Cool-Fusion: Fuse Large Language Models without Training

Anonymous ACL submission

Abstract

We focus on the problem of fusing two or more heterogeneous large language models (LLMs) to leverage their complementary strengths. One of the challenges of model fusion is high computational load, specifically in fine-tuning or aligning vocabularies. To address this, we propose Cool-Fusion, a simple yet effective approach that fuses the knowledge of source LLMs, which does not require training. Unlike ensemble methods, Cool-Fusion is applicable to any set of source LLMs that have different vocabularies. To overcome the vocabulary discrepancies among LLMs, we ensemble LLMs on text level, allowing them to rerank the generated texts by each other with different granularities. Extensive experiments have been conducted across a variety of benchmark datasets. On GSM8K, Cool-Fusion increases accuracy from three strong source LLMs by a significant margin of 17.4%. We will make our source code in the attachment publicly available.

1 Introduction

Different large language models (LLMs) exhibit diverse strengths and weaknesses due to various factors, such as datasets used for pre-training and fine-tuning, architectures, optimizers, hyperparameters, and training methodologies. Recent work (Jiang et al., 2023) has found that it is possible to develop fusion methods to harness the complementary potentials of the LLMs for improved general or task-specific performance, such as higher accuracy and better alignment with human preferences.

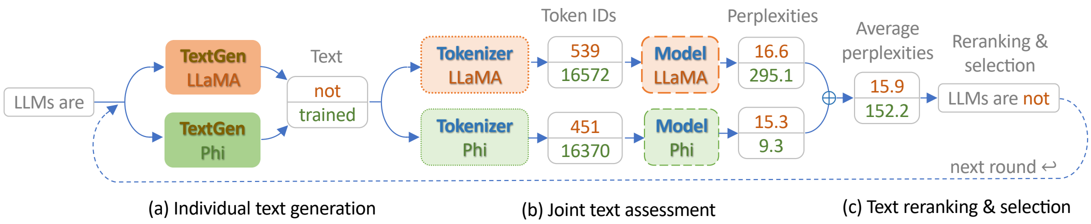
However, conventional ensemble approaches require the source LLMs to have the same token vocabulary, while weight merging (Wortsman et al., 2022; Jolicoeur-Martineau et al., 2024) is further limited to models with identical architectures. Although model fusion (Li et al., 2023a) has attracted increasing interest, it faces a series of challenges, including the formidable computational costs associated with training (Bansal et al., 2024; Xu et al., 2024), fine-tuning (Jiang et al., 2023), distillation (Taori et al., 2023; Wan et al., 2024a,b), and the combinatorial optimization needed for vocabulary alignment (Wan et al., 2024a,b; Fu et al., 2023; Xu et al., 2024). Therefore, existing fusion approaches are daunting for researchers and practitioners who cannot afford to train or fine-tuning LLMs, and are unsuitable for application scenarios that require rapid deployment.

Aiming for a general LLM fusion approach that is applicable to any set of source LLMs with diverse tokenizers, and is both cost-effective and fast to deploy, we propose Cool-Fusion to fuse the knowledge of heterogeneous LLMs without any training. The core of our algorithm combines the source LLMs to rerank text segments that they generate individually, rather than using the ensemble of LLMs as token generators with their own sets of disjoint vocabularies. In Cool-Fusion, we propose to fuse knowledge at text segments of different granularities, and discuss their pros and cons. An overview of Cool-Fusion is shown in Figure 1. In summary, Cool-Fusion has the following properties:

- Simplicity: Cool-Fusion is simple both in concept and for implementation. Unlike prior approaches, Cool-Fusion starts to generate texts as soon as we have the source LLMs, since no training of any type is required. Consequently, we do not need to worry about the problems associated with fine-tuning and training, such as overfitting the training distribution, insufficient hyper-parameter tuning, or loss of generalization ability (Fu et al., 2023).

- Availability: Based on pure inference, Cool-Fusion can be accessed by a larger range of budget-limited researchers and practitioners.

- Scalability: Cool-Fusion alternates between a generation and an evaluation step. Each of the



(a) Individual text generation (b) Joint text assessment (c) Text reranking & selection

Figure 1: An illustration of Cool-Fusion. The TextGen component is illustrated in Figure 2.

#iteration	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
LLaMA-3	**_not**	**_the**	**_only**	**_ones**	_who	**_can**	**_be**	**_used**	**_for**	**_this**	**_purpose**
Phi-3	_trained	_inherently	_only	_ones	**_that**	_can	_be	_used	_for	_this	_purpose
LLaMA-3	_not	_the	_only	_ones	_who	_have	_been	_affected	_by	_the	_pandemic
Phi-3	_trained	_on	_vast	_datasets	_that	_include	_a	_wide	_variety	_of	_human

Table 1: A example of Cool-Fusion for 10 iterations following the example in Figure 1. The first two rows show the text segments predicted by LLaMA-3 and Phi-3 jointly in Cool-Fusion, where the winning text segments are in bold. We use underscores to represent whitespaces. For comparison, the last two rows are text segments predicted by LLaMA-3 and Phi-3 individually.

two steps invoke the source LLMs independently, and small amount of texts and scores are gathered and scattered between the steps. Given k GPUs, Cool-Fusion is scalable to k source LLMs with constant delay.

- Superior performance: Albeit being simple, Cool-Fusion exhibits competitive performances over strong baselines, persistent across a wide range of challenging tasks.

We evaluated extensively on greedy completion benchmarks across various domains, including math (GSM8K, multilingual GSM, and MATH), and Q&A (CoQA, DROP, TriviaQA). We experimented on an array of open-source LLMs, including the most recent state-of-the-art LLMs, namely LLaMA-3 8B (Touvron et al., 2023), Phi-3 mini (et al, 2024), and GLM-4 9B (Zeng et al., 2023). Our results demonstrate that Cool-Fusion significantly outperforms the individual source LLMs as well as recent LLM fusion methods that require training. On the GSM8K dataset, Cool-Fusion increases the prediction accuracy from the best-performing source LLM LLaMA-3 8B by a significant margin of 17.4%.

2 Cool-Fusion: Fuse LLMs without Fine-tuning

Since the token vocabularies are usually different across LLMs, a token predicted by one LLM may not find a deterministic counterpart in another LLM. For instance, the common tokens between LLaMA-3 and Phi-3 account for only 6.4% of their total tokens, and those between Phi-3 and GLM-4 account for only 7.5%. Prior approaches resort to heuristics to find similar tokens across token vocabularies, which introduces errors and requires heavy training due to the combinatorial optimization complexity. In ensemble approaches, the predicted distributions on heterogeneous token vocabularies are first individually mapped into distributions on a shared tokens-vocabulary, and the next jointly predicted token from the shared vocabulary is the one that has the largest sum of logit values from these distributions. Inspired by this, we generalize the element of predicting from a single token to predicting a short text segment containing one or more tokens that can be commonly decoded by all heterogeneous tokenizers, and the criteria from the sum of logit values to the averaged perplexities of the text segment obtained from the LLMs. With this new approach, we can avoid the computation and inaccuracies associated with the mapping from individual token vocabularies into a single shared token vocabulary.

2.1 Overview of Cool-Fusion

A text generation task involves generating a continuation of a given context. Our approach can be easily explained with a real running example, as illustrated in Figure 1. Cool-Fusion features a text generation loop, where a text segment is generated at each iteration of the loop. In the example, we fuse two source LLMs, LLaMA-3 8B and Phi-3 mini, with the input context text being "LLMs are". Each iteration in the text generation loop consists of three steps: (1) each source LLM individually generates a text segment, (2) each source LLM computes a perplexity for every text segment generated in step 1, (3) the text segment with the smallest averaged perplexity is selected as the jointly predicted text segment, which is then broadcast to update all source LLMs. Next, we will discuss more details for each step, as illustrated in Figure 1.

In step 1 of each iteration, a text generation component (TextGen) in each source LLM is responsible for generating text segments. Different implementations of TextGen may generate text segments of different lengths, ranging from minimal decodable text segments consisting of one or a few tokens to phrases containing several words. We will discuss two implementation options in Sections 2.2 and 2.3. In Figure 1, the text segments generated by the two TextGen components are "not" and "trained", respectively.

In step 2, each text segment is sent to all LLMs to obtain a perplexity using the key-value cache from the previous iteration, specifically the key-value cache before the generation of the text segment in the current iteration. Finally, the perplexities of each text segment are gathered from every LLM and are averaged to evaluate the text segment. In Figure 1, the text segment "not" is first encoded by the tokenizers of LLaMA-3 8B and Phi-3 mini into token sequences [539,] and [451,]. These two token sequences are forwarded through their corresponding LLMs, resulting in two perplexities, 16.6 and 15.3, for text segment "not", which are finally averaged to 15.9. For better efficiency in this step, we forward all text segments, i.e. "not" and "trained", together in a batch through all LLMs.

In step 3, the winner among the text segments is selected based on their average perplexity computed in step 2. In Figure 1, the winner is "not", whose average perplexity of 15.9 is better (smaller) than that of "trained", which is 152.2. We justify the adoption of average perplexity with two perspectives: the ensemble perspective and the critic perspective. From the ensemble perspective, the average perplexity is aligned with the cross-entropy objective of the ensemble of the LLMs. From the critic perspective, the LLMs leverage their complementary critical abilities to detect non-factual text segments by giving them high perplexities. Finally, the winning text segment is forwarded through all LLMs, except for the LLM that generated the winning text segment, to update their states before entering the next iteration. The winning text segment selected in our approach may not be optimal, and a natural improvement is to let each LLM generates its top-k text segments in step 1 using beam search.

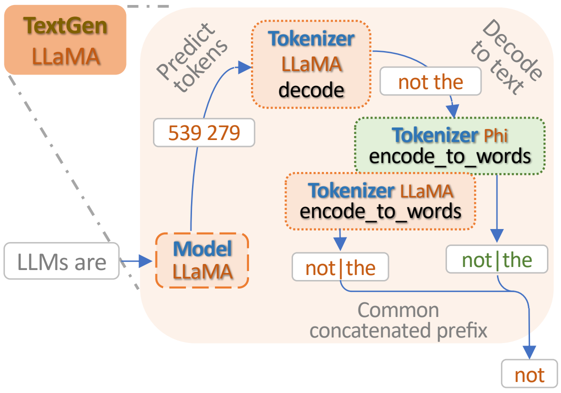


Figure 2: A contrived example illustrates our aligned text segments generation. In this example, the generated token sequence from LLaMA is first decoded into text, and then encoded and decoded by the tokenizers of two source LLMs into text segments: ["not", "the"] and ["no", "t", "the"], respectively. The aligned text segment "not" ends at the first common decodable boundary of all tokenizers, which helps to reduce biases in perplexity assessment due to the uneven text segment lengths across the tokenizers.

Table 1 shows the running results of our Cool-Fusion following the example in Figure 1 with a side-by-side comparison of the generation from the two source LLMs. As we can see from this example, the text generated by Cool-Fusion is seldom identical to that of its source LLM, since the divergence accumulates from the different text segment in each iteration. It seems that shorter text segments can result in more flexibility and lower perplexity; however, this is not necessarily the case. We will present two options for the selection of text segment length in Sections 2.2 and 2.3.

On the other extreme, the entire continuation can be used as a text segment, and sentence-level perplexity is employed to select (rerank) the the best continuation. In our Cool-Fusion approach, we can simultaneously employ an iterative fine-grained text segment selection and a coarse-grained sentence level reranking at the same time with almost no additional overhead. We let each source LLM independently predict a continuation segment in the same batch as each fine-grained text segment, with an additional overhead only on packing their key-value caches together. Then, we obtain k individual continuations in addition to a jointly predicted continuation. Our experiment results show that reranking these $k+1$ continuations on their average perplexities can lead to substantial improvements over using the joint prediction alone.

Name	Model ID	Parameters	Vocab size	Tokenizer category
LLaMA-3 (Touvron et al., 2023)	meta-llama/Meta-Llama-3-8B-Instruct	8B	128,256	LLaMA-3
MPT (Team, 2023)	mosaicml/mpt-7b-instruct	7B	50,277	LLaMA-3
LLaMA-2 (Touvron et al., 2023)	meta-llama/Llama-2-7b-chat-hf	7B	32,000	LLaMA-2
Phi-3 (et al, 2024)	microsoft/Phi-3-mini-128k-instruct	3.8B	32,038	LLaMA-2
OpenLLaMA (Geng and Liu, May 2023)	m-a-p/OpenLLaMA-Reproduce	7B	32,000	LLaMA-2
GLM-4 (GLM, 2024)	THUDM/glm-4-9b-chat	9B	151,343	OTHER
ChatGLM-3 (Zeng et al., 2023)	THUDM/chatglm3-6b	6B	64,796	OTHER
ChatGLM-2 (Zeng et al., 2023)	THUDM/chatglm2-6b	6B	64,787	OTHER
Baichuan2 (Baichuan., 2023)	Baichuan2-7B-Chat	7B	125,696	OTHER

Table 2: Source LLMs used in our experiments are divided into three categories in the last columns according to how to obtain their shortest text segments.

2.2 Shortest Text Segment

We will discuss two implementations of the TextGen component, as shown in Figure 2. We prefer shorter text segments since they suggest finer-grained token selection and are therefore more likely to obtain a similar token sequence from an optimal token level ensemble approach. In this subsection, we will demonstrate how to generate the shortest possible text segments.

We define the shortest text segment as a text that can be decoded from the shortest token sequence generated by a greedy decoding process. Not all token sequences are decodable. For instance, LLaMA-2 uses three Unicode bytes as the tokens to encode a single Chinese character, so the first one or two of these tokens cannot be decoded.

When decoding a token sequence into text, some tokenizers return additional information about the sequence of words in the text and the tokens that decode each of these words. In this case, we adopt the words as our shortest text segments since they are the minimal semantic units underlying a token sequence, although sometimes a word cannot be further divided into decodable subwords.

Specifically, tokenizers from the LLaMA-3 tokenizer provide a `word_ids` function that returns the IDs of the words in each decoded text, and a `word_to_tokens` function that returns the indexes of the first and last tokens for each word id. Tokenizers derived from the LLaMA-2 tokenizer provide an `offsets` property for each token, which contains the starting and ending character indexes in the text for the word decoded from the token.

For tokenizers that return decoded text without information about words, we derive shortest text segments as follows. Iteratively, we build a token sequence that initially contains only the next predicted tokens. Subsequently, a new next token is appended to the token sequence in each new iteration. In some iterations, if we can decode the current token sequence into a text that can be encoded back into the same token sequence, the decoded text is the shortest decodable text segment we need.

Metric	LLaMA-3	Phi-3	Cool$_2$	GLM-4
Avg. perplexity	1.48	1.35	-	1.46
Accuracy	0.6914	0.6831	0.7233	0.6338
	Rerank$_3$	**Cool$_{-align}$**	**Cool**	**Cool+R**
Avg. perplexity	-	-	**1.29**	-
Accuracy	0.7779	0.7445	0.7468	**0.812**

Table 3: Averaged perplexities and accuracies in the GSM8K datasets (Section 3.4).

Table 2 lists the LLMs that we will use in our experiments and their categories according to the above discussion.

2.3 Aligned Text Segments

In this subsection, we propose a better type of text segment to reduce the bias in the average perplexity used to select the best text segments generated by the source LLMs.

Different tokenizers may produce their shortest text segments of varying lengths. For instance, the text “Multi-tasking” is divided by LLaMA-3 and LLaMA-2 tokenizers into words [“Multi”, “-tasking”] and [“Multi-tasking”], respectively.

On the other hand, perplexity, as a measure of how well a given model generates a continuation given a context, is the most widely used metric for evaluating language models due to simplicity and its alignment to the cross-entropy (CE) loss used for the next-token prediction objective. Since the latter confers multi-step reasoning ability to LLMs, we believe that perplexity is not only a measure of language fluency but also an indicator of inference correctness to some extent. The perplexity (PPL) of a token sequence s is proportional to the average of the logits of the tokens:

$$PPL_u(s) = exp(\frac{1}{|s|}\sum_{s_i \in s} - \log p_u(s_i)), \quad (1)$$

where $\log p_u(s_i)$ is the logit output of the LM u for each token s_i.

However, as a measure of uncertainty, $-\log p_u(s_i)$ tends to be larger at the first token of each word. This is comparable to the observation on larger scales that the perplexity of the first word in a sentence is usually larger than those of the words following it, and that the perplexity of the first sentence in a paragraph is larger than those of the sentences following it. Here is a concrete example: in the LLaMA-3 8B model, the $-\log p_u(s_i)$'s values for the three tokens ["Multi", "-task", "ing"] are -2.66, -9.13, -14.69, respectively. Clearly, the model is more uncertain about the first token, and is more confident about the second token "-task" given the first token being "Multi".

Therefore, using perplexity as an assessment will bias towards longer text segments, and towards LLMs with tokenizers that generate text segments of larger average lengths. Suppose both LLaMA-3 and LLaMA-2 will predict the word "Multi-tasking", and their next text segments should be tied. Based on perplexity, however, the text segment "Multi-" from LLaMA-3 (-2.66) is regarded as worse than "Multi-tasking" from LLaMA-2 ($(-2.66 - 9.13 - 14.69)/3 = -8.83$).

To mitigate this problem, we must reduce the discrepancies between the average lengths of the text segments generated by different source LLMs. To this end, we define a new aligned text segment for each source LLM as the shortest text segment that is generated by the LLM and is decodable by the tokenizers of all source LLMs.

2.4 Incremental Encoding & Decoding

Both shortest text segments and aligned text segments require more frequent invocations of tokenizers than conventional decoding. In this subsection, we investigate an implementation issue about how to make decoding and encoding more efficient. First, let us examine the problem that not all tokenizers encode and decode incrementally, that is, the next text cannot be decoded solely from the next tokens, which results in significant delays as the encoding/decoding sequence increases.

It is expected that the text input and the tokenized sequence are reversibly convertible. For multilingual tokenizers, whitespace is treated as a normal symbol and preserved in the segmented tokens, allowing us to de-tokenize text without relying on language-specific rules such as: there is whitespace between two English words, but not between Chinese and Japanese words.

An instance of non-incremental encoding and decoding is the LLaMA-2 tokenizer, whose `encode` and `decode` functions are context-dependent and require complete token sequences or text to work correctly. For example, the `decode` function in LLaMA-2 decodes the token [839] into "If" or " If" (with a preceeding space) depending on whether or not the token is the first token in the token sequence. Therefore, we cannot encode a new token in isolation, and the conventional method to decode a few new tokens is to encode the concatenation of all previous tokens and the new tokens, which makes it inefficient for long sequences.

To enable incremental decoding, we only prepend the tokens belonging to the previous k decoded words to the new tokens, and we remove these k words from the decoded text after decoding. We handle incremental encoding similarly by prepending k decoded words to new text to be encode. Thus, we can encode and decode with constant computational complexity regardless of the context length. We empirically found that $k = 4$ ensures correctness for both incremental encoding and decoding.

3 Experiments

Our experiments are conducted in a challenging scenario for LLM fusion, where the tokenizers of the source LLMs have very different token vocabularies and define text segments differently. A wide range of datasets is used to make our evaluations comprehensive. The questions that we want to answer from our experiments include: How does Cool-Fusion's performance compare with recent work? What are the contributions of its individual components, such as fine-grained perplexity-based text segment selection, shortest text segments, and aligned text segments? Is it a general method that performs well in various domains? Can it improve multilingual performance? Does its performance persist when fusing different LLMs?

3.1 Settings and Datasets

We conduct experiments with several recent state-of-the-art open-source LLMs as our source LLMs, as listed in Table 2.

To assess the performance of Cool-Fusion, we conduct experiments using the LM-Evaluation-Harness (Gao et al., 2023), a benchmark framework designed to evaluate LLMs' few-shot capabilities across various domains. We use its default settings, except for employing 3-shot prompting in

Method	src LLMs	Training	GSM8K
LLaMA2-7B-Chat	-	-	24.64
ChatGLM2-6B	-	-	30.78
Baichuan2-7B-Chat	-	-	29.95
InternLM-7B-Chat	-	-	32.30
TigerBot-7B-Chat-V3	-	-	27.29
Vicuna-7B-V1.5	-	-	18.88
ChineseAlpaca2-7B	-	-	13.12
MBR	7 above	-	36.47 (+4.17)
PairRanker	7 above	Ranker	39.58 (+7.28)
LLM-Blender	7 above	Merger	34.80 (+2.50)
EVA	7 above	Vocab Map	42.91(+10.61)
LLaMA2-7B-Chat	-	-	19.3
ChatGLM2-6B	-	-	25.9
Baichuan2-7B-Chat	-	-	26.9
Cool	3 above	Training-free	33.5 (+6.6)

Table 4: We compare our results with recent model fusion algorithms that use training. Data in the first two blocks are from (Xu et al., 2024), and those in the last two blocks are our results. Please note that this is not an apple-to-apple comparison: (1) we are comparing a training-free method to those that require different types of training: PairRanker (Chen et al., 2023), LLM-Blender (Jiang et al., 2023), EVA (Xu et al., 2024), (2) the results of our Cool-Fusion are based on fusing three source LLMs due to our resource limitations, and (3) the scores of our source LLMs are on average more than 4 points lower than those reported in (Xu et al., 2024) due to differences in experimental settings.

all experiments. We conducted experiments on the following greedy text generation tasks.

CoQA (Reddy et al., 2019) requires understanding a text passage and answer a series of interconnected questions that appear in a conversation.

DROP (Dua et al., 2019) is a crowdsourced, adversarially-created, 96k-question benchmark, in which a system must resolve references in a question, perhaps to multiple input positions, and perform discrete operations over them (such as addition, counting, or sorting).

TriviaQA (Joshi et al., 2017) is challenging as the answers for a question may not be directly obtained by span prediction and the context is long.

MATH (Hendrycks et al., 2021) is a dataset of 12,500 challenging competition mathematics problems. Each problem in MATH has a full step-by-step answer derivations and explanations.

GSM8K (Cobbe et al., 2021) is a dataset of high quality linguistically diverse grade school math word problems, that take between 2 and 8 steps of elementary calculations ($+ - \times \div$) to solve.

MGSM (Saparov and He, 2023) stands for Multilingual Grade School Math Benchmark, where the same 250 problems from GSM8K are each translated in 10 languages other than English.

Unscramble is used for evaluating language models' ability to handle text manipulation tasks.

Method	Training	GSM8K
FuseLLM-7B [*]	Yes, via distillation	13.8
Cool (ours)	No	12.3

Table 5: Comparison on the GSM8K dataset. Both methods use source LLMs: LLaMA2-7B-Chat (Touvron et al., 2023), MPT 7B (Team, 2023), and OpenLLaMA-7B (Geng and Liu, May 2023).

It contains several datasets that require creative generation (Anagrams), pattern manipulation (Cycle Letters), and contextual awareness (Random Insertion) rather than structured reasoning.

3.2 Ablation study

We compare Cool-Fusion with several source LLMs as baselines in Table 3. Cool$_2$, which fuses LLaMA-3 and Phi-3, immediately increases accuracy by 4.6% and 5.9%, respectively. Cool, which fuses LLaMA-3, Phi-3, and GLM-4, further increases the increments to 8.0%, 9.3%, and 17.8%, respectively. This verifies the effectiveness of our fine-grained perplexity-based reranking.

$Cool_{-align}$ is an implementation using shortest text segment (Section 2.2), while $Cool$ implements aligned text segment (Section 2.3). $Cool_{-align}$ leads to a 0.3% relative decrement, suffered from occasional bias in perplexity assessment.

Rerank is a simple reranking method, where each source LLM predicts a continuation individually, and these continuations are reranked using their average perplexities from all source LLMs. Rerank turns out to be very effective and it obtains a 12.5% increment over LLaMA-3. Cool+R is a combination of Cool-Fusion and Rerank, which achieves a significant accuracy improvement of 17.4% over LLaMA-3 and 4.4% over Rerank.

3.3 Compare with Other LLM Fusion Methods.

We compare several existing fusion algorithms in Table 4 on GSM8K. Due to resource limitations, we can only fuse three LLMs in our experiments. It is difficult to make an apple-to-apple comparison. Due to differences in experimental setting, our scores for the source LLMs are, on average, 4 points lower than those reported in (Xu et al., 2024). Table 4 shows that, although using only the three source LLMs and requiring no training, our Cool-Fusion reports a comparable score increment to existing methods that require different types of training to fuse all of the seven source LLMs.

The results in Table 5 demonstrate that Cool

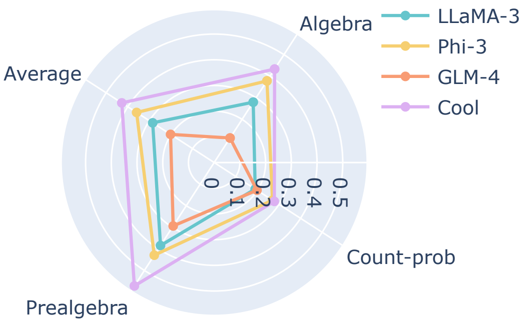


Figure 3: Accuracies in the Math dataset.

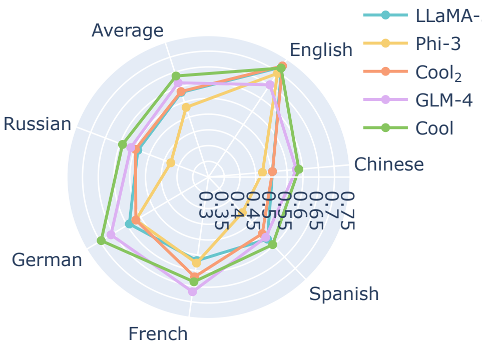


Figure 4: Accuracies in the multilingual GSM datasets.

achieves competitive performance (12.3) without requiring any training, while FuseLLM-7B, which relies on distillation, achieves only a marginally higher accuracy (13.8).

These results underscores the efficiency and practicality of our training-free approach across different sets of models, making it a compelling alternative to resource-intensive methods.

3.4 Cross-domain Performances

Next, we examine the general performance of Cool-Fusion in three different domains, where not all of source LLMs used have good performance. On the Q&A datasets (Figure 5), LLaMA-3 performs best, but GLM-4 fails to follow the output format in our 3-shot prompts. On the other hand, in the multilingual GSM datasets (Figure 4), the overall performance of GLM-4 is the best, while Phi-3 does not perform well on multilingual data (et al, 2024). Finally, on the Math dataset (Figure 3) and the Unscramble dataset (Figure 6), Phi-3 is the best performer, and the other two LLMs lag behind with significant gaps. It is therefore challenging to fuse LLMs in these datasets where the performance of the source LLMs differs and fluctuates dramatically. Cool-Fusion either outperforms all source LLMs or is comparable to the best performer and not being affected by the poorer ones, which shows that Cool-Fusion is a stable method for fusing source LLMs across different domains.

Comparing the performance of Cool-Fusion in Table 3 with that in Figure 3, we can see that Cool-Fusion performs much better on GSM8K than on multilingual GSM, although the latter is a translated subset of the former. This is probably because multilingual GSM contains a larger proportion of hard problems than in GSM8K.

3.5 Summary of Experiments

In this section, we verify the effectiveness of the components in Cool-Fusion through ablation studies, which shows that our Cool-Fusion achieves significant improvements over the source LLMs on challenging tasks. It is able to achieve further advances when combined with other approaches, and persistently being better or comparable to the best-performing source LLMs even when some of them exhibit deteriorated performance. Our Cool-Fusion shows comparable performance with recent state-of-the-art LLM fusion methods that require training, and its performance persist when fusing different LLMs.

4 Related Work

Cool-Fusion aligns with established ensemble principles: (1) the Condorcet Jury Theorem, which justifies more independent models and (2) the bias-variance tradeoff, which suggests reduced variance with more models. In this section, we summarize prior work on model and LLM fusion. To our knowledge, prior work on fusion of heterogeneous LLMs involve different types of training.

Reranking methods first generate multiple candidates via probabilistic sampling, or by prompting LLMs. The quality of the candidates are then assessed using different scoring methods (Ravaut et al., 2022; Jiang et al., 2023).

Alignment matches the units of prediction from multiple models, i.e. the vocabularies of different LLMs. Since finding the optimal alignment is a combinatorial optimization problem, alignment between vocabulary is still an open problem. FuseLLM (Wan et al., 2024a), FuseChat (Wan et al., 2024b), and Specialized (Fu et al., 2023) use the edit-distance between tokens to map token distributions between LLMs, while EVA (Xu et al., 2024) trains a vocabulary projection matrix.

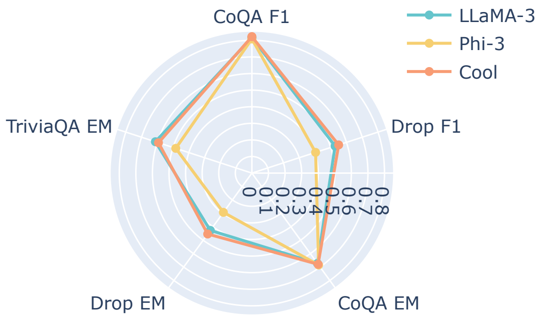


Figure 5: F1 & EM in the four Q&A datasets.

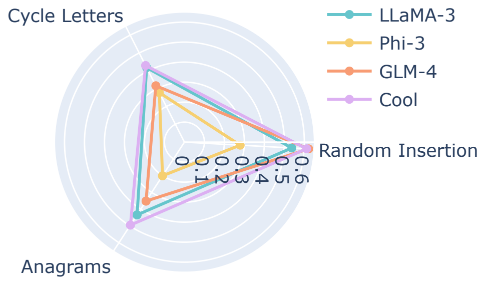


Figure 6: Accuracies in the Unscramble dataset.

However, it is unclear if the alignment approaches (Mavromatis et al., 2024; Xu et al., 2024), which assume substantial amount of common tokens across vocabularies, will work for Unicode vocabularies, where different tokenizers may share little portion of their symbols: Unicode bytes are the basic symbols in Phi-3 (et al, 2024), while subword tokenization for Chinese (Si et al., 2023) uses glyph or pronunciation encoding.

Ensembling approaches conventionally require the source models to have the same token vocabulary, which can be partially relaxed by vocabulary alignment (Mavromatis et al., 2024). LLM-Blender (Jiang et al., 2023) ensembles the outputs from several source LLMs by firstly using a fine-tuned ranking model to predict the top-ranked outputs, then it uses another fine-tuned LLM to generates a fused output. EVA (Xu et al., 2024) proposes to ensemble LLMs via a pre-trained vocabulary alignment matrix to enable a fine-grained token-level ensemble at each generation step.

Weight average. Researchers do not limit themselves to predictions, e.g. logics. Model soups (Wortsman et al., 2022), which average the weights of multiple models fine-tuned with different hyperparameter configurations, often improves accuracy and robustness. PAPA (Jolicoeur-Martineau et al., 2024) obtains a strong single model by training a population of models and averaging them once-in-a-while or slowly pushing them toward the average. These methods require no training data, but the models to fuse must be of the same architecture.

Knowledge distillation. Alpaca (Taori et al., 2023) used text-davinci-003 to generate the instruction data to distill a 7B LLaMA (Touvron et al., 2023) model to reduce the cost of training LLMs from scratch. FuseLLM (Wan et al., 2024a,b) applies cost-effective distillation to merge pre-trained LLMs into a more potent model.

Multi-agent approaches enable an orchestration of a collection of LLM modules working together, each with different potentials. MetaGPT (Hong et al., 2024) encodes a standardized operating procedure (SOP) for software development into a prompt sequence. It breaks down complex tasks into subtasks, allowing agents with different domain expertise–such as architecture design and code debugging–to work harmoniously.

Beam search is importance for generation tasks like summarization and machine translation, and beam search with a single LLM often outperforms multi-LLM fusion on tasks like machine translation (Farinhas and et al., 2023) (e.g., MBR decoding achieves state-of-the-art results). This paper focuses on challenging generation tasks that require deep understanding and reasoning. Greedy generation was chosen for its simplicity and effectiveness, as it is widely used in practice and has been shown to perform well for large language models (LLMs).

Others Contrastive decoding (Li et al., 2023b) exploits the contrasts between expert and amateur LLMs by choosing tokens that maximize their log-likelihood difference to amplify the good expert behavior and diminish the undesired amateur behavior. CALM (Bansal et al., 2024) introduces cross-attention between models to compose their representations and enable new capabilities.

5 Conclusion and Future Directions

In this work, we propose Cool-Fusion, a simple yet effective approach that fuses the knowledge of heterogeneous source LLMs. Extensive experiments with challenging datasets and strong source LLMs verify the persistent improvements and robustness of our proposal. Future work can focus on improving inference speed, including streamlining different inference processes to fill GPU vacancies waiting for communication, parallelizing tokenizers to find out whether a text segment is decodable by all tokenizers, using longer text segments to reduce communication overhead between LLMs.

Limitations

The inference speed of our implementation of Cool-Fusion is about six times slower than that of a standard LLM, mainly due to the additional communication among LLMs and the frequent invocation of tokenizers. Further optimizations, such as streamlining different inference processes or implementing parallel tokenizers, might increase the speed of Cool-Fusion.

Due to resource limitations, we only conduct experiments with two and three source LLMs. We used the automatic metrics that come with the Evaluation Harness (Gao et al., 2023). Human or GPT-4 evaluations could provide us with more reliable and comprehensive results.

Ethical Statement

This work fully complies with the ACL Ethics Policy. We declare that there are no ethical issues in this paper, to the best of our knowledge.

References

Baichuan. 2023. Baichuan 2: Open large-scale language models. *Preprint*, arXiv:2309.10305.

Rachit Bansal, Bidisha Samanta, Siddharth Dalmia, Nitish Gupta, Sriram Ganapathy, Abhishek Bapna, Prateek Jain, and Partha Talukdar. 2024. LLM augmented LLMs: Expanding capabilities through composition. In *The Twelfth International Conference on Learning Representations*.

Lingjiao Chen, Matei Zaharia, and James Zou. 2023. Frugalgpt: How to use large language models while reducing cost and improving performance. *Preprint*, arXiv:2305.05176.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. 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)*, pages 2368–2378, Minneapolis, Minnesota. Association for Computational Linguistics.

Marah Abdin et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *Preprint*, arXiv:2404.14219.

Antonio Farinhas and et al. 2023. An empirical study of translation hypothesis ensembling with large language models. *Preprint*, arXiv:2310.11430.

Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. 2023. Specializing smaller language models towards multi-step reasoning. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. A framework for few-shot language model evaluation.

Xinyang Geng and Hao Liu. May 2023. Openllama: An open reproduction of llama.

Team GLM. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *Preprint*, arXiv:2406.12793.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023. LLM-blender: Ensembling large language models with pairwise ranking and generative fusion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14165–14178, Toronto, Canada. Association for Computational Linguistics.

Alexia Jolicoeur-Martineau, Emy Gervais, Kilian FATRAS, Yan Zhang, and Simon Lacoste-Julien. 2024. Population parameter averaging (PAPA). *Transactions on Machine Learning Research*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Weishi Li, Yong Peng, Miao Zhang, Liang Ding, Han Hu, and Li Shen. 2023a. Deep model fusion: A survey. *arXiv preprint arXiv:2309.15698*.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. 2023b. Contrastive decoding: Open-ended text generation as optimization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12286–12312, Toronto, Canada. Association for Computational Linguistics.

Costas Mavromatis, Petros Karypis, and George Karypis. 2024. Pack of llms: Model fusion at test-time via perplexity optimization. *Preprint*, arXiv:2404.11531.

Mathieu Ravaut, Shafiq Joty, and Nancy Chen. 2022. SummaReranker: A multi-task mixture-of-experts re-ranking framework for abstractive summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4504–4524, Dublin, Ireland. Association for Computational Linguistics.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2019. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.

Abulhair Saparov and He He. 2023. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In *The Eleventh International Conference on Learning Representations*.

Chenglei Si, Zhengyan Zhang, Yingfa Chen, Fanchao Qi, Xiaozhi Wang, Zhiyuan Liu, Yasheng Wang, Qun Liu, and Maosong Sun. 2023. Sub-character tokenization for Chinese pretrained language models. *Transactions of the Association for Computational Linguistics*, 11:469–487.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

MosaicML NLP Team. 2023. Introducing mpt-7b: A new standard for open-source, commercially usable llms. Accessed: 2023-05-05.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *ArXiv*, abs/2302.13971.

Fanqi Wan, Xinting Huang, Deng Cai, Xiaojun Quan, Wei Bi, and Shuming Shi. 2024a. Knowledge fusion of large language models. In *The Twelfth International Conference on Learning Representations*.

Fanqi Wan, Ziyi Yang, Longguang Zhong, Xiaojun Quan, Xinting Huang, and Wei Bi. 2024b. Fusechat: Knowledge fusion of chat models. *arXiv preprint arXiv:2402.16107*.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 23965–23998. PMLR.

Yangyifan Xu, Jinliang Lu, and Jiajun Zhang. 2024. Bridging the gap between different vocabularies for LLM ensemble. In *2024 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. 2023. GLM-130B: an open bilingual pre-trained model. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*.