Data-free Functional Projection of Large Language Models onto Social Media Tagging Domain

Wenchuan Mu[0009−0007−2395−9731] and Kwan Hui Lim[0000−0002−4569−0901]

Singapore University of Technology and Design, Singapore
{wenchuan_mu, kwanhui_lim}@sutd.edu.sg

Abstract. Text tagging and recommendation are crucial tasks in social media applications. Specific and personalised tagging algorithms are in demand alongside the growing diversity of user needs, content creators, and social media companies. However, the rapid development of these algorithms is hindered by the scarcity of relevant data and the fast-changing nature of topics. Large language models (LLMs) offer a flexible solution with zero-shot classification capability, yet their substantial computational demands make them impractical for frequent, instantaneous tagging. While existing distillation methods aim to address this, they often require additional fine-tuning data or auxiliary data generators. Note that there exists a simple, yet easily overlooked fact that together with a well-formed prompt, a well-trained LLM already contains all the necessary information. In this work, we propose a novel approach that directly extracts a lightweight model from an LLM when we are already given a specific, functional prompt. Using functional projection, we project LLM's capabilities onto a targeted domain. This extraction process relies solely on numerical float-point data, and our experimental results demonstrate that this method provides a timely and accurate solution to the fast-evolving problem of social media tagging.

Keywords: Social media · Orthogonal projection · Large language model.

1 Introduction

Text tagging and recommendation systems are increasingly essential in social media, with over 5.16 billion[1] active social media users globally in 2024, representing around 59.3% of the world's population. The fast growth and diversification of user needs, content creators, and social media companies calls for personalised tagging algorithms. However, the development of these algorithms faces significant challenges, primarily due to the scarcity of relevant data and the rapid evolution of social media topics. In particular, social platforms like Facebook[2], which boasts over 3.15 billion monthly active users, and Instagram, with 1.68 billion, highlight the scale at which these systems must operate. Furthermore, the average social media user now spends approximately 2 hours and 20 minutes per day on these platforms[3], highlighting the need for efficient tagging solutions to match the pace of user engagement and content production.

[1] Priori Data
[2] DataReportal - Global Digital Insights
[3] Adam Connell

As a recent panacea, large language models (LLMs) can effectively tag texts given a task-specific prompt [11]. While LLMs generalise human language and have zero-shot capability, they require significant computational resources and time, making them impractical for frequent, instantaneous tagging. For instance, GPT-MoE, which boasts 1.8 trillion parameters [4], and Llama-2 [21], with variants ranging from 7 billion to 70 billion parameters, are notorious for their extensive computational and memory requirements. These models process large amounts of data, making them powerful but impractical for tasks requiring frequent, real-time responses, such as social media text tagging [11]. Additionally, LLMs are challenging to deploy efficiently in real-time environments. For instance, deploying LLMs often asks for engineering methods like tensor parallelism and model sharding to handle their heavy computational load, which further increases infrastructure costs [5]. Thus, while LLMs are capable of social media tagging, Their high resource demands highlight the need for more efficient options in applications where real-time performance is essential, like social media tagging [25].

To address this problem, we look at the following task: given a specific application and an LLM that performs well on it, we compress the model to a smaller and more efficient architecture without compromising performance. There are two wildly used categories: model pruning [14] and knowledge distillation [6]. While model pruning reduces the model size by removing unnecessary components, it is constrained by the original model's architecture [28]. In contrast, knowledge distillation offers a more flexible solution by transferring the knowledge from a large teacher model—or an ensemble of models—to a smaller student model [9]. This process involves aligning the student's predictions or internal activation with those of the teacher. Knowledge distillation enables a shift in model architecture during compression. However, most existing distillation methods rely on access to training data, which poses significant challenges in data-scarce scenarios [2]. Some researchers have attempted to overcome this limitation using synthetic data, but this approach can be computationally intensive and contradict the goals of distillation [27].

In this study, we address the problem of text data dependency. We propose to extract a lightweight model directly from an LLM, using a given specific prompt as the projection subspace. The rationale is that together with a well-formed prompt, a well-trained LLM already contains all the necessary information. Our method leverages the orthogonal projection of functions in Hilbert space, a widely studied and applied concept in functional analysis. The scalar coefficient in functional projection suggests including a learnable parameter in training, which stabilises the convergence. Furthermore, to completely set the fine-tuning process free from text data dependency, we fine-tune the lightweight classifier with random vector inputs and reuse the LLM embedding layer in the lightweight classifier. The extraction process relies solely on numerical float-point data and effectively projects the capabilities of the LLM onto a specific classification domain, *e.g.*, social media tagging. Experimental results show that our method efficiently produces accurate classifiers, which can potentially contribute to the fast-evolving social media tagging problem. Our code and model are made publicly available at https://github.com/cestwc/llm-projection.

[4] NVIDIA Blackwell: A new platform to power trillion parameter LLMs

2 Preliminary and Problem Definition

In this section, we review the relevant background of this study, including the fundamentals of language models and text tagging. We also revisit functional analysis, highlighting its relevance to classifiers. After that, we define our research problem.

2.1 Autoregressive Language Models

Autoregressive language models predict the next token in a sequence based on the tokens that have already been observed. These models are foundational in natural language processing tasks such as text generation, translation, and completion. The core idea behind autoregressive models is that the joint probability of a sequence of tokens can be factorised into a product of conditional probabilities, each conditioned on the preceding tokens. Formally, given a sequence of tokens $(x_1, x_2, \ldots, x_T)$, where x_t represents the token at time step t, the joint probability of the entire sequence can be expressed as

$$P(x_1, x_2, \ldots, x_T) = \prod_{t=1}^{T} P(x_t \mid x_1, x_2, \ldots, x_{t-1}), \tag{1}$$

where T is the sequence length and $P(x_t \mid x_1, x_2, \ldots, x_{t-1})$ denotes the conditional probability of the token x_t given all previous tokens in the sequence. This factorisation is the key mechanism by which autoregressive models generate sequences one token at a time, ensuring that each token is generated in the context of all preceding tokens.

The objective of training an autoregressive language model is to maximise the likelihood of the training data under the model. Given a set of sequences $\{(x_1, \ldots, x_T)^{(i)}\}_{i=1}^{N}$, where N is the number of sequences, the training objective can be formulated as maximizing the log-likelihood:

$$\frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T_i} \log P(x_t^{(i)} \mid x_1^{(i)}, x_2^{(i)}, \ldots, x_{t-1}^{(i)}; \theta) \tag{2}$$

where θ represents the parameters of the model, and T_i is the length of the i-th sequence. Autoregressive language models are typically implemented using recurrent neural networks, or, more recently, transformer architectures [23].

Once trained, autoregressive language models generate sequences by sampling tokens sequentially from the conditional distributions. At each time step t, the model samples a token x_t from the distribution $P(x_t \mid x_1, x_2, \ldots, x_{t-1})$. This process continues until a special end-of-sequence token is generated or a predefined maximum length is reached. Formally, the sampling process can be expressed as

$$x_t \sim P(x_t \mid x_1, x_2, \ldots, x_{t-1}; \theta). \tag{3}$$

This sampling procedure, while effective, can sometimes lead to issues such as repetition or incoherence in the generated text, which motivates the exploration of alternative sampling strategies, such as top-k sampling or nucleus sampling.

2.2 Large Language Models

Large Language Models (LLMs) have become a central component in natural language processing and artificial intelligence due to their ability to understand and generate human language with remarkable fluency and accuracy. These models, often based on deep neural networks, are trained on broad corpora of text data to capture the intricate patterns and structures within language. The mathematical foundation and the scale at which these models operate distinguish them from earlier language models.

Large language models leverage autoregressive architectures. In particular, the transformer model [23] is capable of modelling long-range dependencies through self-attention mechanisms. The core operation of the transformer is the self-attention mechanism, where attention scores weigh the importance of different tokens in a sequence relative to each other as

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{4}$$

where Q (query), K (key), and V (value) are matrices derived from the input embeddings, and d_k is the dimensionality of the key vectors. Input embeddings convert discrete tokens x_t into a continuous vector space $\mathbb{R}^d$, where d is the embedding dimension. The embeddings also preserve the positional information via positional encoding. The softmax function ensures that the attention scores sum to one, effectively creating a weighted average of the value vectors V.

"Large" language models are typically characterised by the number of parameters. The number of parameters θ in a model directly correlates with the model's capacity to learn and represent complex patterns in language. For instance, GPT-3 has 175 billions of parameters. Scaling laws in LLMs suggest that performance improves predictably as the model size, dataset size, and computational resources increase [7].

2.3 Projection of a Function

The projection of a function onto an axis is a fundamental concept in both multivariable calculus and functional analysis. This operation reduces function dimensionality along a specific direction or axis.

Consider a function $f : \mathbb{R}^d \to \mathbb{R}^v$ defined over d-dimensional space. The function f can be written as $f(\boldsymbol{u}) = f(u_1, u_2, \ldots, u_d)$, where $\boldsymbol{x} = (u_1, u_2, \ldots, u_d)$ is a vector in $\mathbb{R}^d$. The projection of f onto one of the coordinate axes, say the u_j-axis, involves examining how f varies with respect to the variable u_j while treating all other variables as constants or integrating them out. Formally, the projection of f onto the u_j-axis can be expressed as a function $g_j : \mathbb{R} \to \mathbb{R}^v$ defined by

$$g_j(u_j) = f(u_1^*, u_2^*, \ldots, u_{j-1}^*, u_j, u_{i+1}^*, \ldots, u_d^*) \tag{5}$$

where $u_1^*, u_2^*, \ldots, u_{j-1}^*, u_{j+1}^*, \ldots, u_d^*$ are fixed constants. This form of projection results in a function g_j that captures the behaviour of f along the u_j-axis by isolating the effect of u_j while holding other variables constant. The projection of functions onto axes is commonly used in visualising the structure of the data or analyzing specific aspects of the signal.

We may also project our function f onto another, typically used to find the best approximation of f in the space spanned by another function (or set of functions). This is often done by minimizing the difference (error) between the two functions in some vector space. The inner product of two functions $f(\boldsymbol{u})$ and $g(\boldsymbol{u})$ can be expressed as follows, where where Ω is the interval over which the functions are defined,

$$\langle f, g \rangle = \int_{\Omega} f(\boldsymbol{u}) g(\boldsymbol{u}) d\boldsymbol{u}. \tag{6}$$

A scalar coefficient c can be found such that $\mathrm{Projected}(f)(\boldsymbol{u}) = c \cdot g(\boldsymbol{u})$. The coefficient c is determined by Equation (7). An intuitive example illustrating the concept of function projection is presented below (Example 1).

$$c = \frac{\langle f, g \rangle}{\langle g, g \rangle}. \tag{7}$$

Example 1. Projecting u^2 onto u over the interval $[0, 1]$ will give us $\frac{3}{4}u$, *i.e.*, $c = \frac{3}{4}$.

Function projection onto subspaces is widely applied in optimisation and machine learning. Typical uses include reducing the complexity of the objective function or constraints and projecting data onto the principal subspace that captures the most variance [24]. In the context of large models, projecting functions onto subspaces can help in regularisation, feature selection, and the interpretation of model behaviour.

2.4 Problem Definition: Projecting a Large Language Model to a Specific Classification Task

This work studies how to project a sophisticated large language model onto a specific domain. Theoretically, a well-trained large language model contains all the information used to make a domain-specific classification (suppose its performance is good enough). Therefore, we are probably able to avoid the traditional compressing methods like distillation which requires at least some unlabelled text data to fine-tune the student model. In this study, we place the model itself in the centre of our methodology, and try to find its projections.

3 Projecting Large Language Model

In the following, we present the function projection of large language models. Specifically, we first formulate how a large model can be projected onto a specific classification domain with the coefficient in Equation (7). Then, we show the joint optimisation of the coefficient and the lightweight model in the subspace.

3.1 Functional Projection with Coefficient

Equation (3) suggests that the discrete output tokens of an autoregressive language model follow a conditional probability. It observes all input tokens and predicts the

next one. Since we use an autoregressive model for classification, it is fair to consider that this sampling process only occurs once, no matter whether an end token is produced. Formally, given an input sequence $(x_1, x_2, \ldots, x_T)$ and prompt sequence $(x_1, x_2, \ldots, x_{T'})$, an autoregressive classification function is formulated as

$$\begin{aligned} &\bar{f}\left(x_1, x_2, \ldots, x_{T'}, x_{(T'+1)}, x_{(T'+2)}, \ldots, x_{(T'+T+1)}\right) \\ &= P\left(\mathrm{y} \mid x_1, x_2, \ldots, x_{T'}, x_{(T'+1)}, x_{(T'+2)}, \ldots, x_{(T'+T+1)}; \theta\right). \end{aligned} \tag{8}$$

Next, if a fixed prompt sequence is given, *i.e.*, $(x_1^*, x_2^*, \ldots, x_{T'}^*)$, then the first T' input tokens of the function $\bar{f}$ are fixed constants. According to Equation (5), and

$$\begin{aligned} &\bar{g}(x_{(T'+1)}, x_{(T'+2)}, \ldots, x_{(T'+T+1)}) \\ &= \bar{f}(x_1^*, x_2^*, \ldots, x_{T'}^*, x_{i-1}^*, x_{(T'+1)}, x_{(T'+2)}, \ldots, x_{(T'+T+1)}). \end{aligned} \tag{9}$$

Intuitively, we give an example taking sentiment analysis as this specific domain. Suppose we would like to classify the positive/negative sentiment from IMDB movie reviews [15]. The simplistic method is to optimise a classifier using the training set in IMDB. However, now we consider the case if we do not have access to this (or any) training set for this task. We further consider that we have access to some well-trained large language models like Llama-2 [21] and pre-know that a prompt like Example 2 tailors large language models to make classifications in this task accurately.

Example 2. Is the sentiment of this movie review positive or negative?

If we use a simple word tokenisation, the prompt in Example 2 has a sequence length $T' = 11$. Function $\bar{g}$ thus treats these 11 tokens as fixed constants, and the mapping is from the movie review space (to prediction space) without the prompt in front.

Although $\bar{g}$ is a projection of $\bar{f}$ in a specific direction, adding this fixed-value domain-specific prompt does not effectively simplify the original large model $\bar{f}$. Thus, let $\bar{h}$ denote our potential lightweight classifier, and have the same mapping space as $\bar{g}$. Next, we project $\bar{g}$ onto $\bar{h}$ by calculating the inner products, as captured in Equation (10).

$$\mathrm{Projected}(\bar{g}) = \frac{\langle \bar{g}, \bar{h} \rangle}{\langle \bar{h}, \bar{h} \rangle} \bar{h} \tag{10}$$

Further, let the input domain of $\bar{g}$ and $\bar{h}$ be distinct (but potentially very dense and infinitely many) inputs. Equation (10) can be expressed as follows.

$$\begin{aligned} &\mathrm{Projected}(\bar{g})\left(x_{(T'+1)}, \ldots, x_{(T'+T+1)}\right) \\ &\quad= \frac{\langle \bar{g}, \bar{h} \rangle}{\langle \bar{h}, \bar{h} \rangle} \bar{h}\left(x_{(T'+1)}, \ldots, x_{(T'+T+1)}\right) \\ &\quad= \frac{\sum_i \bar{g}\left((x_{(T'+1)}, \ldots)^{(i)}\right) \bar{h}\left((x_{(T'+1)}, \ldots)^{(i)}\right)}{\sum_{i'} \bar{h}\left((x_{(T'+1)}, \ldots)^{(i')}\right) \bar{h}\left((x_{(T'+1)}, \ldots)^{(i')}\right)} \bar{h}\left(x_{(T'+1)}, \ldots\right) \end{aligned} \tag{11}$$

Till here, we may calculate the projection of $\bar{g}$ from Equation (11) because each term is straightforward to compute. Practically, as the number of inputs increases, the coefficient (fraction in Equation (11)) likely converges. Hence, our next step is to ensure or make this coefficient large enough, meaning that the angle between $\bar{g}$ and $\bar{h}$ is small.

Algorithm 1 Joint Optimisation

Initialise: $\theta \leftarrow \theta_{\text{initial}}, c \leftarrow \frac{1}{2}, \eta \leftarrow$ learning rate
for each input sequence $\left(x_{(T'+1)}, \ldots, x_{(T'+T+1)}\right)^{(i)}$ **do**
  $J(\theta, c) \leftarrow \sum_i \bar{g}\left(\left(x_{(T'+1)}, \ldots, x_{(T'+T+1)}\right)^{(i)}\right) \log\left(\frac{\bar{g}\left(\left(x_{(T'+1)}, \ldots, x_{(T'+T+1)}\right)^{(i)}\right)}{c \cdot \bar{h}\left(\left(x_{(T'+1)}, \ldots, x_{(T'+T+1)}\right)^{(i)}; \theta\right)}\right)$
  $\theta \leftarrow \theta - \eta \cdot \nabla_\theta J(\theta, c)$
  $c \leftarrow c - \eta \cdot \frac{\partial}{\partial c} J(\theta, c)$
end for
Return: Optimised parameters θ and coefficient c

3.2 Joint Optimisation of the Coefficient and Lightweight Model

It is still challenging to ensure that the coefficient is close to 1. To this end, the upcoming step is to optimise the lightweight model to decrease the angle between the two models. We present how a trainable coefficient c could help this optimisation.

Equation (7) (and its preceding paragraph) suggests that the function projection ($\mathrm{Projected}(\bar{g})$ here) is proportional to the reference function ($\bar{h}$ here). Vice versa, h is proportional to $\mathrm{Projected}(\bar{g})$ with a coefficient $\frac{1}{c}$. Moreover, this $\frac{1}{c}\,\mathrm{Projected}(\bar{g})$ is the 'closest' function to $\bar{g}$. This is formally captured in Theorem 1, and we may again refer to Example 1 to help understanding.

Theorem 1. *Let G be a Hilbert space and H a closed subspace of G. For every element g in G, there exists a unique element h^* in H such that $g - h^*$ is orthogonal to every element in H. This element h^* is called the orthogonal projection of g onto H, and it satisfies*

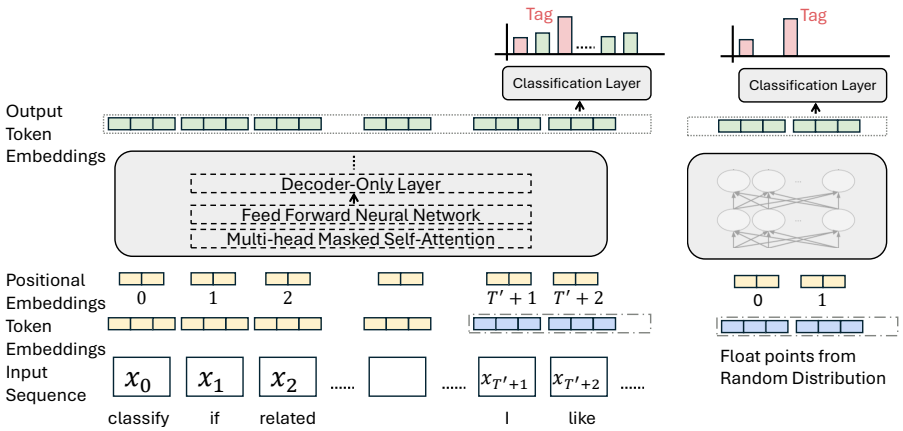
$$\|h^* - g\| = \min_{h \in H} \|g - h\|. \tag{12}$$

Furthermore, the orthogonality condition implies for all $h \in H$, $\langle h - g, h^* \rangle = 0$.

Proof. Detailed proof of Theorem 1 can be found in reference [10,18].

Theorem 1 suggests using $c \cdot \bar{h}$ to match $\bar{g}$ during training. This drives the optimisation of $\bar{h}$ to focus on minimising the angle rather than the scale difference. The value of c also monitors the convergence of $\bar{h}$. Furthermore, this trainable coefficient is simple but not trivial in the convergence, and a related example is the soft switch of copying probability in pointer-generator [20]. The training is briefly presented in Section 3.2 and advanced optimisers, losses, and weighting hyperparameters may be applied from a practice perspective. For instance, a loss like $-c^2$ may be used to properly accelerate the convergence.

Random Sampling in Embedding Space Equation (11) captures the calculation for functional projection of a large language model. However, it does not address the challenge of working with unlabelled text data during fine-tuning. Each sequence. Each sequence $\left(x_{(T'+1)}, \ldots, x_{(T'+T+1)}\right)^{(i)}$ is inherently a text string, *e.g.*, a movie review. Our motivation is that the large model should be effective even without access to any



(a) Pre-trained large language model with zero-shot predicting (b) Lightweight model

Fig. 1: A lightweight classifier can be fine-tuned without text data. The large language model is directly projected to the lightweight classifier in the embedding space. Here in the fine-tuning, random token embeddings are randomly sampled as input (blue colour) whereas the prompt embeddings (including the separator), *i.e.*, from x_0 to $x_{T'}$ are fixed (yellow colour). Positional embeddings are also fixed. From the last layer, logits or probability prediction from the large language model is used to train the lightweight classifier, and only necessary labels (pink colour) are used. The key difference from traditional distillation is that the presented process does not require text data, not even synthetic ones, improving the efficiency and robustness of the learned lightweight model.

fine-tuning text data, including synthetic ones. This raises a crucial question: how can we compute Equation (11) in the absence of text sequences?

To handle this problem, we propose to leverage the model's intrinsic capabilities by exploiting embeddings directly. Instead of relying on actual or synthetic text sequences, we can utilise random vectors in the embedding space. Here, random vectors do not mean to capture any semantic information. The prediction given by the large language model does not mean to tell whether it is indeed a "positive movie review", either. Instead, the random vector input and the prediction jointly model the inherent behaviour of the neural networks in between. Formally, let $e(x_{(t)})$ denote the token embedding function that converts x_t to a d-dimensional embedding vector, $\boldsymbol{e}_t$. Then, $\bar{g}$ can be decomposed into a function g and function e as Equation (13).

$$\begin{aligned} &\bar{g}\left(x_{(T'+1)}, x_{(T'+2)}, \ldots, x_{(T'+T+1)}\right) \\ &= g\left(e(x_{(T'+1)}), e(x_{(T'+2)}), \ldots, e(x_{(T'+T+1)})\right) \end{aligned} \tag{13}$$

$$\begin{aligned} &\bar{h}\left(x_{(T'+1)}, x_{(T'+2)}, \ldots, x_{(T'+T+1)}\right) \\ &= h\left(e(x_{(T'+1)}), e(x_{(T'+2)}), \ldots, e(x_{(T'+T+1)})\right) \end{aligned} \tag{14}$$

Similarly, $\bar{h}$ can be decomposed into h and e as shown in Equation (14). Next, instead of obtaining embedding vectors from tokens using e function, we use random d-dimensional vectors $\boldsymbol{e}^{\text{rand}} \sim \text{Uniform}([0,1]^d)$. We can then replace x_t in Equation (11) with $\boldsymbol{e}^{\text{rand}}$, $\bar{g}$ with g, and $\bar{h}$ with h to get the new form of functional projection. The same replacement can be done to Section 3.2 to enable text-free fine-tuning. An illustration of this approach is shown in Figure 1. A constraint is that the lightweight classifier must share the same embedding layer as the large language model, otherwise embeddings will be mismatched.

4 Experiment

In the following, we present our experiment to verify the behaviour of the proposed method. These analyses aim to answer the following research questions.

1. Does the angle between large and lightweight models decrease during training? What is the value that it converges to?
2. How is the classification performance of this optimised lightweight classifier? How is its accuracy compared with other training approaches?
3. How efficient is the proposed method in obtaining a lightweight domain-specific classifier, in terms of data, time, and computation resource?

4.1 RQ1: The Changes in Angle Between Large and Lightweight Models

Although we project the large language model onto a lightweight one, the lightweight model continuously changes during training. Consequently, the angle between the two models, denoted as α, also changes over time. This angle can be expressed as

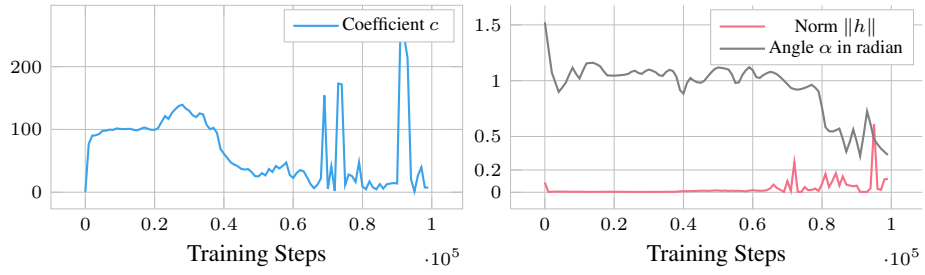
$$\alpha = \arccos\left(\frac{\langle g, h\rangle}{\|g\|\,\|h\|}\right). \tag{15}$$

We use this angle as an indicator of our training performance. The ideal case is that If g and h are collinear, *i.e.*, $\alpha = 0$. Since we already know the expression of coefficient c from Equation (7), we could derive α from c as

$$\alpha = \arccos\left(\frac{c\,\|h\|}{\|g\|}\right). \tag{16}$$

The norm of h can be estimated by the square root of $\langle h, h\rangle$, and it is similar to computing the norm of g. We can pre-compute $\|g\|$ as it is a fixed value. We compute $\|h\|$ at individual training states.

Setup To answer this question, we instantiate the LLM-function $\bar{f}$ using Llama-2 [21]. The specific domain is in movie review sentiment analysis, and the prompt is in Example 2. Thus, in function g, input embeddings starting from the 12th take random vectors in $d = 2048$ dimension. For the prediction, we take the first output token's logit values, *i.e.*, the value without the softmax function. among the 32,000 logit values, we take 10



(a) Coefficient c (b) Norm $\|h\|$ and Angle α

Fig. 2: The change of coefficient c, norm $\|h\|$, and the angle between g and h.

of them corresponding to 'Positive', 'Negative', 'Favorable', 'Unfavorable', 'Good', 'Bad', 'Excellent', 'Poor', 'Great', and 'Terrible'. We then take the maximum logit value among the five positive labels, and also the maximum logit value among the five negative labels. After that, a softmax function applies on the two maximum logit values to get respective probability. The positive class probability is then taken as the function value of g. Specifically, the Llama-2 is the Llama-2-7b model from Huggingface.[5]

We instantiate the lightweight classification function h using RoBERTa-Large [13]. This is a BERT-like [4] model with a 2048 embedding size, 24 layers, 16 attention heads, and 355 million parameters, which is an improved version of BERT. RoBERTa-Large is 19.7 times smaller than the LLaMA-2 7B model, but sophisticated enough in this movie review sentiment analysis task. Here, h is fine-tuned for 100,000 steps and in every 1,000 steps, we collect the values of c and $\|h\|$.

Result The norm of the large model converges to $\|g\| = 0.9296$, we divide $c\,\|h\|$ by this constant value to get the cosine value of α. The result is presented in Figure 2. Overall, Figure 2b shows that the angle does decrease over training, indicating the effectiveness of our approach. We observe that the lightweight classifier h still has about 0.4-radian angle when compared with g, indicating the challenge of a complete alignment. Figure 2b also shows that the norm $\|h\|$ is steadily small in the first 60,000 training steps, and starts growing after that. Almost at the same time point, we observe the significant oscillation in the value of coefficient c. This likely implies that when the norm $\|h\|$ somehow changes due to loss reduction, the timely change in coefficient c can well regularise it. Additionally, we find that while angle α decreases because of effective training, the coefficient c and norm $\|h\|$ do not necessarily converge.

4.2 RQ2: Accuracy of Our Lightweight Classifier

Apart from an intrinsic comparison against the original large model g, we would also like to evaluate the performance of our lightweight classifier on benchmarks.

[5] meta-llama/Llama-2-7b

Table 1: Macro-F1 (%) and Accuracy Score (%) of classification models fine-tuned (trained) using different approaches. Bold scores represent the highest in each category.

	IMDB [15]		AG's news [26]		Reddit [3]	
Approach	F1	Acc.	F1	Acc.	F1	Acc.
Direct Training RoBERTa	**96.2**	**96.6**	**93.8**	**94.2**	**93.7**	**94.6**
Distil Llama-2	*89.0*	*89.0*	*93.6*	*93.6*	*91.7*	*91.8*
Distil Llama-2 with DistilBERT	81.1	81.2	91.5	91.6	91.8	91.8
Distil Llama-2 with synthetic text	82.4	82.4	92.9	92.9	87.5	87.7
Our optimisation	**87.6**	**87.6**	**93.1**	**93.1**	**92.4**	**92.6**

Setup We take our fine-tuned model h and compose $\bar{h}$ using h and embedding function e. Here, the embedding layer is an unchanged copy from Llama-2-7b. The benchmarking dataset is the test set of IMDB movie reviews [15]. The IMDb movie review dataset contains 50,000 reviews split equally into 25,000 for training and 25,000 for testing, with each review labelled as either positive or negative. Each review consists of around 500 words. This dataset is commonly used for binary sentiment classification tasks, and the state-of-the-art supervised methods achieve 96.6% test accuracy.

Several baseline approaches are of interest. 1) Directly fine-tuning the same architecture (RoBERTa-Large) on IMDB training set. 2) Knowledge distillation of Llama-2-7b using the same architecture. 3) Knowledge distillation Llama-2-7b using some distillation-favoured architecture, *i.e.*, DistilBERT [19]. 4) Knowledge distillation of Llama-2-7b using the same architecture and with synthetic augmented texts [12]. Similarly, we extend our experiment to another two tasks: AG's news [26] and Reddit emotion [3]. We measure the macro-F1 score and accuracy for all approaches on all tasks.

Result Table 1 compares various approaches and shows that our optimised lightweight classifier has a consistently higher accuracy and F1 score over other methods that do not have access to original training texts. Specifically, the optimised model achieves a Macro-F1 score of 87.6% and an accuracy of 87.6% on the IMDB dataset, with a significant 5.2 percentage point higher than the data-limited baseline approach. A similar trend is observed on the AG's news and Reddit emotion tasks, with 0.2 and 4.9 percentage points higher than the data-limited baseline respectively.

On the other hand, although our method is designed for data-limited scenarios, it demonstrates competitive performance (average accuracy 91.0% and Macro F1 91.1%) against the conventional knowledge distillation method (average accuracy 91.4% and Macro F1 91.4%) where data is available. We also observe that the best performance always comes from directly trained models (average accuracy 95.1% and Macro F1 94.5%), indicating the improvement space of the proposed method.

4.3 RQ3: Efficiency

We analyse efficiency from the data and computation time perspective. We use the same setup as in RQ2, but time (estimated using a single NVIDIA RTX 2080 Ti GPU) is only counted on the IMDB classification task.

Table 2: Comparing data efficiency and time efficiency of various fine-tuning methods.

Approach	Need data?	Data Preparing time	Training time
Direct Training RoBERTa	Need labelled data	0	$<$ 1 hour
Distil Llama-2	Need data	10 ± 2 hours	2 hours
Distil Llama-2 with DistilBERT	Need data	10 ± 2 hours	1.5 hours
Distil Llama-2 with synthetic text	No	30 ± 2 hours	2 hours
Our optimisation	No	7 ± 2 hours	1.5 hours

Table 2 shows that the direct supervised training does not spend time in data preparation and has the fastest training time of under 1 hour, but it requires labelled data, unlike our optimisation, which avoids this need and reduces data preparation time by 30% to 7 ± 2 hours while maintaining a relatively quick training time of 1.5 hours. In contrast, conventional distillation methods, while effective, are less time-efficient, with data preparation taking 10 ± 2 hours and training times ranging from 1.5 to 2 hours. Distillation with synthetic text data is efficient from a data perspective, but significantly time-consuming in data preparation, because LLM is used to generate data and then generate labels. This leaves a great obstacle for rapid real-time text tagging. Overall, our method offers the most balanced and efficient approach for data-limited scenarios.

5 Related Work

To mitigate the challenge given by the large size of LLM, there is also a growing interest in smaller, more efficient models. For example, the GPT-3 Small model, with only 2.7 billion parameters, manages to outperform much larger models in certain tasks by optimizing its training data and utilizing innovative scaling techniques [1,16]. This trend towards smaller models, such as Zephyr 7B [22] and SOLAR 10.7B [8], aims to maintain high performance while significantly reducing computational costs, making these models more suitable for applications requiring rapid responses [17].

6 Conclusion

This study shows that LLMs can be effectively projected onto specific domains through functional projection without relying on traditional fine-tuning methods. Given any prompt, we successfully extract a lightweight model that retains the LLM's capabilities while significantly reducing computational demands. Our experiments show that this approach achieves competitive accuracy and efficiency in social media tagging tasks, providing a viable solution for applications requiring real-time performance.

Acknowledgments. This research is supported in part by the Ministry of Education, Singapore, under its Academic Research Fund Tier 2 (Award No. MOE-T2EP20123-0015). Any opinions, findings and conclusions, or recommendations expressed in this material are those of the authors and do not reflect the views of the Ministry of Education, Singapore.

References

1. 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. Advances in neural information processing systems **33**, 1877–1901 (2020)
2. Cho, J.H., Hariharan, B.: On the efficacy of knowledge distillation. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 4793–4801 (2019). https://doi.org/10.1109/ICCV.2019.00489
3. Demszky, D., Movshovitz-Attias, D., Ko, J., Cowen, A., Nemade, G., Ravi, S.: GoEmotions: A dataset of fine-grained emotions. In: Jurafsky, D., Chai, J., Schluter, N., Tetreault, J. (eds.) Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 4040–4054. Association for Computational Linguistics, Online (Jul 2020). https://doi.org/10.18653/v1/2020.acl-main.372, https://aclanthology.org/2020.acl-main.372
4. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Burstein, J., Doran, C., Solorio, T. (eds.) Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019). https://doi.org/10.18653/v1/N19-1423, https://aclanthology.org/N19-1423
5. Duan, J., Zhang, S., Wang, Z., Jiang, L., Qu, W., Hu, Q., Wang, G., Weng, Q., Yan, H., Zhang, X., et al.: Efficient training of large language models on distributed infrastructures: A survey. arXiv preprint arXiv:2407.20018 (2024)
6. Gou, J., Yu, B., Maybank, S.J., Tao, D.: Knowledge distillation: A survey. International Journal of Computer Vision **129**(6), 1789–1819 (Jun 2021). https://doi.org/10.1007/s11263-021-01453-z
7. 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 (2020)
8. Kim, D., Park, C., Kim, S., Lee, W., Song, W., Kim, Y., Kim, H., Kim, Y., Lee, H., Kim, J., Ahn, C., Yang, S., Lee, S., Park, H., Gim, G., Cha, M., Lee, H., Kim, S.: SOLAR 10.7b: Scaling large language models with simple yet effective depth up-scaling (2023). https://doi.org/10.48550/ARXIV.2312.15166
9. Kim, Y., Rush, A.M.: Sequence-level knowledge distillation. In: Su, J., Duh, K., Carreras, X. (eds.) Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. pp. 1317–1327. Association for Computational Linguistics, Austin, Texas (Nov 2016). https://doi.org/10.18653/v1/D16-1139, https://aclanthology.org/D16-1139
10. Kreyszig, E.: Introductory Functional Analysis with Applications, Wiley classics library, vol. 17. Wiley India Pvt. Limited (2007), https://books.google.com.sg/books?id=osXw-pRsptoC
11. Li, C., Ge, Y., Mao, J., Li, D., Shan, Y.: Taggpt: Large language models are zero-shot multimodal taggers (2023). https://doi.org/10.48550/ARXIV.2304.03022
12. Li, Z., Zhu, H., Lu, Z., Yin, M.: Synthetic data generation with large language models for text classification: Potential and limitations. In: Bouamor, H., Pino, J., Bali, K. (eds.) Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. pp. 10443–10461. Association for Computational Linguistics, Singapore (Dec 2023). https://doi.org/10.18653/v1/2023.emnlp-main.647, https://aclanthology.org/2023.emnlp-main.647
13. 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 (2019)
14. Liu, Z., Sun, M., Zhou, T., Huang, G., Darrell, T.: Rethinking the value of network pruning. In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net (2019), https://openreview.net/forum?id=rJlnB3C5Ym

15. Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., Potts, C.: Learning word vectors for sentiment analysis. In: Lin, D., Matsumoto, Y., Mihalcea, R. (eds.) Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. pp. 142–150. Association for Computational Linguistics, Portland, Oregon, USA (Jun 2011), https://aclanthology.org/P11-1015
16. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al.: Language models are unsupervised multitask learners. OpenAI blog **1**(8), 9 (2019)
17. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer (2019)
18. Rudin, W.: Principles of Mathematical Analysis. International series in pure and applied mathematics, McGraw-Hill (1976), https://books.google.com.sg/books?id=kwqzPAAACAAJ
19. Sanh, V., Debut, L., Chaumond, J., Wolf, T.: Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter (2019)
20. See, A., Liu, P.J., Manning, C.D.: Get to the point: Summarization with pointer-generator networks. In: Barzilay, R., Kan, M.Y. (eds.) Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 1073–1083. Association for Computational Linguistics, Vancouver, Canada (Jul 2017). https://doi.org/10.18653/v1/P17-1099, https://aclanthology.org/P17-1099
21. Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., et al.: Llama 2: Open foundation and fine-tuned chat models (2023). https://doi.org/10.48550/ARXIV.2307.09288
22. Tunstall, L., Beeching, E., Lambert, N., Rajani, N., Rasul, K., Belkada, Y., Huang, S., von Werra, L., Fourrier, C., Habib, N., Sarrazin, N., Sanseviero, O., Rush, A.M., Wolf, T.: Zephyr: Direct distillation of LM alignment (2023). https://doi.org/10.48550/ARXIV.2310.16944
23. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 30. Curran Associates, Inc. (2017), https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf
24. Wold, S., Esbensen, K., Geladi, P.: Principal component analysis. Chemometrics and Intelligent Laboratory Systems **2**(1), 37–52 (1987). https://doi.org/https://doi.org/10.1016/0169-7439(87)80084-9, proceedings of the Multivariate Statistical Workshop for Geologists and Geochemists
25. Yu, M., Huang, Q., Qin, H., Scheele, C., Yang, C.: Deep learning for real-time social media text classification for situation awareness–using hurricanes sandy, harvey, and irma as case studies. In: Li, Z., Huang, Q., Emrich, C.T. (eds.) Social Sensing and Big Data Computing for Disaster Management, pp. 33–50. Routledge (2020). https://doi.org/10.4324/9781003106494
26. Zhang, X., Zhao, J., LeCun, Y.: Character-level convolutional networks for text classification. In: Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 28. Curran Associates, Inc. (2015), https://proceedings.neurips.cc/paper_files/paper/2015/file/250cf8b51c773f3f8dc8b4be867a9a02-Paper.pdf
27. Zhou, T., Chiam, K.H.: Synthetic data generation method for data-free knowledge distillation in regression neural networks. Expert Systems with Applications **227**, 120327 (2023). https://doi.org/https://doi.org/10.1016/j.eswa.2023.120327
28. Zhu, M., Gupta, S.: To prune, or not to prune: Exploring the efficacy of pruning for model compression. In: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings. OpenReview.net (2018), https://openreview.net/forum?id=Sy1iIDkPM