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 facilitate their complementary strengths. One of the challenges on model fusion is high computational load, i.e. to fine-tune or to align vocabularies via combinatorial optimization. To this end, we propose *Cool-Fusion*, a simple yet effective approach that fuses the knowledge of heterogeneous source LLMs to leverage their complementary strengths. *Cool-Fusion* is the first method that does not require any type of training like the ensemble approaches. But unlike ensemble methods, it is applicable to any set of source LLMs that have different vocabularies. The basic idea is to have each source LLM individually generate tokens until the tokens can be decoded into a text segment that ends at word boundaries common to all source LLMs. Then, the source LLMs jointly rerank the generated text segment and select the best one, which is the fused text generation in one step. Extensive experiments are conducted across a variety of benchmark datasets. On *GSM8K*, *Cool-Fusion* increases accuracy from three strong source LLMs by a significant 8%-17.8%.

1 Introduction

Different large language models (LLMs) respectively exhibit diverse strengths and weaknesses due to various factors, such as datasets used for pre-training and fine-tuning, architectures, optimizers, hyper-parameters, and training methodologies. Recent work (Jiang et al., 2023) 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 human preference alignment.

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 cost on 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 for vocabulary alignment (Wan et al., 2024a,b; Fu et al., 2023; Xu et al., 2024). As a result, existing fusion approaches are unsuitable for researchers and practitioners for whom training or fine-tuning is cost-prohibitive, or for application scenarios that require rapid deployment.

Aiming at a general LLM fusion approach that is applicable to a set of source LLMs with any tokenizers and is cheaper and faster to deploy, we propose *Cool-Fusion*, which fuses the knowledge of heterogeneous LLMs without any training. The core of our algorithm is to combine the source LLMs to rerank text segments they generated individually, instead of using the ensemble of LLMs as a token generator, which might not be possible with different vocabularies. *Cool-Fusion* tries to fuse the knowledge as early as possible on each generation step, which consists of two major components: (1) An individual tokens generation to decode text segments that end at word boundaries common to all source LLMs, which results in text segments that can be properly assessed by every source LLM. (2) A joint reranking by all source LLMs to select the best text segment in terms of average perplexity. 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, no training of any type is required, and it can start to generate texts as soon as we have the source LLMs. Therefore, we do not

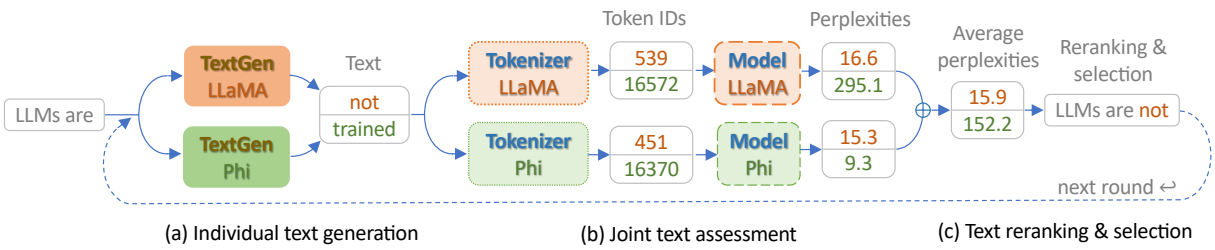

Figure 1: An illustration of *Cool-Fusion*. The TextGen component is illustrated in Figure 2.

need to worry about the problems associated with training, such as overfitting the tuning distribution, insufficient hyper-parameter tuning, or loss of generic ability (Fu et al., 2023).

- Availability: Based only on pure inference, *Cool-Fusion* can be accessed by a larger range of budget-limited researchers and practitioners. Additionally, *Cool-Fusion* can also be extended to closed source LLMs, as long as they have the required APIs as shown in Table 1). We will make our source code publicly available.

- Scalability: Inferences in the two components in each generation step of *Cool-Fusion* are completely independent and can run in parallel, and only a small amount to texts and scores are required to be exchanged between the components. Therefore, given a cluster of large enough number of GPU nodes connected by ordinary computer networks, *Cool-Fusion* is scalable to any number of LLMs in parallel, with negligible increase in generation delay in theory.

- Superior performance: As a result of pursuing for a lower fine-grained average perplexity, *Cool-Fusion* outperforms its source LLMs and baselines in 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 use three recent state-of-the-art open-source LLMs, i.e. *LLaMA-3 8B* (Touvron et al., 2023), *Phi-3 mini* (et al, 2024), and *GLM-4 9B* (Zeng et al., 2023), as our source LLMs. Our results demonstrate that *Cool-Fusion* significantly outperforms the individual source LLMs on challenging tasks. On the *GSM8K* dateset, *Cool-Fusion* increase the prediction accuracy from the best performing source LLM *LLaMA-3 8B* by a significant margin of 8% and the worst performing source LLM by 17.8%. When combined with reranking, we further obtain a 17.4% increase over *LLaMA-3 8B*.

2 *Cool-Fusion*: Fuse LLMs without Fine-tuning

As illustrated in Figure 1, *Cool-Fusion* has a text generation loop, where a text segment is generated at each iteration of the loop. In the example in the figure, the two source LLMs to be fused are *LLaMA* and *Phi*, and the input context text is "LLMs are". Each iteration in the text generation loop contains three steps: (a) individual text generation, (b) joint text assessment, and (c) text reranking and selection.

In the first step, there is a *TextGen* component for each source LLM, which generates a text segment independently. In Figure 1, the text segments generated by the two *TextGen* components are "not" and "trained", respectively. We will discuss the *TextGen* component in the next sub-section.

In the second step, each of the source LLM first individually computes the perplexities for all generated text segments. As shown in Figure 1, the two LLMs first encode text segments "not" and "trained" into token IDs using their own tokenizers, and then compute the perplexities using these token IDs. The perplexities computed by *LLaMA* are 16.6 and 29.5, and those by *Phi* are 15.3 and 9.3. These scores are then averaged over the same LLM, resulting in an average perplexity of 15.9 for "not" and 152.2 for "trained".

In the third step, the average perplexities for the text segments are used to rerank them, and the text segment with the minimal average perplexity is selected as the final text segment for current iteration of the text generation loop. In the following, we will formally describe the first two steps with greater details.

2.1 Aligned Text Segments

Generated by each source LLM individually, the text segments need to be decodable by the tokenizers of all source LLMs before the computation of their perplexities. Secondly, the text segments should be assessed by an unbiased perplexity measurement.

Before presenting our text segmentation approach, we want to show that neither token boundaries nor word boundaries are idea for use as the boundaries for text segments. We do not assume any particular tokenizers, but we assume tokenizers where the string text input and the tokenized sequence are reversibly convertible. In these tokenizers, whitespace is treated as a normal symbol and preserved in the segmented tokens, so that we can de-tokenize text without relying on language-specific rules, e.g. there is no whitespace between Chinese and Japanese words.

In ensemble approaches (Mavromatis et al., 2024), the next predicted token is the one with the maximum sum of logits, which requires that the source LLMs share the same vocabulary, and is therefore not applicable to arbitrary source LLMs. For instance, the text "Multi-tasking" is encoded by *LLaMA-3* and *Phi-3* into the following tokens, respectively:

LLaMA-3: `['Multi', '-task', 'ing']`

Phi-3: `['Multi', '-', 'task', 'ing']`

Unlike the above example where each token is readable, for tokenizers that use unicode bytes as their basic symbols, not all token sequence can be decoded to a readable text. For instance, *Phi-3* uses three unicode bytes as the tokens to encode a single Chinese character, so the first two of such three tokens are decodable.

Contemporary work (Mavromatis et al., 2024; Xu et al., 2024) try to align tokens across vocabularies, which faces two open challenges: (1) optimal alignment is a combinatorial optimization problem and (2) their assumption that the vocabularies share lexically similar basic symbols, or that a substantial amount of tokens are common across vocabularies, is not universal, since some tokenizers share little of their basic symbols. For example, *Phi-3* uses unicode bytes as basic symbols (et al, 2024), while sub-word tokenization for Chinese (Si et al., 2023) may use glyph or pronunciation encoding.

To summarize, token-level boundaries cannot be use for our text segements, since not all token sequence are decodable or translatable to the token sequence of every other tokenizer.

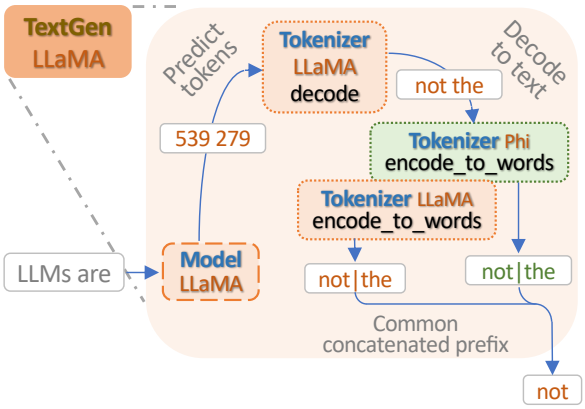

Figure 2: Illustration of the individual text generation component (Section: 2.2).

Next, we consider word-level boundaries. Tokenizers explicitly or implicitly provide tokens-to-word mapping, from which we can effectively obtain word-level boundaries. However, tokenizers may define word-level boundaries differently. For instance, text "Multi-tasking" is encoded by *LLaMA-3* and *Phi-3* into tokens that are grouped into the following words, respectively:

LLaMA-3: `['Multi', '-tasking']`

Phi-3: `['Multi-tasking']`

On the other hand, perplexity is defined as an average value:

$$PPL(s) = exp(\sum_{tok_i \in s} -\log p(tok_i)), \quad (1)$$

where tok_i is a token in sequence s, $\log p(tok_i)$ is the logit output of the LM for tok_i. As a measure of uncertainty, $-\log p(tok_i)$ tents to be larger at the first words of each sentence or paragraph. At a smaller scale, $-\log p(tok_i)$ is also typically larger at the first token of each word. For instance, the tokens for word "Multi-tasking" and their corresponding $-\log p(tok_i)$ in *LLaMA-3* are:

tok_i: `['Multi', '-task', 'ing']`

$-\log p(tok_i)$: [-2.66, -9.13, -14.69].

Since tokenizers may not agree on their word boundaries, for text segments ending at the word boundaries of different tokenizers, the use of perplexity is biased in favor of longer words. For example, suppose two LLMs both have "Multi-tasking" in their mind, the one predicts "Multi-tasking" as its next word wins over the one predicts "Multi" in terms of perplexity.

We propose to require text segments to end at common word-level boundaries for all tokenizers,

which means the text segment generated by each LLM does not ends at the word-level boundaries of its own tokenizer, but at those common to all tokenizers. As an instance following our previous example, the first text generated by *LLaMA-3* will be `'Multi-tasking'` instead of `'Multi'`, since the latter does not ends at the word-level boundaries of *Phi-3*'s tokenizer. Thus, our text segments prevent uneven word lengths across tokenizers and biased perplexities as a consequence.

To sum up, to be encodable from tokens, text segments must end at word-level boundaries of its own LLM, while aligning to the common word-level boundaries of all LLMs avoids biased perplexity assessments. We will verify the effectiveness of the latter with experiments in Section 3.3.

2.2 Individual Text Generation

Cool-Fusion generates its next text segment by first having each source LLM generate a text segment individually before selecting the best one from them.

To generate a text segment that ends at the common word-level boundaries common to the tokenizers of all k LLMs, each source LLM m^u follows these steps: (1) keeps generating tokens $t_1^u, t_2^u, \ldots, t_i^u$, (2) decodes the currently generated token sequence $[t_1^u, t_2^u, \ldots, t_i^u]$ into a text segment s_i^u using m^u's tokenizer, (3) encode s_i^u into a word sequences $[w_1^v, w_2^v, \ldots, w_j^v]$ using the tokenizer of each source LLM m^v for $1 \leq v \leq k$, (4) if there exists a text sequence s^u that is equal to the concatenation of some prefix for every sequence $[w_1^v, w_2^v, \ldots, w_{j-1}^v]$ for $1 \leq v \leq k$, then s^u is the text segment generated by m^u in the current iteration. Otherwise, m^u continue with generating its next tokens by going back to step (1).

In step (4) above, we require that s^u equals the concatenation of a prefix of $[w_1^v, w_2^v, \ldots, w_{j-1}^v]$ instead of $[w_1^v, w_2^v, \ldots, w_j^v]$, since the last word w_j^v might not be a complete word according to the word-level boundary of tokenizer m^v. For instance, if s_i^u is `'Multi'` and m^v is *Phi-3*, the tokenizer of *Phi-3* will only be able to decode s_i^u as $[w_1^v]$ = `['Multi']`, where w_1^v is not a complete word for *Phi-3*.

Figure 2 is an example that illustrate the individual text generation process above. Given an input text "LLMs are", m^u (*LLaMA-3*) generates tokens [539, 279], which are then decoded into text segment s_i^u = "not the". s_i^u is then encoded by the tokenizer of every LLM (*LLaMA-3* and *Phi-3*) into word sequence $[w_1^v, w_2^v, \ldots, w_j^v]$, respectively.

#	Name	Description
M1	open	Start a new section with a text context. Return a handle of the current model state (key-values cache).
M2	next_token	Predict the next token given a temp model state. Discard the given temp model state and return the handle of a new temp model state.
M3	feed_tokens	Feed the model with a sequence of tokens. Return the perplexity of the tokens and a handle of a temp model state.
M4	update_state	Set the current model state to a given temp model state.
T1	next_text	Generate the next aligned text segment (Section 2.2).
T2	encode_text _into_words _and_tokens	Given a text string, encode it into a token sequence and a word sequence (Section 2.1).
T3	decode_tokens	Decode a token sequence to get a text.

Table 1: APIs of each worker node (Section 2.4).

In this example, both word sequences are ["not", "the"], and we have s^u = "not".

2.3 Joint Text Assessment

Now, we have the generated segment s^u from each source LLM m_u, we will rerank them and select the one with the smallest average perplexity as the final generated text in the current iteration.

We use the perplexity (PPL) between the predicted distribution and the one-hot distribution of the token sequnce encoding each predicted text segment s_u as the assessment from each LLM m_v, which is defined as:

$$PPL_v(s_u) = exp(\sum_{tok_i \in s_u} -\log p_v(tok_i)), \quad (2)$$

where tok_i is the i-th token on the token sequence encoding s_u with the tokenizer of m_v, and $logp_v(tok_i)$ is the logit value for tok_i predicted by the model of m_v.

Perplexity is a measure of how well a given model generate a continuation given a context, or how well is a continuation given a model and a context. Perplexity is the most widely used metric for evaluating language models due to simplicity and its alignment to the training loss for causal language models: the cross-entropy (CE) loss equals the $\log$ of PPL.

We use the average perplexity of the source LLMs as a metric to rerank each text segments that are individually predicted by each source LLMs at each iteration. Our method can be regarded as an ensemble of LLMs for reranking, instead of for generation. The best text segment s^* with the

Name	Model ID	Parameters	Vocab size
LLaMA -3 (Touvron et al., 2023)	meta-llama/Meta-Llama-3-8B-Instruct	8B	128,256
Phi-3 (et al, 2024)	microsoft/Phi-3-mini-128k-instruct	3.8B	32,038
GLM-4 (Zeng et al., 2023)	THUDM/glm-4-9b-chat	9B	151,343

Table 2: Source LLMs used in our experiments (Section 3.1).

smallest average perplexity is selected:

$$s^* = \arg\min_{s_u, 1 \le u \le k} \frac{1}{k} \sum_{1 \le v \le k} \mathsf{PPL}_v(s_u), \quad (3)$$

which averages over the perplexities $\mathsf{PPL}_v(s_u)$'s by model m_v's, and selects from all s_u's that are individually generated by models m_u's.

The winner text segment s^* will be broadcasted to update the model of all source LLMs. If s^* does not indicate end-of-sentence (*eos*) by the the LLM that generated it, *Cool-Fusion* will continue with its next iteration of individual text generation.

2.4 A Parallel Implementation

We have completed our explanation on *Cool-Fusion*. However, there are many independent generation steps in *Cool-Fusion*, which calls for an efficient parallel implementation. This sub-section illustrates our distributed implementation to show that *Cool-Fusion* can scale to k source LLMs without much increment in delay. We assume that the computational cost on the tokenizers are negligible compared with those of the models, and we focus on parallelizing the invocations of GPU operations.

For ease of presentation, we assume that the k source LLMs are running on a cluster with one master node and k worker nodes, each equipped with a single GPU. Each worker runs an instance of the model of a unique source LLM as well as multiple instances of tokenizers, one for every source LLM. This enables each worker to conduct the individual text generation as illustrated in Figure 2. The workers provide stateful services to the master via APIs as listed in Table 1.

To start to generate a continuation from a context, the master invokes *M1* in parallel on every worker with the context. *M1* returns a handle/reference to the current model state, which primarily contains the key-values cache of the model.

The master then enters the following text-generation loop, which predicts a stream of text segments, one at each iteration. At each iteration, the master runs the following steps, where each invocation of a worker API is done in parallel on every worker: (1) invokes *T1* to obtain a text segment s_u from every worker u, where *T1* internally invokes *M2* one or more times for tokens, and invokes *T2* and *T3* to check whether s^u aligns to the word boundaries common to all tokenizers; (2) gathers the s_u's from all workers, broadcasts them to invoke *M3* to obtain the $\mathsf{PPL}_v(s_u)$'s and a list of temporary model states from all workers; (3) select a winning text segment s^* according to Equation 3, and finally (4) update the models and the tokenizers by invoking *M4*, where *M4* updates the key-value cache of each model with the its previous temporary state resulting from invoking *M3* with s^*.

In *M3*, where we feed all s_u's into each model m_v to obtain $\mathsf{PPL}_v(s_u)$'s, we can feed s_u's in batch or concatenate them into a sequence with modified position IDs, such that *M3* handles all s_u's in parallel. Assuming invocations to models dominate the computational cost, the bottleneck is our approch is *M2*, which may be invoked multiple times in each iteration for the tokens to decode s_u. Therefore, the theoretical computation required by *Cool-Fusion* should be comparable to a parallel ensemble approach, which requires the same amount computation as in *M2*.

Currently, our implementation is only for prototyping our parallel implementation, and its efficiency can be further optimized from the following aspects: (1) reuse the extra tokens generated in *M2* in the next round of text segment generation, (2) the *forward* method in the LLM model can be rewritten to reduce the splitting and concatenation on the key-values caches, (3) the current encoding and decoding functions in the tokenizers may not be implemented in the most efficient way, and they can also be implemented using faster programming languages or using more optimized compilers, and (4) implement the aforementioned parallel invocation inside *M3*.

3 Experiments

In our experiments, we consider a challenging scenario for LLM fusion, where the tokenizers of the source LLMs define very different tokens and words. A wide range of datasets are used to make our evaluations comprehensive. The questions that we want to answer from our experiments include:

Metric	LLaMA-3	Phi-3	$Cool_2$	GLM-4	$Rerank_3$	$Cool_{-align}$	Cool	$Cool+Rerank_4$
Avg. perplexity	1.48	1.35	-	1.46	-	-	**1.29**	-
Accuracy	0.6914	0.6831	0.7233	0.6338	0.7779	0.7445	0.7468	**0.812**

Table 3: Averaged perplexities and accuracies on the *GSM8K* datasets (Section 3.3).

Dataset	LLaMA-3	Phi-3	GLM-4	Cool
Algebra	0.2797	0.3783	0.1137	**0.4330**
Count-prob	0.1899	0.27	0.1983	**0.2785**
Prealgebra	0.3846	0.4294	0.2939	**0.5718**
Average	0.2847	0.3592	0.2020	**0.4278**

Table 4: Accuracies on the *Math* dataset.

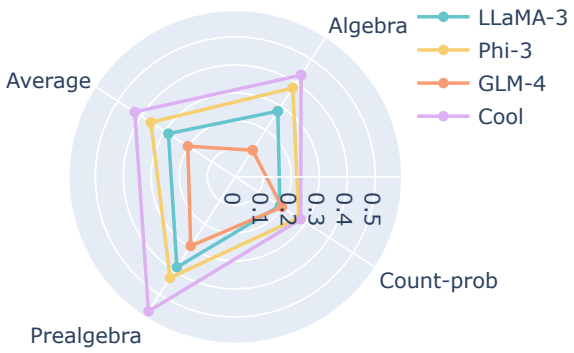

Figure 3: Accuracies on the Math dataset.

What are the contributions of *Cool-Fusion*'s individual components, e.g. aligned text segments, fine-grained perplexity-based reranking? How good is *Cool-Fusion* when combined with orthogonal algorithms? Is *Cool-Fusion* a general method that performs well in various domains? Can *Cool-Fusion* improve multilingual performance? Does the performance of *Cool-Fusion* persists when fusing different number of LLMs?

3.1 Source LLMs

We conduct experiments with three most recent state-of-the-art open-source LLMs as our source LLMs, including *LLaMA-3 8B* (Touvron et al., 2023), *Phi-3 mini* (et al, 2024), and *GLM-4 9B* (Zeng et al., 2023), which are compared as listed in Table 2. We use these LLMs not only because they are strong baselines, but also due to the fact that they have quite disjoint vocabularies: the common tokens between the vocabularies of *LLaMA-3* and *Phi-3* account for only 6.4% of their total tokens, those between *Phi-3* and *GLM-4* account for only 7.5%, and those between *LLaMA-3* and *GLM-4* account for 21.1%.

3.2 Settings and Datasets

To access 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 3-shot prompting and the default settings in all experiments. We conducted experiments on the following datasets, which evaluate LLMs on the task of greedy text generation.

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 very long.

MATH (Hendrycks et al., 2021) is a new dataset of 12,500 challenging competition mathematics problems. Each problem in MATH has a full step-by-step solution which can be used to teach models to generate 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 via human annotators in 10 languages other than English.

3.3 Results

We compare *Cool-Fusion* with several baselines in Table 3. $Cool_2$, which fuses *LLaMA-3* and *Phi-3*, immediately increases accuracy by 4.6% and 5.9%, respectively. $Cool_3$, which fuses *LLaMA-3*, *Phi-3*, and *GLM-4*, furthers 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 where each LLM generates text segments that may not align to

Dataset	LLaMA-3	Phi-3	$Cool_2$	GLM-4	Cool
English	0.72	0.696	**0.724**	0.652	0.716
Chinese	0.504	0.472	0.504	0.58	**0.588**
Spanish	0.572	0.456	0.548	0.564	**0.596**
French	0.568	0.576	0.62	**0.668**	0.636
German	0.592	0.568	0.568	0.66	**0.696**
Russian	0.54	0.428	0.548	0.564	**0.592**
Average	0.5827	0.5327	0.5853	0.6147	**0.6373**

Table 5: Accuracies on the *multilingual GSM* datasets.

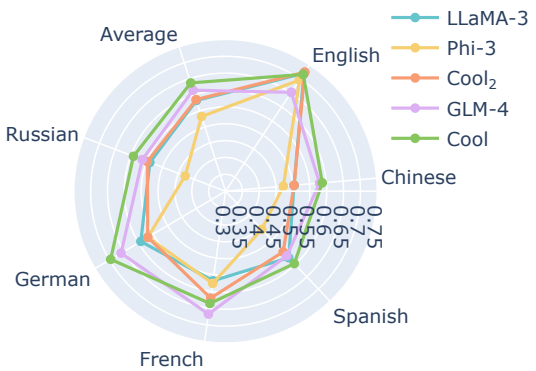

Figure 4: Accuracies on the *multilingual GSM* datasets.

the word-level boundaries of other LMMs. Our full implementation $Cool$ have a 0.3% relative improvement, since $Cool_{-align}$ suffers from occasional biased perplexity, as discussed in Section 2.1 .

We did not compare with existing LLM fusion methods that require training, but we propose a simple reranking method, named *Rerank*, as a baseline, which simply use the average perplexity of the source LLMs, *LLaMA-3*, *Phi-3*, and *GLM-4*, to rerank the complete and individual generated texts from these LLMs. Comparatively, *Cool* is a fine-grained step-by-step reranking approach. *Rerank* turns out to be very effective and obtain a 12.5% increment over *LLaMA-3*. Being orthogonal approaches, when combining *Cool* with *Rerank*, we obtain a huge accuracy improvement of 17.4% over *LLaMA-3*, and 4.4% over *Rerank*. This shows that the generations of *Cool* are very diverse from those of its source LLMs.

Next, we examine the general performance of *Cool* in three different domains, where not all of our source LLMs have good performance. On the *Q&A* datasets (Table 6 and Figure 5) *LLaMA-3* performs best but *GLM-4* fails to follow the output format in the 3-shot prompts. On the other hand, on the *multilingual GSM* datasets (Table 5 and Figure 4), the overall performance of *GLM-4* is the best, while *Phi-3* is not very good at multilingual data (et al, 2024). Finally, on the *Math* dataset (Table 4 and Figure 3), *Phi-3* is the best performer, and the other two LLMs lag behind with a significant gaps. It is a challenging to fuse LLMs in these datasets where the performance of the source LLMs differ and fluctuate dramatically. *Cool* either outperforms all source LLMs or being comparable to the best performer and not being affected by the worser ones, which show that *Cool* is a very safe fusion of source LLMs across different domains.

Compare the performance of *Cool* in Table 3 with those Table 4, we found that *Cool* performs much better in *GSM8K* than in *multilingual GSM*, although the latter is a translated subset of the former. This is probably because *multilingual GSM* does not preserve the difficult distribution in *GSM8K*. We believe that our performance in *GSM8K* is more representative, since it is around five times larger than *multilingual GSM*.

3.4 Summary of Experiments

In this section, we verified the effectiveness of the components in *Cool-Fusion*, which shows significant improvement on very challenging tasks, achieves further advances when combined with other approaches, and persistently being better or comparable to the best performer even when some of the source LLMs had deteriorated performance. We did not perform unfair comparison between *Cool-Fusion* and existing LLM fuse methods that require training, which will be discussed in the next section.

4 Related Work

In this section, we summarize prior work on model and LLM fusion. To our best knowledge, all prior approaches for the fusion of heterogeneous LLMs involve different types of training or fine-tuning.

Reranking. Reranking methods first generate multiple candidates from via probabilistic sampling, or using different prompts or LLMs. Then, different scoring methods (Ravaut et al., 2022; Jiang et al., 2023) are used to assess the quality of these candidates.

Alignment. Alignment matches the units of multiple models, i.e. the vocabularies of different LLMs. However, finding the optimal alignment is a combinatorial optimization problem, and heuristics with specific metrics are used for reducing the vocabulary differences between models. Currently, alignment between vocabulary is still an open problem. It is unclear if the alignment ap-

Dataset	LLaMA-3	Phi-3	Cool
F1			
CoQA	0.8172	0.8091	**0.8225**
Drop	0.5277	0.4041	**0.5504**
Average	0.6724	0.6066	**0.6865**
EM			
CoQA	0.6727	**0.6848**	0.6795
Drop	0.4268	0.2925	**0.4539**
TriviaQA	**0.6121**	0.4829	0.5927
Average	0.5705	0.4867	**0.5754**

Table 6: F1 & EM on the four Q&A datasets.

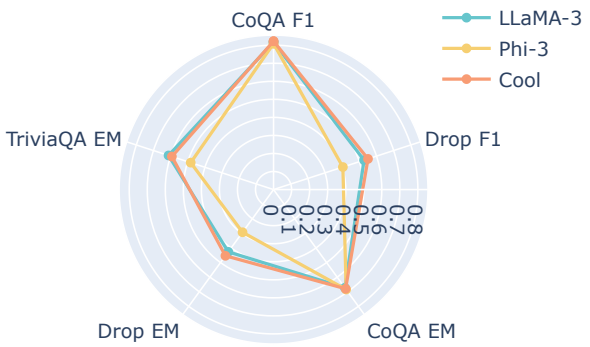

Figure 5: F1 & EM on the four Q&A datasets.

proaches (Mavromatis et al., 2024; Xu et al., 2024) that rely on lexical similarity will work for unicode vocabularies such as Chinese, where tokenizers might not share the same set of basic symbols. For instance, unicode bytes are the basic symbols in *Phi-3* (et al, 2024), while sub-word tokenization for Chinese (Si et al., 2023) uses glyph or pronunciation encoding. *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 of a source LLM to that of a target LLM. *EVA* (Xu et al., 2024) trains a vocabulary projection matrix to map from each non-pivot model to the pivot model before ensembling their logits.

Ensembling. Conventional ensembling approaches 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, conditions on which, another fine-tuned *Flan-T5-XL* 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, which is similar to our approach.

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 hyper-parameter configurations, often improves accuracy and robustness. *PAPA* (Jolicoeur-Martineau et al., 2024) proposes to obtain 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 method gain improved performance without accessing the training data, however, they require that the models to fuse share the same architecture.

Knowledge distillation. Training LLMs from scratch comes at significant costs, *Alpaca* (Taori et al., 2023) used *text-davinci-003* to generate the instruction data to distill a 7B *LLaMA* (Touvron et al., 2023) model. Recent work (Wan et al., 2024a,b) applies distillation as a cost-effective approach to merge existing pre-trained LLMs into a more potent model.

Multi-agent. 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. This breaks down complex tasks into subtasks, allowing agents with different domain expertise–such as architecture design and code debugging–to work harmoniously.

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. *Composition to Augment Language Models* (*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 proposed *Cool-Fusion*, a simple yet effective approach that fuses the knowledge of heterogeneous source LLMs. Extensive experiments with challenging datasets and strong source LLMs verified the persistent improvements and the robustness of our proposal. Future directions include: efficient implementation, fusion of larger number of LLMs, fusion of larger LLMs, and exploration of other reranking metrics other than average perplexity.

6 Limitations

Efficiency. The complexity of generating a sequence with k source LLMs is $O(k^2)$ times of a single source LLM, thus resulting in a much less efficient solution. We showed that those inferences can be executed in parallel because they are completely independent. Our released code is intended for prototyping our approach and has not been optimized for efficiency. We pointed out optimization directions for faster decoding (Section 2.4).

Human evaluation. We used automatic metrics come with Evaluation Harness (Gao et al., 2023). Human or GPT-4 evaluation could provide us with more reliable and comprehensive results. However, due to the number of models and datasets in our experiments, we cannot afford large-scale human evaluation.

References

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*.

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.

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.

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.

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*.