LLMEmbed: Rethinking Lightweight LLM's Genuine Function in Text Classification

Anonymous ACL submission

Abstract

With the booming of Large Language Models (LLMs), prompt-learning has become a promising method mainly researched in various research areas. Recently, many attempts based on prompt-learning have been made to improve the performance of text classification. However, most of these methods are based on heuristic Chain-of-Thought (CoT), and tend to be more complex but less efficient. In this paper, we rethink the LLM-based text classification methodology, propose a simple and effective transfer learning strategy, namely LLMEmbed, to address this classical but challenging task. Specifically, we first study how to properly extract and fuse the text embeddings via various lightweight LLMs at differeny network depths to improve their robustness and discrimination, then adapt such embeddings to train the classifier. We perform extensive experiments on publicly available datasets, and the results show that LLMEmbed achieves strong performance while enjoys low training overhead using lightweight LLM backbones compared to recent methods based on larger LLMs, *i.e.* GPT-3, and sophisticated prompt-based strategies.

1 Introduction

Recently Large Language Models (LLMs) have shown remarkable abilities on various NLP applications, such as GPT (Brown et al., 2020; OpenAI, 2023), PaLM (Chowdhery et al., 2023) and LLaMA (Touvron et al., 2023a,b), offering services to users through dialogue. Since LLMs refer to large-scale PLMs that undergo extensive training on massive textual corpora to understand the complexity and relationships within language, LLMs exhibit amazing *emergent abilities* in comprehension and reasoning (Wei et al., 2022a). This phenomenon further promotes research of prompt learning for LLMs (White et al., 2023), such as Chain-of-Thought (CoT) (Wei et al., 2022b) and Tree-of-Thoughts (ToT) (Yao et al., 2023).

Prompt-learning (Liu et al., 2023) is closely connected to the training and inference of LLMs. Instead of adapting the pre-trained language models (PLMs) to address downstream tasks, prompt-learning directly adapts LLMs to cloze-style prediction, autoregressive modeling, or sequence to sequence generation, leading to promising performances on various tasks (Ding et al., 2022). Its major advantage of is that, given a suite of appropriate prompts, LLMs can be used to solve a great number of tasks (Brown et al., 2020; Sun et al., 2021) with no necessity of training from the scratch. However, such a paradigm is deeply involved with prompt engineering to find the most appropriate prompts to improve the overall performance.

Based on the sophisticated prompts, LLMs are guided to generate results and have achieved counterpart, or even stronger performance comparable to supervised baselines in various downstream NLP tasks such as natural language inference (Sheng et al., 2023), question answering (Trivedi et al., 2022), information extraction (Josifoski et al., 2023), named entity recognition (Wang et al., 2023) and relation extraction (Wadhwa et al., 2023). Nevertheless, as a generative model, LLMs still underperform discriminative models in text classification. Recently Sun *et al.* introduced the so-called clue and reasoning prompting (CARP) (Sun et al., 2023) to guide GPT-3 through text classification and got the SOTA performances on 5 widely-used text classification benchmarks.

For specific scenario customization of LLMs (Zhang et al., 2024) or considerations regarding user data privacy security (Tan et al., 2024), there is still a need for the localized deployment of LLMs. However, deploying LLMs is not always easily available as it requires enormous amounts of computation for training and inference (Xia et al., 2023). Moreover, it is worthy noting that only lightweight LLMs (*e.g.* LLaMA2-7B) are open-source (compared to GPT-3 with 175B parameter scale, 7B is

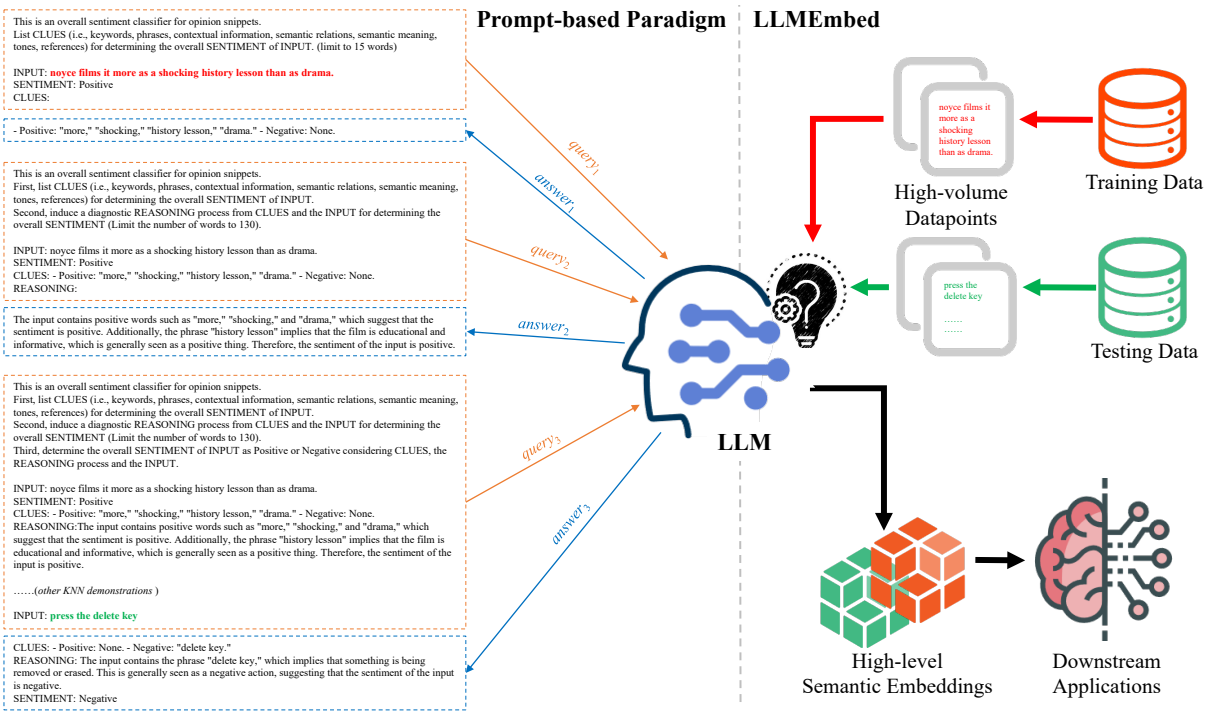

Figure 1: The principle of our proposed LLMEmbed. The left part shows how recent prompt-based methods work to classify texts. It can be seen that such a multi-step reasoning process can merely be performed serially, thus leading to high inference overhead. For comparison, instead of using LLM's content generation ability, we use the latent semantic embeddings extracted by LLMs to realize a much more effective adaptation for downstream classification tasks.

lightweight) by far. Subsequently, most users can merely deploy the open-source lightweight LLMs. As the parameter scale of open-source lightweight LLMs is far smaller than online LLMs, the emergent abilities of lightweight LLMs fall behind (Wei et al., 2022a).

In this paper, we rethink if the prompt-based paradigm remain effective to lightweight LLMs for text classification, and propose a novel and effective paradigm, namely LLMEmbed to improve the overall training efficiency and generalized performance. Specifically, we fully investigate the effectiveness of prompt-based methodology in text classification, and realize a low-cost and easy-to-use transfer learning framework via the use of LLM embeddings. We perform extensive experiments on publicly available datasets and fairly compare LLMEmbed with recent baselines. It is observed that the the inference of prompt-based paradigms is complex and highly costly. Moreover, their performance is not as promising as CARP (Sun et al., 2023) claimed when lightweight LLMs are used, some of the generated results even deviates from the inputs, *i.e. hallucination* (Zhang et al., 2023; Huang et al., 2023). For comparison, our LLMEmbed enjoys a highly robust performance and very low training overhead in all scenarios, which further highlights the usefulness of LLMEmbed.

In summary, our contributions in this paper are listed as follows:

- To the best of our knowledge, we are the first to adapt lightweight LLM's semantic embeddings for text classification. We propose a simple but effective paradigm, namely LLMEmbed, based on lightweight LLMs to address the text classification task. Our paradigm achieves SOTA results compared with prompt-based methods with the same lightweight LLM backbone, and comparable performance to methods using large-scale LLMs.
- Our LLMEmbed paradigm directly constructs the mapping from input texts to output classification results. Therefore, there is no need for users to design sophisticated prompts to align inputs and outputs, *i.e.* there exists no hallucination. Compared to existing works, LLMEmbed is budget-saving since no extra token overhead is required.
- Our LLMEmbed is more flexible, scalable and

efficient compared to prompt-based methods. LLMEmbed can combine the embeddings of lightweight LLMs with discriminative models (*e.g.* RoBERTa and BERT), or employ other representation learning methods to improve the classification performance. Moreover, LLMEmbed can perform classification in a high-speed parallel manner by feeding a large batch of text datapoints as input, whereas prompt-based paradigms cannot.

2 Related Work

2.1 Lightweight Large Language Models

Scaling up the parameters of a language model has demonstrated the effectiveness on various NLP tasks (Brown et al., 2020; Chowdhery et al., 2023). Especially, the breakthrough achieved by ChatGPT/GPT-4 (OpenAI, 2023) has made LLMs a promising approach to understanding language and generating contents. However, considering specific scenario customization of LLMs (Zhang et al., 2024), user data privacy security (Tan et al., 2024), and limited computational resource (Xia et al., 2023), open-source lightweight LLMs also aroused researchers' interests, such as LLaMA2-7B (Touvron et al., 2023b), OPT-6.7B (Zhang et al., 2022a), Pythia-6.9B (Biderman et al., 2023).

2.2 Prompt-learning

Prompt works as the input to guide LLMs to generate contents satisfying users' expectations. With the recent booming of generative LLMs, the prompt engineering (Liu et al., 2023), which focuses on designing effective prompts, has been a promising research topic. *In-Context Learning* (ICL) is a kind of typical prompt-learning which lists examples of the dataset as demonstrations to the LLM without adjusting the LLM's network architecture (Brown et al., 2020; Schick and Schütze, 2021b). Besides the demonstrations, *Instruction-Following* introduces task-describing instructions with the desired responses into the prompt, and then fine-tunes LLMs on the instructional data (Yi et al., 2019; Wei et al., 2021; Mishra et al., 2022; Ouyang et al., 2022; Wang et al., 2022b). *Impersonation* is another technique which make LLMs pretend to be a domain expert when answering a domain-specific question (Salewski et al., 2023).

From the viewpoint of human being's thinking process when solving complicated tasks, Wei et al. proposes *Chain-of-Thought* (CoT) prompting to make LLMs decompose the problem into intermediate steps and solve each before giving the final answer (Wei et al., 2022b). Inspired by CoT, extensions such as *zero-shot variants* (Kojima et al., 2022) and *Auto-CoT* (Zhang et al., 2022b) have been introduced. *Self-consistency* samples multiple reasoning paths and selects the most consistent answer via a vote (Wang et al., 2022a). *Least-to-Most* decomposes a given complex problem and solves subproblems iteratively to get the final answer (Zhou et al., 2022). *ReAct* allows the language model to interact with external environments such as Wikipedia to incorporate knowledge to inference (Yao et al., 2022). *Self-refine* makes the LLM generate an initial output and then iteratively provide feedback on the previous output, which is used to revise the output (Madaan et al., 2023). *Tree of Thoughts* (ToT) generalize CoT to maintain a tree of thoughts containing multiple different steps, enabling the LLM to self-evaluate the best way (Yao et al., 2023).

2.3 Prompt-based Text Classification

Based on the designed prompt, language models are guided to generate answers as the classification results. Schick et al. reformulated input texts into cloze-style phrases and generated the labels (Schick and Schütze, 2021a). Han et al. designed sub-prompts and composed them into final prompts based on logic rules, to guide the language model to generate results (Han et al., 2022). Liu et al. designed prompt based on the test sample's semantically similar examples (Liu et al., 2022), and Shi et al. used k-nearest neighbor (KNN) algorithm to retrieve similar examples as demonstrations of ICL prompts (Shi et al., 2022).

Further combining ICL prompts, CoT and KNN algorithm, Sun et al. proposed a method called *clue and reasoning prompting* (CARP) (Sun et al., 2023) to guide GPT-3 through text classification and got the SOTA performances on 5 widely-used benchmarks. First, they used texts and labels of training dataset to generate clues and reasoning of text classification by GPT-3. Next, they used PLM such as SimCSE (Gao et al., 2021) to compute the sentence level representation of training dataset, and then got the most semantically similar demonstrations to the test sample based on KNN algorithm. Finally, based on the reasoning process of these demonstrations, GPT-3 performs reasoning on the test sample and generates the classification result with many rounds of voting.

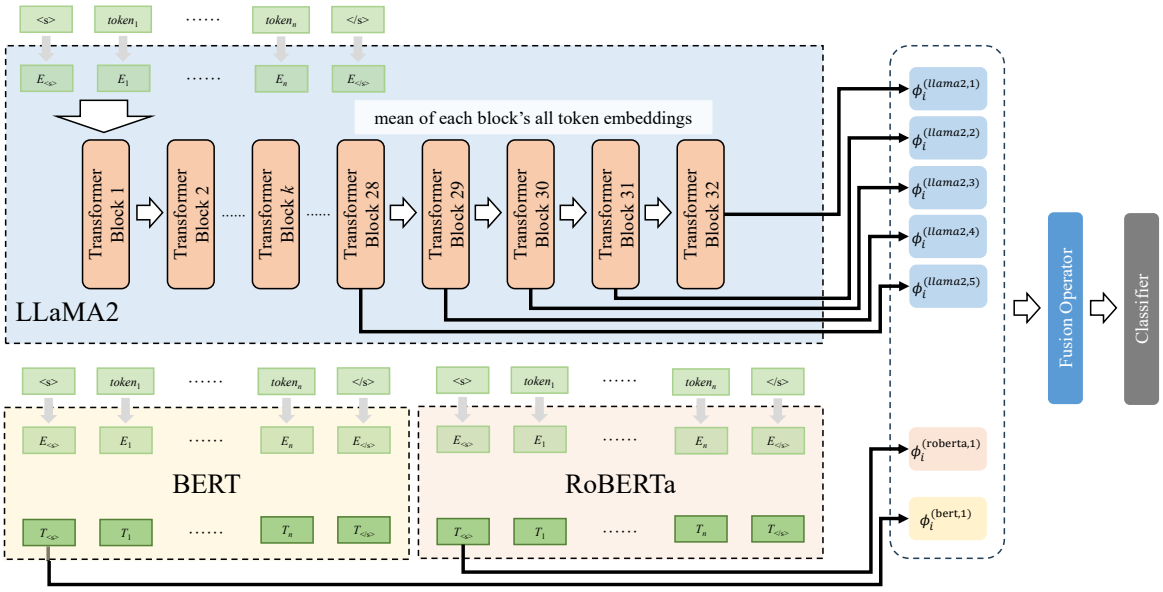

Figure 2: The demonstration of our LLEmbed method. The whole pipeline is a kind of typical transfer learning framework in which the parameters of backbone are pre-trained and frozen, and only the parameters of classifier head is fine-tuned during training. We investigate to fuse the semantic embeddings extracted from **llama2**, **roberta** and **bert**. Moreover, for **llama2**, we extract the embeddings at multiple network depths, and fuse them later via pooling operators to improve the embeddings' generalized ability.

3 Methodology

As mentioned in the Section 1, improving the performance of locally deployed lightweight LLMs is urgently important, considering specific scenario customization of LLMs (Zhang et al., 2024), user data privacy security (Tan et al., 2024), and limited computational resource (Xia et al., 2023). However, the emergent abilities of lightweight LLMs fall behind online LLMs due to their far smaller parameter scale (Wei et al., 2022a), which subsequently limits the effectiveness of lightweight LLMs in prompt-based text classification methods. Therefore, we present the so-called LLMEmbed, which directly employ the lightweight LLMs to extract text embeddings, and improve the overall performance of text classification.

In this paper, we extract and fuse embeddings extracted by multiple backbones at different depths to improve the robustness and generalization. Let us take $f(\cdot|m, d_m)$ to denote the embedding extraction where m is the model and d_m is the network depth. $g(\cdot|\theta_g)$ refers to the classifier head. $D = \{x_i, y_i\}_{i=1}^{N}$ is the dataset where x_i is the datapoint and y_i is the label. D is divided into D_{train} and D_{test} for training and evaluation .

During training, we first randomly a batch of datapoints $\{x_i, y_i\}_1^{BS} \in D_{train}$, then feed them into $f(\cdot|m, d_m)$ to extract the semantic embeddings $\{\boldsymbol{\phi}_i^{(m,d_m)}\}$.

$$\boldsymbol{\phi}_i^{(m,d_m)} = f(x_i|m, d_m), \tag{1}$$

where $m \in \{\textbf{llama2}, \textbf{roberta}, and\ \textbf{bert}\}$ refers to the backbone selected, d_m denotes the embedding is extracted from the last d_m-th block of model m. Specifically, $d_m \in \{1...5\}$ for **llama2** where the embedding of each block is the mean of this block's all token embeddings, $d_m = 1$ for **roberta** and **bert** where the final sentence semantic is used as the embedding.

Following we fuse all available embeddings via the operator $v(\cdot)$ to get the final semantic embedding $\{\boldsymbol{\psi}_i\}$.

$$\boldsymbol{\psi}_i = v(\{\boldsymbol{\phi}_i^{(m,d_m)}\}) \tag{2}$$

The fusion operators investigated in this paper are listed in Table 1. It can be seen that average pooling, max pooling, co-occurrence pooling and concatenation are considered. Due to its complexity, here we take co-occurrence pooling as example to demonstrate the fusion process in detail.

Co-occurence pooling. Given the embedding set $\{\boldsymbol{\phi}_i^{m,d_m}\}$ for each datapoint x_i, we first use linear mappings to project them into the same latent space and align them to the same length $1 \times k$,

Table 1: Fusion strategies employed in the LLMEmbed. The abbreviations Avg, Max, Co and Cat refer to average pooling, max pooling, co-occurrence pooling and concatenation, respectively. $v \rightarrow \hat{v}$ represents aligning an embedding vector v to a new embedding vector $\hat{v}$ of new size. Note that the resize operation is executed by a Linear layer, and the sizes of $\boldsymbol{\phi}_i^{(llama2,1)}$, $\boldsymbol{\phi}_i^{(roberta,1)}$, $\boldsymbol{\phi}_i^{(bert,1)}$ are 4096-d, 1024-d, 1024-d, respectively.

Index	Embeddings	Operator $v(\cdot)$	Description
1	$\boldsymbol{\phi}_i^{(llama2,1)}$	/	/
2	$\{\boldsymbol{\phi}_i^{(llama2,d_m)}\}_{d_m=[1...5]}$	Avg	$\boldsymbol{\psi}_i = Avg(\{\boldsymbol{\phi}_i^{(llama2,d_m)}\})$
3		Max	$\boldsymbol{\psi}_i = Max(\{\boldsymbol{\phi}_i^{(llama2,d_m)}\})$
4		Cat	$\boldsymbol{\psi}_i = Cat(\{\boldsymbol{\phi}_i^{(llama2,d_m)}\})$
5	$\{\boldsymbol{\phi}_i^{(llama2,d_m)}\}_{d_m=[1...5]}$, $\boldsymbol{\phi}_i^{(bert,1)}$	$Avg + Cat$	$\boldsymbol{\psi}_i = Cat(Avg(\{\boldsymbol{\phi}_i^{(llama2,d_m)}\}), \boldsymbol{\phi}_i^{(bert,1)})$
6		$Max + Cat$	$\boldsymbol{\psi}_i = Cat(Max(\{\boldsymbol{\phi}_i^{(llama2,d_m)}\}), \boldsymbol{\phi}_i^{(bert,1)})$
7		Cat	$\boldsymbol{\psi}_i = Cat(\{\boldsymbol{\phi}_i^{(llama2,d_m)}\}, \boldsymbol{\phi}_i^{(bert,1)})$
8	$\{\boldsymbol{\phi}_i^{(llama2,d_m)}\}_{d_m=[1...5]}$, $\boldsymbol{\phi}_i^{(roberta,1)}$	$Avg + Cat$	$\boldsymbol{\psi}_i = Cat(Avg(\{\boldsymbol{\phi}_i^{(llama2,d_m)}\}), \boldsymbol{\phi}_i^{(roberta,1)})$
9		$Max + Cat$	$\boldsymbol{\psi}_i = Cat(Max(\{\boldsymbol{\phi}_i^{(llama2,d_m)}\}), \boldsymbol{\phi}_i^{(roberta,1)})$
10		Cat	$\boldsymbol{\psi}_i = Cat(\{\boldsymbol{\phi}_i^{(llama2,d_m)}\}, \boldsymbol{\phi}_i^{(roberta,1)})$
11	$\{\boldsymbol{\phi}_i^{(llama2,d_m)}\}_{d_m=[1...5]}$, $\boldsymbol{\phi}_i^{(bert,1)}$, $\boldsymbol{\phi}_i^{(roberta,1)}$	$Avg + Cat$	$\boldsymbol{\psi}_i = Cat(Avg(\{\boldsymbol{\phi}_i^{(llama2,d_m)}\}), \boldsymbol{\phi}_i^{(bert,1)}, \boldsymbol{\phi}_i^{(roberta,1)})$
12		$Max + Cat$	$\boldsymbol{\psi}_i = Cat(Max(\{\boldsymbol{\phi}_i^{(llama2,d_m)}\}), \boldsymbol{\phi}_i^{(bert,1)}, \boldsymbol{\phi}_i^{(roberta,1)})$
13		Cat	$\boldsymbol{\psi}_i = Cat(\{\boldsymbol{\phi}_i^{(llama2,d_m)}\}, \boldsymbol{\phi}_i^{(bert,1)}, \boldsymbol{\phi}_i^{(roberta,1)})$
14		$Cat + Co$	$\mathbf{1}: \{\boldsymbol{\phi}_i^{(llama2,d_m)}\}_{5\times4096} \rightarrow \{\hat{\boldsymbol{\phi}}_i^{(llama2,d_m)}\}_{5\times1024}$ $\mathbf{2}: Cat(\{\hat{\boldsymbol{\phi}}_i^{(llama2,d_m)}\}, \boldsymbol{\phi}_i^{(bert,1)}, \boldsymbol{\phi}_i^{(roberta,1)}) \rightarrow \mathbf{X}_{7\times1024}$ $\mathbf{3}: \boldsymbol{\psi}_i = PN(Cat(\mathbf{X}\mathbf{X}^{\mathrm{T}}[1:7]), \sigma)$
15		$Cat + Co$ $+Avg + Cat$	$\mathbf{1}: \{\boldsymbol{\phi}_i^{(llama2,d_m)}\}_{5\times4096} \rightarrow \{\hat{\boldsymbol{\phi}}_i^{(llama2,d_m)}\}_{5\times1024}$ $\mathbf{2}: Cat(\{\hat{\boldsymbol{\phi}}_i^{(llama2,d_m)}\}, \boldsymbol{\phi}_i^{(bert,1)}, \boldsymbol{\phi}_i^{(roberta,1)}) \rightarrow \mathbf{X}_{7\times1024}$ $\mathbf{3}: \boldsymbol{\psi}_i = Cat(PN(Cat(\mathbf{X}\mathbf{X}^{\mathrm{T}}[1:7]), \sigma), Avg(\{\boldsymbol{\phi}_i^{(llama2,d_m)}\}))$

thus making $\boldsymbol{\phi}_i^{m,d_m} \in \mathbb{R}^{1\times K} \quad \forall m, d$. Then they are stacked to formulate the $\hat{\boldsymbol{\phi}}_i \in \mathbb{R}^{H\times K}$ where $H = \sum_m m \cdot d_m$. Following, we calculate their co-occurrence statistics $\psi_i \in \mathbb{K} \times \mathbb{K}$.

$$\boldsymbol{\psi}_i = PN(\hat{\boldsymbol{\phi}}_i \otimes \hat{\boldsymbol{\phi}}_i^T, \sigma); \tag{3}$$

$$s.\,t.\ PN(\mathbf{X};\sigma) = \frac{1-e^{-\sigma\mathbf{X}}}{1+e^{-\sigma\mathbf{X}}} = \tanh(2\sigma\mathbf{X}), \tag{4}$$

where $PN(\cdot)$ is a power normalization function used to balance the power distribution of co-occurrences, σ is the hyper-parameter to control the slope of $PN(\cdot)$ function, and the empirical value is 0.2 in this paper.

Finally, all fused embeddings are fed into the classifier head to get the predictions $\{\tilde{y}_i = g(\boldsymbol{\psi}_i|\theta_g)\}$, and we perform CrossEntropy loss to train θ_g.

$$\theta_g \leftarrow \arg\min_{\theta_g} \mathcal{L} = \sum_i y_i \log \tilde{y}_i \tag{5}$$

Overall, it can be seen that our pipeline is a typical transfer learning framework, which focuses on difference fusion strategies on LLM embeddings. As extracting embedding is a fast process, our classifier can be effectively trained and evaluated at a very high speed.

4 Experiment

4.1 Setups

We run experiments on five widely-used text classification benchmarks as CARP (Sun et al., 2023) does: SST-2 [1], MR[2], AGNews[3], R8 and R52[4].

SST-2 (Socher et al., 2013) is sampled from snippets of Rotten Tomatoes HTML files. The amounts of train set and test set are 67349 and 872 respectively, and the max length of words is 56.

MR (Maas et al., 2011) contains movie reviews representing positive or negative sentiment. The corpus has 40000 training data and 10000 testing data, of which the max length of words is 2470.

AGNews (Zhang et al., 2015) consists of 4 types of news articles from the AG's corpus. The dataset contains 120000 training and 7600 testing examples, and the max length of words is 177.

R8 and **R52** are two subsections of Reuters collection, containing 8 and 52 classifications. The R8 dataset is composed of 5485 documents for training and 2189 documents for testing, of which the max length of words is 964. The R52 dataset is composed of 6532 training and 2568 testing documents,

[1] https://nlp.stanford.edu/sentiment/
[2] http://www.cs.cornell.edu/people/pabo/movie-review-data/
[3] http://groups.di.unipi.it/∼gulli/AG_corpus_of_news_articles.html
[4] https://www.cs.umb.edu/∼smimarog/textmining/datasets/

Table 2: The accuracy performance of different settings on 5 publicly available datasets. The last row reports the mean accuracy of each method over benchmarks. The bold results indicate the best performance for each dataset.

	Backbone	SST-2	MR	AGNews	R8	R52	Avg. Acc.
PLM Methods							
(Kenton and Toutanova, 2019)	BERT-large	0.8761	0.8130	0.8216	0.8250	0.6581	0.7988
(Liu et al., 2019)	RoBERTa-large	0.9025	0.9346	0.9441	0.9676	0.4217	0.8341
Prompt-based LLM Methods							
IO (Brown et al., 2020)	LLaMA2 7B	0.8922	0.9065	0.7420	0.2481	0.1869	0.5951
CARP (Sun et al., 2023)	LLaMA2 7B	0.8842	0.8394	0.8518	0.7510	0.7305	0.8114
CARP (Sun et al., 2023)	GPT-3 175B	0.9569	0.9074	0.9525	0.9783	**0.9627**	0.9516
LLMEmbed-based Methods							
Embeddings: $\phi_i^{(llama2,1)}$ or $\{\phi_i^{(llama2,d_m)}\}_{d_m=[1\ldots5]}$							
$/$	LLaMA2 7B	0.9518	0.9522	0.9571	0.9794	0.9455	0.9572
Avg	LLaMA2 7B	0.9530	0.9545	0.9566	0.9794	0.9533	0.9594
Max	LLaMA2 7B	0.9530	0.9517	0.9555	0.9799	0.9486	0.9577
Cat	LLaMA2 7B	0.9518	0.9537	0.9359	0.9785	0.4217	0.8483
Embeddings: $\{\phi_i^{(llama2,d_m)}\}_{d_m=[1\ldots5]}$ and $\phi_i^{(bert,1)}$							
$Avg + Cat$	LLaMA2 7B	0.9553	0.9543	0.9554	0.9799	0.9517	0.9593
$Max + Cat$	LLaMA2 7B	0.9530	0.9526	0.9553	0.9794	0.9502	0.9581
Cat	LLaMA2 7B	0.9495	0.9531	0.9364	0.9781	0.8501	0.9334
Embeddings: $\{\phi_i^{(llama2,d_m)}\}_{d_m=[1\ldots5]}$ and $\phi_i^{(roberta,1)}$							
$Avg + Cat$	LLaMA2 7B	0.9541	0.9547	0.9562	0.9808	0.9544	0.9600
$Max + Cat$	LLaMA2 7B	0.9537	0.9538	0.9555	0.9794	0.9467	0.9578
Cat	LLaMA2 7B	0.9507	0.9536	0.9491	0.9790	0.4217	0.8508
Embeddings: $\{\phi_i^{(llama2,d_m)}\}_{d_m=[1\ldots5]}$, $\phi_i^{(bert,1)}$, and $\phi_i^{(roberta,1)}$							
$Avg + Cat$	LLaMA2 7B	0.9553	0.9534	0.9574	0.9808	0.9548	0.9603
$Max + Cat$	LLaMA2 7B	0.9541	0.9542	0.9555	0.9799	0.9529	0.9593
Cat	LLaMA2 7B	0.9484	0.9540	0.9505	0.9785	0.9326	0.9528
$Cat + Co$	LLaMA2 7B	0.9553	0.9533	0.9549	0.9794	0.8895	0.9465
$Cat + Co + Avg + Cat$	LLaMA2 7B	**0.9576**	**0.9549**	**0.9583**	**0.9822**	0.9568	**0.9620**

Table 3: The runtime of each method. The reported results are formulated by 'hh:mm:ss', which refer to hours, minutes and seconds respectively. (The computational device is Nvidia A100-40G.)

Method	Process	SST-2	MR	AGNews	R8	R52	Avg. Time
CARP	clues and reasoning generating	67:54:13	42:52:27	133:32:52	09:28:47	08:33:56	
	KNN with embedding extracting	00:24:54	00:15:34	00:45:11	00:02:03	00:02:28	
	inference	11:12:02	138:25:05	97:32:00	30:37:57	36:19:59	
	total runtime	79:31:09	181:33:06	231:50:03	40:08:47	44:56:23	115:36:01
LLMEmbed	train embedding extraction	00:13:30	05:43:45	00:49:04	00:10:30	00:12:39	
	test embedding extraction	00:00:10	00:33:21	00:03:09	00:04:15	00:04:57	
	training (50 epochs)	00:08:30	00:05:13	00:14:57	00:01:08	00:01:09	
	inference	00:00:01	00:00:02	00:00:02	00:00:01	00:00:01	
	total runtime	00:22:11	06:22:21	01:07:11	00:15:54	00:18:45	01:41:16

of which the max length of words is 1039.

Moreover, we solve these text classification tasks on Nvidia A100-40G GPU, and employ LLaMA2-7B (Touvron et al., 2023b) as the lightweight LLM backbone throughout all of our main experiments. The batch size we set for each dataset is 1024. The classifier is trained for 100 epochs with initial learning rate 1×10^{-4}. Our LLMEmbed is compared with recent PLM- and prompt-based baselines *w.r.t.* performance, efficiency and budget.

4.2 Performance Analysis

For the conventional PLM method, we employ the widely used BERT and RoBERTa to extract embeddings and adapt them to the downstream text classification tasks. Though the PLM method demonstrates satisfactory performance, there is still scope for enhancement, such as the RoBERTa hardly converges on R52 with only 42.17% accuracy.

For the prompt-based method, we initially leverage the Input-Output (IO) prompting (Brown et al., 2020), which provides input-output pairs as demonstrations to guide the LLM in generating results. Additionally, We implement the SOTA prompt method, CARP (Sun et al., 2023). Under the detailed guidance of CARP, the overall performance of prompt paradigm improve significantly (the average accuracy improve 21.63%). However, due to the far small parameter scale of LLaMA2(7B) compared to GPT-3(175B), the emergent abilities are somewhat limited. We note that the CARP also hallucinates, and if the prompts exceeds the

Table 4: The estimated electricity consumption of each method, which is calculated based on the the training and inference runtime. (The computational device is Nvidia A100-40G.)

Method	Process	SST-2	MR	AGNews	R8	R52	Avg. Elec Cost
CARP	clues and reasoning generating	$11.54kWh$	$7.29kWh$	$22.70kWh$	$1.61kWh$	$1.46kWh$	
	KNN with representation extracting	$0.10kWh$	$0.06kWh$	$0.19kWh$	$0.01kWh$	$0.01kWh$	
	result generating	$2.13kWh$	$26.30kWh$	$18.53kWh$	$5.82kWh$	$6.90kWh$	
	total electricity consumption	$13.77kWh$	$33.65kWh$	$41.42kWh$	$7.44kWh$	$8.37kWh$	$20.93kWh$
LLMEmbed	representation extracting	$0.05kWh$	$1.51kWh$	$0.21kWh$	$0.06kWh$	$0.07kWh$	
	downstream model	$0.01kWh$	$0.004kWh$	$0.01kWh$	$0.0009kWh$	$0.0009kWh$	
	total electricity consumption	$0.06kWh$	$1.51kWh$	$0.22kWh$	$0.06kWh$	$0.07kWh$	$0.38kWh$

Table 5: The budget comparison between online prompt-based CARP and LLMEmbed.

		SST-2	MR	AGNews	R8	R52	Avg. budget
CARP	tokens of generating clues and reasoning	22227252	41993238	32010577	2243198	4024957	
	tokens of generating result	6492145	235107723	75016765	33625747	45442485	
	total tokens	28719397	277100961	107027342	35868945	49467442	
	total token consumption	$57.44	$554.20	$214.05	$71.74	$98.93	$199.27
LLMEmbed	total electricity consumption	$0.06kWh$	$1.51kWh$	$0.22kWh$	$0.06kWh$	$0.07kWh$	
	total electricity bill	$0.0039	$0.09815	$0.0143	$0.0039	$0.00455	$0.025

LLaMA2's input length limitation, CARP will probably generate irrelevant content.

For the LLMEmbed paradigm, we employed fusion strategies as mentioned in Table 1. As shown in Table 2, it is evident that average pooling of LLM's embeddings, further concatenated with different models' embeddings is most effective.

Average pooling of LLM's embeddings: We observe that the overall performance of average pooling surpasses max pooling by 0.1%∼0.2% and only-concatenating embeddings by 3%∼10%. The average pooling retains more information than max pooling. The only-concatenation degrades the performance due to the too large space of embeddings.

Concatenating with different models' embeddings: We find that after average pooling of LLM's embeddings, concatenating it with different models' embeddings will further improve the performance, *e.g.* $Avg + Cat$ of three models outperforms two models. This is due to fusing embeddings of generative LLM and discriminative models can complement each other in semantic space.

Co-occurrence pooling: This pooling itself may be not the most effective method, but it extract the high-order representation. We concatenate this representation further and get the SOTA performance based on the LLaMA2 backbone. The SOTA LLMEmbed outperforms CARP by 7.34%, 11.55%, 10.65%, 23.12% and 22.63% for SST2, MR, AGNews, R8 and R52, respectively.

Furthermore, we also compare local lightweight LLM-based LLMEmbed with online LLM-based CARP. Despite LLaMA2's significant parameter scale disadvantage compared to GPT-3, the LLMEmbed achieves a comparable performance to the GPT-3 CARP by employing fusion strategies, even outperforms over SST-2, MR, Agnews, R8.

4.3 Efficiency

We have measured each process's time cost of employing the prompt paradigm and LLMEmbed paradigm for text classification, and listed the results in Table 3. Overall, the proposed LLMEmbed paradigm is significantly more efficient than the prompt paradigm.

As shown in Figure 3, the downstream model reaches convergence at around 50 epochs. For SST-2, LLMEmbed's time cost is only 0.46% ($22min11s/79h31min9s$) of the prompt-based paradigm. Similarly, the ratio of time cost is 3.51% for MR, 0.48% for AGNews, 0.66% for R8 and 0.70% for R52.

The reasons for the surge in LLMEmbed-based efficiency are: 1) The lightweight LLM only needs to extract the representation of the input texts, without spending time generating answers. 2) LLMEmbed only processes the original input texts without any additional sophisticated prompt words. So the length of input is much smaller than the prompt-based method, resulting in much less computation. 3) LLMEmbed can conduct the text classification in a parallel manner by feeding a batch of texts to the lightweight LLM, whereas prompt-based paradigm can't. We note that if feed a batch of prompts to guide the lightweight LLM to generate results, prompts of this batch will effect each other, leading to a low performance.

As shown in Table 3, LLMEmbed costs the most time when solving the MR task. Since the length of each text in MR is generally much longer than other datasets, solving MR task requires larger GPU memory, resulting in limited parallel process-

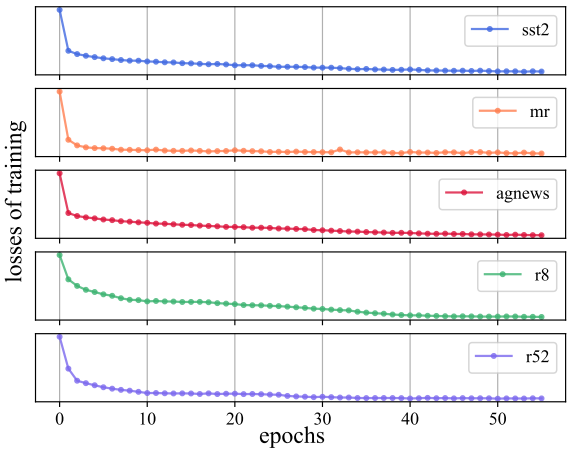

Figure 3: The losses of training the downstream model.

ing. So, the lightweight LLM spends the most time extracting the representation of MR.

4.4 Budget of Users

We first compare the electricity consumption between prompt paradigm and LLMEmbed paradigm. For the prompt-based CARP, the power of generating clues and reasoning is about $170W$, KNN with representation extracting is about $250W$, and generating result is about $190W$. So, referring to the time cost listed in Table 3, the total electricity consumption of SST-2, MR, AGNews, R8 and R52 is $13.77kWh$, $33.65kWh$, $41.42kWh$, $7.44kWh$ and $8.37kWh$ respectively.

For LLMEmbed paradigm, the power of representation extracting is about $240W$. When training or testing the downstream model, the power is $35W \sim 55W$, which is almost equivalent to the power of a GPU just booting up without running any program. We take the average $45W$ as the power of these processes. As shown in Figure 3, the downstream model converges at around 50 epochs. Combined with the time cost in Table 3, the total electricity consumption of SST-2, MR, AGNews, R8 and R52 is $0.06kWh$, $1.51kWh$, $0.22kWh$, $0.06kWh$ and $0.07kWh$ respectively.

The detailed results of electricity consumption on five benchmarks have been listed in Table 4. For SST-2, LLMEmbed's electricity consumption is merely 0.44% ($0.06kWh/13.77kWh$) of the prompt-based paradigm. Similarly, the ratio of electricity consumption is 4.49% for MR, 0.53% for AGNews, 0.81% for R8 and 0.84% for R52. So, we can see that LLMEmbed paradigm is greatly more energy efficient than prompt paradigm.

Then, we compare the budget of users between the local LLMEmbed with the online prompt-based LLM,(*e.g.* GPT[5], as CARP does). The pricing of GPT is about \$0.002 per $1k$ tokens[6]. We tokenize all the input and output of CARP, and calculate the sum of these tokens as well as its budget. For the local LLMEmbed, we have calculated the electricity consumption above. Taking Beijing's electricity tariffs as an example, it is \$0.065/kWh. So, we can further work out the electricity bill.

All the results have been listed in Table 5. Overall, the budget of local LLMEmbed is much smaller than the online prompt-based LLM. The ratio of budget is 0.01% for SST-2, 0.02% for MR, 0.01% for AGNews, 0.01% for R8 and 0.005% for R52.

The LMMEmbed paradigm extracts the representations from the original input text, thus obtaining the semantic space through the powerful language comprehension capabilities of LLMs. In this rich semantic space, we can improve the performance of LMMEmbed by employing other various semantic representation optimization methods to boost the text classification. Further, we can also adapt this paradigm to other downstream tasks by constructing a proper mapping from the semantic space to the output. This novel LLMEmbed paradigm can be beneficial for generative LLMs in handling other tasks such as information extraction.

5 Conclusion

In this paper, we propose a concise and effective LLM-based paradigm, namely **LLMEmbed**, to address the text classification. This novel LLMEmbed paradigm achieves the state-of-the-art performance when solving text classification with lightweight LLM, even comparable to the performance of LLMs (*e.g.* GPT-3) with sophisticated prompt-based strategies. The LLMEmbed paradigm is also flexible and scalable that the text embeddings via different lightweight LLMs can be fused to improve the overall performance. Moreover, the proposed LLMEmbed paradigm is far more efficient and budget saving than the prompt-based paradigm. Furthermore, we hope that our novel LLMEmbed paradigm can be beneficial for generative LLMs in handling tasks such as text classification (but not limited to text classification), which used to be solved by discriminative models.

[5]InstructGPT-3 (text-davinci-003, 175B)
[6]https://openai.com/pricing

References

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Ning Ding, Shengding Hu, Weilin Zhao, Yulin Chen, Zhiyuan Liu, Haitao Zheng, and Maosong Sun. 2022. Openprompt: An open-source framework for prompt-learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 105–113.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910.

Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. 2022. Ptr: Prompt tuning with rules for text classification. *AI Open*, 3:182–192.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv preprint arXiv:2311.05232*.

Martin Josifoski, Marija Sakota, Maxime Peyrard, and Robert West. 2023. Exploiting asymmetry for synthetic training data generation: SynthIE and the case of information extraction. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1555–1574.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, William B Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for gpt-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. Cross-task generalization via natural language crowdsourcing instructions. In *60th Annual Meeting of the Association for Computational Linguistics, ACL 2022*, pages 3470–3487. Association for Computational Linguistics (ACL).

OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Leonard Salewski, Stephan Alaniz, Isabel Rio-Torto, Eric Schulz, and Zeynep Akata. 2023. In-context impersonation reveals large language models' strengths and biases. *arXiv preprint arXiv:2305.14930*.

Timo Schick and Hinrich Schütze. 2021a. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269.

Timo Schick and Hinrich Schütze. 2021b. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352.

Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. 2023. Flexgen: High-throughput generative inference of large language models with a single gpu. In *International Conference on Machine Learning*, pages 31094–31116. PMLR.

Weijia Shi, Julian Michael, Suchin Gururangan, and Luke Zettlemoyer. 2022. knn-prompt: Nearest neighbor zero-shot inference. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3254–3265.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Xiaofei Sun, Xiaoya Li, Jiwei Li, Fei Wu, Shangwei Guo, Tianwei Zhang, and Guoyin Wang. 2023. Text classification via large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8990–9005.

Yu Sun, Shuohuan Wang, Shikun Feng, Siyu Ding, Chao Pang, Junyuan Shang, Jiaxiang Liu, Xuyi Chen, Yanbin Zhao, Yuxiang Lu, et al. 2021. Ernie 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation. *arXiv preprint arXiv:2107.02137*.

Zhi-Hao Tan, Jian-Dong Liu, Xiao-Dong Bi, Peng Tan, Qin-Cheng Zheng, Hai-Tian Liu, Yi Xie, Xiao-Chuan Zou, Yang Yu, and Zhi-Hua Zhou. 2024. Beimingwu: A learnware dock system. *arXiv preprint arXiv:2401.14427*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *arXiv preprint arXiv:2212.10509*.

Somin Wadhwa, Silvio Amir, and Byron C Wallace. 2023. Revisiting relation extraction in the era of large language models. In *Proceedings of the 61th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15566–15589.

Shuhe Wang, Xiaofei Sun, Xiaoya Li, Rongbin Ouyang, Fei Wu, Tianwei Zhang, Jiwei Li, and Guoyin Wang. 2023. Gpt-ner: Named entity recognition via large language models. *arXiv preprint arXiv:2304.10428*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022a. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, et al. 2022b. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5085–5109.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022a. Emergent abilities of large language models. *Transactions on Machine Learning Research*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022b. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C Schmidt. 2023. A prompt pattern catalog to enhance prompt engineering with chatgpt. *arXiv preprint arXiv:2302.11382*.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2023. Sheared llama: Accelerating language model pre-training via structured pruning. In *Workshop on Advancing Neural Network Training: Computational Efficiency, Scalability, and Resource Optimization (WANT@ NeurIPS 2023)*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In

Thirty-seventh Conference on Neural Information Processing Systems.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

Sanghyun Yi, Rahul Goel, Chandra Khatri, Alessandra Cervone, Tagyoung Chung, Behnam Hedayatnia, Anu Venkatesh, Raefer Gabriel, and Dilek Hakkani-Tur. 2019. Towards coherent and engaging spoken dialog response generation using automatic conversation evaluators. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 65–75.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022a. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. 2023. Siren's song in the ai ocean: A survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219*.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022b. Automatic chain of thought prompting in large language models. In *The Eleventh International Conference on Learning Representations*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, et al. 2022. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.