of F1 score) but the gap was not as large that of 16-bit and 8-bit. We also found that few-shot prompting significantly outperforms zero-shot prompting. Among the selection methods, semantic similarity retrieval proved to be the most effective, while randomly selecting demonstrations yielded results nearly equivalent to zero-shot prompting. In our LoRA experiments, we observed that applying LoRA significantly reduces the performance compared to full fine-tuning. Additionally, increasing LoRA dropout rates negatively impacted classification performance, and there was little difference in performance between standard LoRA and its variants, such as weight-decomposed LoRA and rank-stabilized LoRA. Fine-tuning with the `nf4` 4-bit data type outperformed the `fp4` data type. Other hyperparameters we studied included weight decay, where different values had no significant effect on performance. Increasing the number of epochs improved accuracy up to a certain point, after which performance declined and then plateaued. Lower learning rates produced better results, with further increases in learning rates showing diminishing returns after a certain point. These findings emphasize the need to choose carefully the appropriate model and adaptation method based on the specific task and computational limitations. For instance, quantization allows users to experiment with larger language models, though it may slightly reduce accuracy. The goal is to find a balance between accuracy, efficiency, and model complexity and cost. Detailed guidance provided by our results is presented in Section 4 under specific analysis goals.

Author Contributions: Conceptualization, P.T.; Methodology, P.T.; Validation, R.M.; Writing—original draft, P.T.; Writing—review & editing, R.M.; Supervision, R.M.; Funding acquisition, R.. All authors have read and agreed to the published version of the manuscript.

Funding: This publication has emanated from research conducted with the financial support of Science Foundation Ireland under grant number 18/CRT/6222.

Data Availability Statement: The datasets used in this study are publicly accessible on Huggingface https://huggingface.co/dataset (accessed on 8 August 2024) and are thoroughly described in the data Section 3.6.

Acknowledgments: We thank the Insight SFI Research Center for Data Analytics at University College Cork for providing access to the GPU used for the experiments. This publication has emanated from research conducted with the financial support of Science Foundation Ireland under grant number 18/CRT/6222.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kowsari, K.; Jafari Meimandi, K.; Heidarysafa, M.; Mendu, S.; Barnes, L.; Brown, D. Text classification algorithms: A survey. *Information* **2019**, *10*, 150. [CrossRef]
2. Medhat, W.; Hassan, A.; Korashy, H. Sentiment analysis algorithms and applications: A survey. *Ain Shams Eng. J.* **2014**, *5*, 1093–1113. [CrossRef]
3. Trust, P.; Zahran, A.; Minghim, R. Understanding the influence of news on society decision making: application to economic policy uncertainty. *Neural Comput. Appl.* **2023**, *35*, 14929–14945. [CrossRef]

4. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In 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), Minneapolis, MN, USA, 2–7 June 2019; pp. 4171–4186. [CrossRef]
5. Li, Q.; Peng, H.; Li, J.; Xia, C.; Yang, R.; Sun, L.; Yu, P.S.; He, L. A survey on text classification: From shallow to deep learning. *arXiv* **2020**, arXiv:2008.00364.
6. Weiss, K.; Khoshgoftaar, T.M.; Wang, D. A survey of transfer learning. *J. Big Data* **2016**, *3*, 1–40. [CrossRef]
7. Kotsiantis, S.B.; Zaharakis, I.; Pintelas, P. Supervised machine learning: A review of classification techniques. *Emerg. Artif. Intell. Appl. Comput. Eng.* **2007**, *160*, 3–24.
8. Zhang, Y.; Nivre, J. Transition-based dependency parsing with rich non-local features. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Portland, OR, USA, 19–24 June 2011; pp. 188–193.
9. Guyon, I.; Weston, J.; Barnhill, S.; Vapnik, V. Gene selection for cancer classification using support vector machines. *Mach. Learn.* **2002**, *46*, 389–422. [CrossRef]
10. Bengio, Y.; Ducharme, R.; Vincent, P. A neural probabilistic language model. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2000; Volume 13.
11. Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; Kuksa, P. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.* **2011**, *12*, 2493–2537.
12. Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I. *Improving Language Understanding by Generative Pre-Training*; OpenAI: San Francisco, CA, USA, 2018.
13. Zhuang, L.; Wayne, L.; Ya, S.; Jun, Z. A Robustly Optimized BERT Pre-training Approach with Post-training. In Proceedings of the 20th Chinese National Conference on Computational Linguistics, Huhhot, China, 13–15 August 2021; pp. 1218–1227.
14. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*; Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H., Eds.; Curran Associates, Inc.: New York, NY, USA, 2020; Volume 33, pp. 1877–1901.
15. Scao, T.L.; Fan, A.; Akiki, C.; Pavlick, E.; Ilić, S.; Hesslow, D.; Castagné, R.; Luccioni, A.S.; Yvon, F.; Gallé, M.; et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv* **2022**, arXiv:2211.05100.
16. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2020; Volume 33, pp. 1877–1901.
17. Liu, P.; Yuan, W.; Fu, J.; Jiang, Z.; Hayashi, H.; Neubig, G. Pre-Train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *ACM Comput. Surv.* **2023**, *55*, 195. [CrossRef]
18. Li, X.L.; Liang, P. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Online, 1–6 August 2021; Zong, C., Xia, F., Li, W., Navigli, R., Eds.; pp. 4582–4597. [CrossRef]
19. Lester, B.; Al-Rfou, R.; Constant, N. The Power of Scale for Parameter-Efficient Prompt Tuning. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Punta Cana, Dominican Republic, 7–11 November 2021; Association for Computational Linguistics: Stroudsburg, PA, USA, 2021; pp. 3045–3059. [CrossRef]
20. Hu, E.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, L.; Chen, W. LoRA: Low-Rank Adaptation of Large Language Models. *arXiv* **2021**, arXiv:cs.CL/2106.09685.
21. Dettmers, T.; Pagnoni, A.; Holtzman, A.; Zettlemoyer, L. Qlora: Efficient finetuning of quantized llms. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2024; Volume 33.
22. Tunstall, L.; Beeching, E.; Lambert, N.; Rajani, N.; Rasul, K.; Belkada, Y.; Huang, S.; von Werra, L.; Fourrier, C.; Habib, N.; et al. Zephyr: Direct distillation of lm alignment. *arXiv* **2023**, arXiv:2310.16944.
23. Pang, B.; Lee, L. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.* **2008**, *2*, 1–135. [CrossRef]
24. Zhang, Y.; Jin, R.; Zhou, Z.H. Understanding bag-of-words model: A statistical framework. *Int. J. Mach. Learn. Cybern.* **2010**, *1*, 43–52. [CrossRef]
25. Cavnar, W.B.; Trenkle, J.M. N-gram-based text categorization. In Proceedings of the SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Eetrieval, Las Vegas, NV, USA, 11–13 April 1994; Volume 161175, p. 14.
26. Robertson, S. Understanding inverse document frequency: On theoretical arguments for IDF. *J. Doc.* **2004**, *60*, 503–520. [CrossRef]
27. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. *arXiv* **2013**, arXiv:1301.3781.
28. Maron, M.E. Automatic indexing: an experimental inquiry. *J. ACM* **1961**, *8*, 404–417. [CrossRef]
29. Cover, T. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theory* **1968**, *4*, 515–516. [CrossRef]
30. Vapnik, V. Support-vector networks. *Mach. Learn.* **1995**, *20*, 273–297.
31. Joachims, T. Text categorization with support vector machines: Learning with many relevant features. In Proceedings of the European Conference on Machine Learning, Chemnitz, Germany, 21–23 April 1998; Springer: Berlin/Heidelberg, Germany, 1998; pp. 137–142.
32. Quinlan, J.R. *C4. 5: Programs for Machine Learning*; Elsevier: Amsterdam, The Netherlands, 2014.

33. Mishu, S.Z.; Rafiuddin, S. Performance analysis of supervised machine learning algorithms for text classification. In Proceedings of the 2016 19th International Conference on Computer and Information Technology (ICCIT), Dhaka, Bangladesh, 18–20 December 2016; pp. 409–413.
34. Nigam, K.; McCallum, A.; Mitchell, T. *Semi-Supervised Text Classification Using EM*; MIT Press: Cambridge, MA, USA, 2006.
35. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
36. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2024; Volume 27.
37. Liu, P.; Qiu, X.; Huang, X. Recurrent neural network for text classification with multi-task learning. *arXiv* **2016**, arXiv:1605.05101.
38. Wang, Z.; Hamza, W.; Florian, R. Bilateral multi-perspective matching for natural language sentences. *arXiv* **2017**, arXiv:1702.03814.
39. Kim, Y. Convolutional neural networks for sentence classification. *arXiv* **2014**, arXiv:1408.5882.
40. Zhou, P.; Qi, Z.; Zheng, S.; Xu, J.; Bao, H.; Xu, B. Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling. *arXiv* **2016**, arXiv:1611.06639.
41. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2017; Volume 30.
42. Kaplan, J.; McCandlish, S.; Henighan, T.; Brown, T.B.; Chess, B.; Child, R.; Gray, S.; Radford, A.; Wu, J.; Amodei, D. Scaling laws for neural language models. *arXiv* **2020**, arXiv:2001.08361.
43. OpenAI, R. Gpt-4 technical report. arxiv 2303.08774. *arXiv* **2023**, arXiv:2303.08774.
44. Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv* **2023**, arXiv:2307.09288.
45. Webson, A.; Pavlick, E. Do Prompt-Based Models Really Understand the Meaning of Their Prompts? In Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Seattle, WA, USA, 10–15 July 2022; pp. 2300–2344. [CrossRef]
46. Strubell, E.; Ganesh, A.; McCallum, A. Energy and Policy Considerations for Deep Learning in NLP. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019; Korhonen, A., Traum, D., Màrquez, L., Eds.; pp. 3645–3650. [CrossRef]
47. Wang, X.; Na, C.; Strubell, E.; Friedler, S.; Luccioni, S. Energy and Carbon Considerations of Fine-Tuning BERT. *arXiv* **2023**, arXiv:2311.10267.
48. Schwartz, R.; Dodge, J.; Smith, N.A.; Etzioni, O. Green ai. *Commun. ACM* **2020**, *63*, 54–63. [CrossRef]
49. Wang, W.; Wei, F.; Dong, L.; Bao, H.; Yang, N.; Zhou, M. MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers. In *Advances in Neural Information Processing Systems*; Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H., Eds.; MIT Press: Cambridge, MA, USA, 2020; Volume 33, pp. 5776–5788.
50. Dettmers, T.; Pagnoni, A.; Holtzman, A.; Zettlemoyer, L. Qlora: Efficient finetuning of quantized llms. *arXiv* **2023**, arXiv:2305.14314.
51. Sun, X.; Li, X.; Li, J.; Wu, F.; Guo, S.; Zhang, T.; Wang, G. Text Classification via Large Language Models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*; Bouamor, H., Pino, J., Bali, K., Eds.; Association for Computational Linguistics: Singapore, 2023; pp. 8990–9005. [CrossRef]
52. Lepagnol, P.; Gerald, T.; Ghannay, S.; Servan, C.; Rosset, S. Small Language Models Are Good Too: An Empirical Study of Zero-Shot Classification. In Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024), Torino, Italy, 20–25 May 2024; Calzolari, N., Kan, M.Y., Hoste, V., Lenci, A., Sakti, S., Xue, N., Eds.; ICCL: Fort, Mumbai, 2024; pp. 14923–14936.
53. Chae, Y.; Davidson, T. *Large Language Models for Text Classification: From Zero-Shot Learning to Fine-Tuning*; Open Science Foundation: Charlottesville, VA, USA, 2023.
54. Yu, H.; Yang, Z.; Pelrine, K.; Godbout, J.F.; Rabbany, R. Open, Closed, or Small Language Models for Text Classification? *arXiv* **2023**, arXiv:2308.10092.
55. Dong, Q.; Li, L.; Dai, D.; Zheng, C.; Wu, Z.; Chang, B.; Sun, X.; Xu, J.; Sui, Z. A survey on in-context learning. *arXiv* **2022**, arXiv:2301.00234.
56. Maas, A.L.; Daly, R.E.; Pham, P.T.; Huang, D.; Ng, A.Y.; Potts, C. Learning Word Vectors for Sentiment Analysis. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Portland, OR, USA, 19–24 June 2011; pp. 142–150.
57. Zhang, X.; Zhao, J.J.; LeCun, Y. Character-level Convolutional Networks for Text Classification. In Proceedings of the NIPS, Montreal, QC, Canada, 7–10 December 2015.
58. Li, Y.; Su, H.; Shen, X.; Li, W.; Cao, Z.; Niu, S. DailyDialog: A Manually Labelled Multi-turn Dialogue Dataset. In Proceedings of the 8th International Joint Conference on Natural Language Processing (IJCNLP 2017), Taipei, Taiwan, 1 December 2017.
59. Saravia, E.; Liu, H.C.T.; Huang, Y.H.; Wu, J.; Chen, Y.S. CARER: Contextualized Affect Representations for Emotion Recognition. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November 2018; Riloff, E., Chiang, D., Hockenmaier, J., Tsujii, J., Eds.; pp. 3687–3697. [CrossRef]
60. Zeroshot. Twitter Financial News Sentiment. 2023. Available online: https://huggingface.co/datasets/zeroshot/twitter-financial-news-sentiment (accessed on 8 August 2024).

61. Tan, F.A.; Hettiarachchi, H.; Hürriyetoğlu, A.; Caselli, T.; Uca, O.; Liza, F.F.; Oostdijk, N. Event Causality Identification with Causal News Corpus—Shared Task 3, CASE 2022. In Proceedings of the 5th Workshop on Challenges and Applications of Automated Extraction of Socio-Political Events from Text (CASE), Abu Dhabi, United Arab Emirates, 7–8 December 2022; Hürriyetoğlu, A., Tanev, H., Zavarella, V., Yörük, E., Eds.; pp. 195–208. [CrossRef]
62. Ilharco, G.; Ilharco, C.; Turc, I.; Dettmers, T.; Ferreira, F.; Lee, K. High performance natural language processing. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Tutorial Abstracts, Online, 19–20 November 2020; pp. 24–27.
63. Dettmers, T.; Lewis, M.; Belkada, Y.; Zettlemoyer, L. GPT3.int8(): 8-bit Matrix Multiplication for Transformers at Scale. In *Advances in Neural Information Processing Systems*; Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K.; Oh, A., Eds.; Curran Associates, Inc.: New York, NY, USA, 2022; Volume 35, pp. 30318–30332.
64. Penrose, L.S. The elementary statistics of majority voting. *J. R. Stat. Soc.* **1946**, *109*, 53–57. [CrossRef]
65. Ustalov, D.; Pavlichenko, N.; Tseitlin, B. Learning from Crowds with Crowd-Kit. *arXiv* **2021**, arXiv:2109.08584. [CrossRef]
66. Dawid, A.P.; Skene, A.M. Maximum likelihood estimation of observer error-rates using the EM algorithm. *J. R. Stat. Soc. Ser. Appl. Stat.* **1979**, *28*, 20–28. [CrossRef]
67. Zhang, Y.; Chen, X.; Zhou, D.; Jordan, M.I. Spectral methods meet EM: A provably optimal algorithm for crowdsourcing. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2014, Volume 27.
68. Hovy, D.; Berg-Kirkpatrick, T.; Vaswani, A.; Hovy, E. Learning Whom to Trust with MACE. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Atlanta, GA, USA, 9 June 2013; Vanderwende, L.; Daumé, H., III, Kirchhoff, K., Eds.; 2013; pp. 1120–1130.
69. Whitehill, J.; Wu, T.F.; Bergsma, J.; Movellan, J.; Ruvolo, P. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2009; Volume 22.
70. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv* **2019**, arXiv:1907.11692.
71. Zhang, S.; Roller, S.; Goyal, N.; Artetxe, M.; Chen, M.; Chen, S.; Dewan, C.; Diab, M.; Li, X.; Lin, X.V.; et al. Opt: Open pre-trained transformer language models. *arXiv* **2022**, arXiv:2205.01068.
72. Jiang, A.Q.; Sablayrolles, A.; Mensch, A.; Bamford, C.; Chaplot, D.S.; Casas, D.d.l.; Bressand, F.; Lengyel, G.; Lample, G.; Saulnier, L.; et al. Mistral 7B. *arXiv* **2023**, arXiv:2310.06825.
73. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2015**, arXiv:1412.6980.
74. Demšar, J. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.* **2006**, *7*, 1–30.
75. Kalajdzievski, D. A rank stabilization scaling factor for fine-tuning with lora. *arXiv* **2023**, arXiv:2312.03732.
76. Liu, S.Y.; Wang, C.Y.; Yin, H.; Molchanov, P.; Wang, Y.C.F.; Cheng, K.T.; Chen, M.H. Dora: Weight-decomposed low-rank adaptation. *arXiv* **2024**, arXiv:2402.09353.