URG: A Unified Ranking and Generation Method for Ensembling Language Models

Bo Lv[1,2,3*], **Chen Tang**[4†], **Yanan Zhang**[5,6*], **Xin Liu**[2], **Ping Luo**[1,2,3], **Yue Yu**[2]
[1]Key Lab of Intelligent Information Processing of Chinese Academy of Sciences, Institute of Computing Technology
[2]Peng Cheng Laboratory, [3]University of Chinese Academy of Sciences
[4]Department of Computer Science, The University of Manchester, UK
[5]College of Computer Science, Sichuan University
[6]Engineering Research Center of Machine Learning and Industry Intelligence, Ministry of Education, China
lvbo19@mails.ucas.ac.cn, chen.tang@manchester.ac.uk

Abstract

Prior research endeavors of the ensemble Large Language Models (LLMs) achieved great success by employing an individual language model (LM) rank before the text generation. However, the use of an individual LM ranker faces two primary challenges: (1) The time-intensive nature of the ranking process, stemming from the comparisons between models; (2) The issue of error propagation arising from the separate ranking and generation models within the framework. In order to overcome these challenges, we propose a novel ensemble framework, namely Unified Ranking and Generation (URG). URG represents an end-to-end framework that jointly ranks the outputs of LLMs and generates fine-grained fusion results, via utilizing a dedicated cross-attention-based module and noise mitigation training against irrelevant information stemming from bad ranking results. Through extensive experimentation and evaluation, we demonstrate the efficiency and effectiveness of our framework in both the ranking and generation tasks. With the close coordination of the ranking and generation modules, our end-to-end framework achieves the state-of-the-art (SOTA) performance on these tasks, and exhibits substantial enhancements to any of the ensembled models.

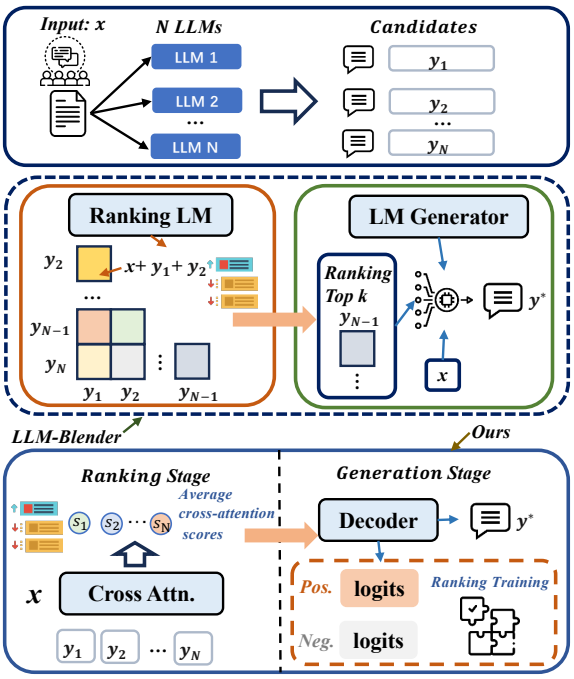

Figure 1: The illustration depicts the proposed framework, URG, alongside a representative prior work in the ensemble LLM models domain, LLM-Blender."Pos." represents positive, and "Neg." represents negative. $y*$ symbolizes the generated text output of the generator. "Cross Attn." denotes the cross attention networks, which share parameters with the decoder cross-attention modules.

1 Introduction

Large language models (LLMs) have shown a superior performance in storing factual knowledge in their parameters (Chowdhery et al., 2022; Lv et al., 2023; Tang et al., 2023b, 2024; Zhao et al., 2024). These off-the-shelf LLMs exhibit diverse strengths and weaknesses due to variations in data, architectures, and hyperparameters, making them complementary to each other. Therefore, by combining their unique contributions, the ensemble of LLMs can alleviate biases, and errors in individual models, resulting in outputs better aligned with human preferences.

In recent years, the Ensembling of Large Language Models (LLMs) has demonstrated remarkable performance through the judicious incorporation of the diverse strengths of multiple LLMs. A simple approach (Yu et al., 2023a) is to concatenate all candidate results as contextual inputs to a new

*Equal contribution.
†Corresponding author.

generative model for fusion output, but meanwhile this approach also introduces a too lengthy input for the ranker, Typically, the prior ensemble approaches (Salazar et al., 2020; Liu and Liu, 2021; Ravaut et al., 2022; Jiang et al., 2023) employed an individual ranking model ranking response outcomes from multiple LLMs, and then select the top-ranked candidate output as the final ensemble result feeding to a generator. The main difference in the prior works relies on the design of the mechanism in the ranking model. We select the state-of-the-art model, LLM-Blender (Jiang et al., 2023), as an example. The authors propose a coarse-grained and fine-grained mixed pipeline that firstly makes a pair-wise comparison feature matrix of N LLMs evaluated by a language model as the scorer, and then a generative model fuses to generate the final output from the top-k ranked outputs.

However, training individual ranking models and generators faces two primary challenges. (1) The time-intensive nature of the ranking process: Due to the input limitations of language models, pairwise comparisons are implemented when comparing multiple models. Consequently, when dealing with N LLMs, the time complexity of the process becomes $O(N^2)$. In addition, the individual ranking model also incurs additional memory and time costs. (2) The error propagation between ranking and generation: The ranking stage and the generation stage represent two distinct tasks with different inputs and outputs. The textual evaluation knowledge acquired by the ranking model cannot be transferred to the subsequent generator, and vice versa. The text generation objective of the generator is not directly connected to the ranking model. Consequently, the framework may suffer from the introduction of poor candidates (noise) during the ranking stage, which may adversely affect the subsequent generation process.

To address the aforementioned issues, we propose a novel unified ranking and generation method, abbreviated as URG. This method realizes an end-to-end framework with an encoder-decoder-based language model, encompassing both the ranking and generation modules. As depicted in Figure 1, our framework initially gathers the encoded embeddings of candidate responses from N LLMs, along with the encodings of the input x. Subsequently, a cross-attention module is implemented to list-wise form the concatenated features composed by the pair of each N candidate and x, which has merely $O(n)$ time complexity. This cross-attention module assigns the attention scores to each LLM candidate pair, and aggregates the feature list to predict the tokens in $y*$. Therefore, we share the parameters of this cross-attention module with that of the decoder in the generator, which also aims to predict the next token of $y*$ based on x with the top k candidates and the history tokens predicted in $y*$. Consequently, the decoder learns the representations of N candidates throughout both the ranking and generation stages. In the subsequent generation stage, we further introduce a Kullback-Leibler (KL) loss to contrastively learn the features of positive and negative LLM candidates. By minimizing the KL loss, the decoder (generator) mitigates the adverse effects on the final output $y*$ from low-quality candidates (negative). This training objective serves to mitigate the influence of noise introduced during ranking, bridging the gap between the ranking stage and the subsequent generation stage, resulting in a more robust ensemble model. We conduct a series of experiments to analyze the effectiveness and efficicenty of our proposed framework. The experimental results demonstrate that our framework achieves state-of-the-art (SOTA) performance in both the ranking and generation stages. Furthermore, our framework exhibits a substantial improvement in running speed compared to the prior SOTA model, LLM-Blender.

Our contributions can be summairzed as follows:

- We propose a novel framework, Unified Ranking and Generation Method (URG), which integrates the ranking and generation stages within an end-to-end framework.
- We introduce a KL-loss based mechanism to contrastively learn the features of positive and negative LLM candidates, thereby enhancing the robustness of the ensemble framework against the adverse impact of low-quality LLM candidates introduced during the ranking stage.
- We conduct a series of experiments demonstrating that URG significantly reduces running time and achieves state-of-the-art (SOTA) performance according to various automatic metrics and human evaluations.

2 Related Work

Ensemble learning is a widely employed technique that enhances accuracy and resilience in forecasting by merging predictions from multiple models (Sagi and Rokach, 2018; Aniol and Pietron, 2019; Wei

et al., 2023). With the rapid increase in the application of large language models (LLMs) (Liu et al., 2020), the ensemble of LLMs has emerged as a significant topic (Yu et al., 2023b). In light of the prior works. There are two primary approaches for ensembling LLMs: selection-based and generation-based methods.

Selection-Based Methods Selection-based methods develop ranking models (Salazar et al., 2020; Liu and Liu, 2021; Ravaut et al., 2022) to compare candidate results generated by the multiple LLMs, selecting the top-1 candidate as the ensemble output. However, these methods are coarse-grained ensembles due to the inherent nature of selection and the limited solution space. As a result, they may limit the ability to harness the advantages of each candidate, potentially hindering the generation of superior outputs.

Generation-Based Methods Generation-based methods employ generative models (Yu et al., 2023a) to fuse the candidate results and generate improved output as the final response. However, these methods face challenges in dealing with the surge in computational complexity and inference time with the increase of input text length (Liu et al., 2022; Sun et al., 2024; Tang et al., 2022; Loakman et al., 2023; Goldsack et al., 2023; Tang et al., 2023a; Yang et al., 2024). To address this issue, Jiang et al. (2023) proposes a rank-then-generate pipeline that first adopts a pairwise ranking model to sort outputs from N LLMs and then utilizes a generative model to generate the final output from the K top-ranked outputs. Despite shortening the length of the input text, this model still faces the problems of slow sorting speed caused by pairwise comparisons and the issue of error propagation arising from separate ranking and generation models.

3 Method

Figure 2 illustrate the overall architecture of our proposed URG framwork, which is an end-to-end model.

3.1 Problem Definition

The base task is instruction-following, and we formulate the task as follows: the input text is x, and the output of this task is to generate a response y. The ground-truths in the data are denoted as $\widehat{y}$. In this paper, we focus on post-hoc ensemble learning, wherein given an input x and N LLMs, $\{M_1, ..., M_N\}$, the task is to combine the candidate results, $Y = \{y_1, ..., y_N\}$, generated by processing x with N LLMs to produce the integrated final output result y^*. Our goal is to develop a novel unified ranking and generation framework that applies a single language model throughout both the ranking and generation stages, to fine-grained fuse the candidate results outputted by the N LLMs.

3.2 Encoding Text

We separately concatenate the input x with each candidate result y_i, forming a unified input sequence with special tokens as separators: <s> <*Question*> x </s> <*response*> y_i, which is denoted as p_i. Subsequently, each input sequence p_i is fed into encoder layers to obtain the encoder's last layer hidden state H_e^i:

$$H_e^i = \text{Encoder}_{\frac{L_{urg}}{2}}(p_i) \in \mathbb{R}^{T_p \times h} \quad (1)$$

where h is the hidden dimension, L_{urg} is the total number of encoder layers and decoder layers, and T_p is the sequence length of p_i. By processing candidates independently in the encoder, the computational complexity of the model grows linearly with the number of candidates, as opposed to quadratic growth when concatenating all candidates.

3.3 Ranking and Filtering Candidates

The objective of the ranking stage is to assess the qualities and rank the generated N candidates from LLMs. Only the top k candidates are retained for the subsequent generation stage, mitigating potential issues associated with increased time complexity and noise resulting from inputting an excessive number of candidates into the subsequent generator. The remaining N candidates are regarded as relatively lower-quality instances answering x, particularly the one ranked last, which is also utilized in the following steps for contrastive learning.

Why Choose Cross-Attention The standard cross-attention computation for a single head in a transformer is:

$$\text{Attn}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \quad (2)$$

where $Q = H_d W_q$ is the product of the decoder states H_d and the query weight matrix W_q; the keys $K = H_e W_k$ are the product of the last encoder hidden states H_e with the key weight matrix W_k; and $V = H_e W_v$ is similarly the product of H_e with the value weight matrix W_v.

We denote the attention score between the query Q_i at position i, and the key K_j at position j as

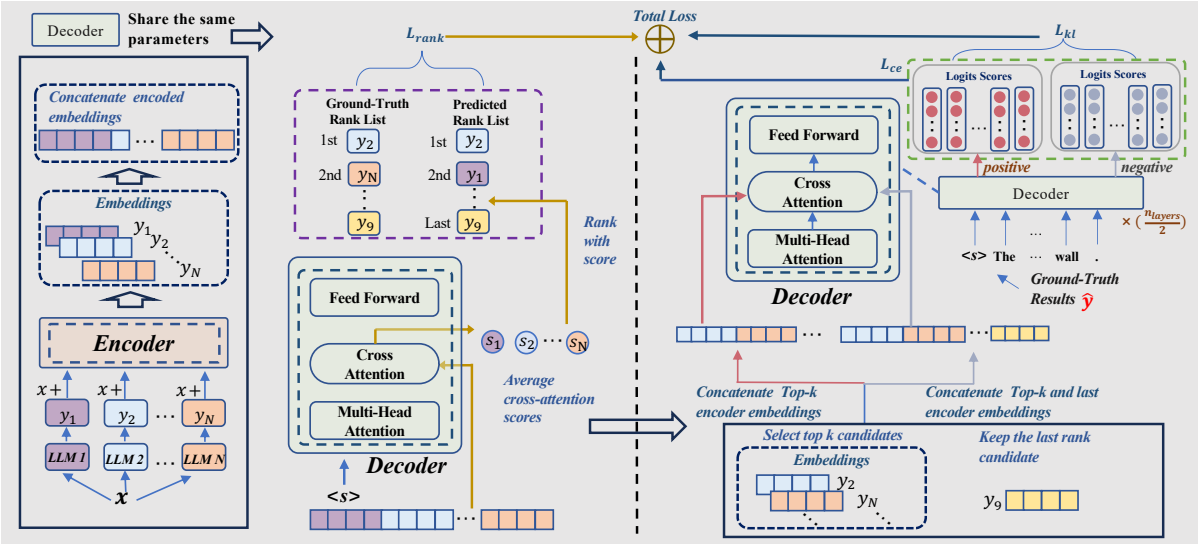

Figure 2: The architecture of our method. On the left side, textual data in text format are converted into vector-format embeddings, so the symbols on the right represent the corresponding embeddings.

$\alpha_{i,j} = Q_i K_j^T / \sqrt{d_k}$. After normalized over the dimension, the attention score is rewritten as:

$$\widetilde{\alpha}_{i,j} = \text{softmax}(\alpha_{i,j}) = \frac{\exp(\alpha_{i,j})}{\exp(\sum_m (\alpha_{i,m}))} \quad (3)$$

where $\frac{1}{\sqrt{d_k}}$ is the scaling factor and the d_k is the dimension of each transformer head. According to Equation 2 and Equation 3, the new representation can be expressed as a sum of the values, weighted by the attention probabilities, before going through a final linear transformation W_o:

$$O_i = W_o \sum_j \widetilde{\alpha}_{i,j} V_j \quad (4)$$

It is evident that the attention score $\widetilde{\alpha}_{i,j}$ quantifies the significance of V_j in the computation of the new representation O_i. A lower attention score associated with V_j suggests its diminished importance for O_i. Consequently, by utilizing the aforementioned steps to compute attention scores , the ranking module is trained to discern the qualities of N candidates.

How to Utilize Cross-Attention In our paper, the query Q of the cross-attention mechanism is the prediction y^*. K and V are established based on the Ncandidates. Initially, the hidden state of all candidate results output from the last layer of the encoder are concatenated and forwarded to the decoder to generate the first token A:

$$A = Decoder([H_e^1; H_e^2; \ldots; H_e^N]) \quad (5)$$

where A represents the initial representations of y^*. In this process, we can obtain attention scores $\widetilde{\alpha}_{1,j}^l$ at each decoder layer l, and then aggregate these scores across all layers and heads to derive $\widetilde{\alpha}_{1,j}^{mean}$. Subsequently, we can compute the ranking score s_k for each candidate result y_k by averaging all the attention scores $\widetilde{\alpha}_{1,j}^{mean}$ corresponding to the H_e^k:

$$s_k = \frac{\sum_{j=b}^{b+T_p} \widetilde{\alpha}_{1,j}^{mean}}{T_p} \quad (6)$$

where b is the start token position of H_e^k within the concatenated sequence, and T_p is the length of H_e^k. Consequently, the predicted rank list of candidate results Y is obtained.

The following loss function is used to optimize this rank result:

$$\mathcal{L}_{rank} = \sum_{i=1}^{N} \sum_{j=1}^{N} max(0, s_i - s_j)_{r_i < r_j} \quad (7)$$

where r_i,r_j are the ground-truth scores of y_i,y_j. The ground-truth score r can be obtained from either the reward model output or human ratings. In this paper, the ranking labels are provided by the MixInstruct dataset, which uses BART scores to rank the candidates. These scores are generated by a BART model that compares candidate outputs with answer responses.

3.4 Enhancing The Model Robustness

Although in an ideal scenario, the top k candidates retained are presumed to exhibit good qualities, practical situations may still entail the presence of low-quality candidates after the ranking stage.

Such low-quality candidates have the potential to exert adverse effects on the final output y^* generated by the generator. In order to enhance the model's resilience against irrelevant information stemming from low-quality candidates within the input, we introduce an additional noise mitigation training approach. Here, the worst candidate (ranked last) is incorporated alongside the top k candidates as part of the input, forcing the model to generate outputs that are consistent with those derived from the top-k candidates.

The encoded embeddings of the top-k candidate results, denoted as $[H_e^{1st}; \ldots; H_e^{k-th}]$, are concatenated to form E_{tk}. Furthermore, the encoded embeddings of the lowest-ranked result are appended to E_{tk}, yielding a new concatenated embedding $E_{tk-l} = [H_e^{1st}; \ldots; H_e^{k-th}; H_e^{last}]$. Given the ground-truth $\widehat{y}$ consisting of L tokens, represented as $T = t_1, t_2, ..., t_L$, the model generates the output logits using E_{tk}^T as the condition. Subsequently, the probabilities of the top-m candidate tokens at each decoder step are normalized using the temperature parameter τ to obtain the probability distribution p:

$$p(t_j^* = V_n | t_{<j}, E_{tk}) = \frac{exp(z(V_n)/\tau)}{\sum_m exp(z(V_i)/\tau)} \quad (8)$$

where the $z(V_n)$ and $z(V_i)$ are the probabilities for the n-th and i-th candidate tokens in the vocabulary, respectively. Inspired by knowledge distillation in neural machine translation tasks, we use top-m candidate tokens at each decoding step, instead of the full distribution, in order to focus on important tokens when calculating the distribution divergence. Inspired by knowledge distillation (Tan et al., 2019; Wei et al., 2021; Lv et al., 2024) in neural machine translation tasks, we use the top-m candidate tokens at each decoding step, instead of the full token distribution. This enables the model to focus on capturing important token information, enhancing its learning capacity. Similarly to the steps producing p, E_{tk-l} is fed into the model to obtain the normalized probability, $q(t_j^* = V_n | t_{<j}, E_{tk-l})$, of the top-$m$ candidate tokens in the new output probability distribution. Finally, we employ the KL (Kullback-Leibler) loss to minimize the divergence between the aforementioned distributions:

$$L_{kl} = \sum_j^L \sum_{n=1}^m p(t_j^* = V_n | t_{<j}, E_{tk}) \times \log \frac{p(t_j^* = V_n | t_{<j}, E_{tk})}{q(t_j^* = V_n | t_{<j}, E_{tk-l})} \quad (9)$$

Additionally, we employ cross-entropy loss to minimize the disparity between the model's predicted probabilities and the ground-truth $\widehat{y}$:

$$L_{ce} = -(\sum_{j=1}^L \log p(t_j | T_{<j}, E_{tk}) + \sum_{j=1}^L \log p(t_j | T_{<j}, E_{tk-l})) \quad (10)$$

The overall loss function of our approach comprises the rank loss, the KL divergence loss, and the cross-entropy loss:

$$L_{total} = \gamma L_{rank} + \beta L_{kl} + L_{ce} \quad (11)$$

where γ and β represent the weights assigned to the rank loss and KL loss, respectively.

3.5 Analyzing Computational Complexity

We analyze the theoretical time complexity of our proposed method compared to the vanilla method. The more practical computational cost comparison is shown in Table 4.

Suppose the length of the ensembled result is denoted as T_a, and the average length of the candidate result (concatenated with question) is T_p. The time complexity of the pairwise ranking model (Jiang et al., 2023) with L layers is $O(N^2 \cdot T_p^2 \cdot d \cdot L)$, where N, d denote the number of candidate results and the embedding dimensions. For list-wise models (Liu and Liu, 2021), the time complexity is $O(N \cdot T_p^2 \cdot d \cdot L)$, as each candidate's score is evaluated only once instead of through pairwise comparisons.

The ranking stage of our URG method firstly utilizes the encoder to encode all the candidate results separately, with a time complexity of $O(N \cdot T_p^2 \cdot d \cdot \frac{L_{urg}}{2})$, where L_{urg} is the total number of layers in URG. Both the encoder and decoder have $\frac{L_{urg}}{2}$ layers. Then, we concatenate all the embeddings and input them into the decoder to perform a decoder step. The time complexity of this part is $O(\frac{L_{urg}}{2} \cdot d \cdot (N \cdot T_p \cdot T_a + T_a^2))$, where $N \cdot T_p \cdot T_a$ comes from the cross-attention mechanism, and T_a is the number of decoder step. Since we only executed a single decoder step, $T_a = 1$, the computation of the decoding phase can be considered negligible compared to the encoding phase. As a result, the overall complexity of the URG ranking stage can be approximated as $O(N \cdot T_p^2 \cdot d \cdot \frac{L_{urg}}{2})$.

Instruction Sources	#Examples	Output Sources
Alpaca-GPT4	22,862	GPT-4
Dolly-15K	7,584	Human
GPT4All-LAION	76,552	ChatGPT
ShareGPT	3,002	ChatGPT
Total	110K	Mix

Table 1: Data statistics of MixInstruct. The output sources refer the sources that provide the responses to the executed instruction.

4 Experiments

4.1 Datasets and Evaluation

MixInstruct MixInstruct was curated by (Jiang et al., 2023) to serve as a benchmark for ensemble models in instruction-following tasks utilizing Large Language Models (LLMs). Table 1 illustrates that this benchmark encompasses a substantial collection of instructional examples sourced primarily from four distinct origins. Specifically, 100k examples are randomly allocated for training, with 5k each designated for validation and testing purposes. To generate candidate results, each instruction is fed into a set of $N = 12$ popular open-source language models (LLMs), such as Vicuna[1], Open Assistant[2], Llama[3], and others (refer to Table 2).

Evaluation To evaluate the efficacy of our framework, a range of automatic metrics are selected for the following experiments. BLEU (Post, 2018) and ROUGE (R-n) (Lin, 2004) evaluate the quality of a generated response in comparison to a reference by measuring n-gram overlap.[4] We additionally adopt some neural network based metrics as a supplement, i.e. BERTScore (Zhang et al., 2019), BARTScore (Yuan et al., 2021) and BLEURT (Sellam et al., 2020) to measure the semantic similarity between the generated results and the references. These metrics leverage pre-trained language models to predict if the model response is semantically equivalent to the gold answer.

Model Sizes We use two configurations for our URG method: (i) Flan-T5-large configuration consisting of 24 layers, 16 attention heads, and 1024 embedding dimensions, leading to 770M parameters, and (ii) a larger Flan-T5-xl configuration consisting of 3B parameters.

Compute Hardware We perform training on instances containing 8 A100 GPUs, each containing 80 GB RAM.

Training Details When training with Flan-T5-xl (3B), we perform training for 10 epochs using Adam with a batch size of 4, dropout value of 0.1, and peak learning rate of 8×10^{-5} with warmup and linear scheduling. Due to the smaller size of Flan-T5-large, we train for 20 epochs with a batch size of 16. We save the model checkpoint every 1000 steps and perform model selection by evaluating it on the development set. To save memory, we use stage 2 and bfloat16 mixed precision in the deepspeed[5] training framework to train the model. In the training phase, we select the top k ranked candidates, where k is set to 3 for the generation phase. When calculating the KL loss, we choose the top 32 scores for the computation, implying that m is set to 32. We set the value of the temperature hyperparameter ($\tau = 2$) and weight coefficient ($\gamma = 5, \beta = 2$) using cross-validation.

4.2 Baselines

In this section, we introduce several baseline methods for ensembling LLMs.

- **SummaReranker** (Ravaut et al., 2022): A selection-based method concatenates the input and each candidate to learn ranking and utilizes binary cross-entropy (BCE) loss to differentiate the best candidate from the others.
- **SimCLS** (Liu and Liu, 2021): A popular contrastive learning-based method encodes the input and each candidate, then computes their cosine similarity as the ranking score.
- **PAIRRANKER** (Jiang et al., 2023): A pairwise comparison method focuses on learning to capture the differences between two candidates and prefers the ones of higher quality.
- **LLM-Blender** (Jiang et al., 2023): A generation-based ensembling framework consisting of a PairRanker and a generation model.
- **SimCLS-Gen**: A generation-based ensembling method first takes the top-3 ranked results from SimCLS and then inputs them into a generative model to generate a new fused result.
- **SummaReranker-Gen**: A method similar to SimCLS-Gen, wherein the rank model is SummaReranker.

The backbone of the ranking models in the base-

[1] https://huggingface.co/lmsys/vicuna-13b-v1.5

[2] https://huggingface.co/OpenAssistant/oasst-sft-4-pythia-12b-epoch-3.5

[3] https://huggingface.co/mosesjun0h/llama-7b-hf-baize-lora-bf16

[4] R-L refers to the longest common subsequence.

[5] https://github.com/microsoft/DeepSpeed

line is DeBERTa-large (418M) (He et al., 2020), and the generation models are all Flan-T5-XL (3B) (Chung et al., 2022b). We train the aforementioned baselines using the code and parameters provided by Jiang et al. (2023). All selection-based methods involve sorting all candidates and outputting the top-1 candidate based on the scores as the ensemble output.

5 Experiments Results

5.1 Main Results

Table 2 shows the performance of N=12 open-source LLMs as well as other baseline models on the MixInstruct test set. The results of both the selection-based ensemble method and the generation-based method are obtained by integrating the outputs of these 12 open-source LLMs. Our proposed URG method demonstrates substantial performance improvements over the previous state-of-the-art (SOTA) model, LLM-Blender, across all evaluation metrics. Notably, even our smaller-sized URG (770M) model exhibits superior performance compared to the LLM-Blender approach, comprising a ranking model with 418M parameters and a generation model with 3B parameters, across most metrics. When compared to ChatGPT, which boasts 175 billion parameters, our URG (3B) model still achieves superior scores on several metrics. These results underscore the effectiveness and efficiency of our method through fine-grained ensembling. Additionally, we compare our ranking results with selection-based methods. Both the smaller and full versions of the URG-rank model outperform the previous SOTA method, PairRanker, across all metrics. This highlights the superiority of our method, which leverages cross-attention scores to simultaneously compare the superiority of all candidates, over previous pair-rank methods that compare candidates pairwise. Moreover, our method exhibits faster inference speeds, as we will analyze in §5.3.

5.2 Ablation Study

Since our proposed URG method works with two novel techniques introduced in ranking mechanism (§3.3) and enhancing robustness mechanism (ERM) (§3.4), we conduct an ablation study to validate the effectiveness of both. As Table 3 illustrates, the performance of URG substantially drops when either of the two mechanisms is removed, demonstrating the effectiveness of both mechanisms.

5.3 Analysis of Inference Speed

We conducted experiments to compare the inference speed of the ranking stage in the URG method with previous ranking models. The results of the experiments are presented in Table 4. Considering both the model size and speed, Our URG method models achieve the fastest inference speed. Furthermore, even though our URG (3B) model has a large number of parameters, it still has advantages over the previous SOTA method, PairRanker, in terms of both performance and inference speed. Aligning with the theoretical analysis in §3.5, even though the URG models have much larger model size (around 1.7 times and 6.7 times than the baselines), the URG models have less computational cost due to the optimized encoding and decoding strategy. Table 7 shows a comparison of the performance and inference speed of our model and the previous model under different batch sizes. We also calculated the corresponding inference speed $\frac{\text{size(M)}\times\text{speed}}{\text{1B}}$ after normalizing the parameters to 1B, in order to explore the comparison between our method and other methods under the same parameter size. The results show that our method is faster than the previous method under different batch sizes, while also achieving higher BERTscore and Bartscore. This indicates that, compared to the state-of-the-art (SOTA) method PairRank, which previously compared candidates pairwise, our method achieves better performance by comparing all candidates together with a single decoder step, and it also has a faster inference speed.

5.4 Analysis of Model Robustness

To compare the ability of our method with other methods in resisting noise introduced in the ranking stage, we conduct a model robustness testing experiment. In this experiment, lower-ranked candidates, regarded as noise candidates, results are fed into the model for decoding. As shown in Table 5, the adverse effects of noise candidates have a significantly lesser impact (almost half) on the performance of the URG generator than that of the LLM-Blender generator. This suggests that our mechanism for enhancing model robustness (see §3.4) effectively bolsters the model's resilience to noisy inputs.

5.5 Analysis of The Top k value

To assess the impact of different values of k on the fusion results, we conducted experiments to exam-

Type	Model	bertscore↑	bartscore↑	bleurt↑	rouge1↑	rouge2↑	bleu↑	rougeLsum↑
Open-Source LLMs	**Alpaca-native (Taori et al., 2023)**	0.7146	-3.5696	-0.5307	0.3276	0.1362	7.6478	0.2915
	ChatGLM-6b (Du et al., 2021)	0.7038	-3.5193	-0.6167	0.3063	0.1192	6.0493	0.2734
	Dolly-v2 (Conover et al., 2023)	0.6226	-3.8331	-0.8654	0.1811	0.0495	2.0620	0.1606
	Flan-T5-xxl (Chung et al., 2022b)	0.6492	-4.5717	-1.2288	0.1444	0.0432	1.6066	0.1296
	Koala-7B (Geng et al., 2023)	0.6396	-3.8496	-0.8354	0.2131	0.0662	3.0983	0.1871
	llama-7b (Touvron et al., 2023)	0.6557	-3.5260	-0.6630	0.2253	0.0781	3.4005	0.2022
	MOSS-sft (Sun et al., 2023)	0.6485	-3.6461	-0.7261	0.2062	0.0686	2.9561	0.1863
	MPT (Team et al., 2023)	0.6165	-3.9419	-0.9636	0.1663	0.0439	1.7392	0.1498
	MPT-instruct (Team et al., 2023)	0.6321	-3.7208	-0.8232	0.1837	0.0524	2.0692	0.1647
	Oasst-12b (Chung et al., 2022a)	0.7468	-3.4486	-0.3908	**0.3813**	0.1738	**10.5046**	**0.3410**
	StableLM (Stability-AI, 2023)	0.6247	-4.1208	-0.9832	0.1904	0.0524	2.5044	0.1672
	vicuna-13b (Chiang et al., 2023)	0.6960	-3.4368	-0.6146	0.3012	0.1223	6.3584	0.2677
Closed-Source LLMs	**ChatGPT**[6]	**0.7798**	**-2.8645**	**-0.2783**	0.3776	**0.1739**	10.4586	0.2732
selection-based	**random**	0.6640	-3.7499	-0.7632	0.2452	0.1025	4.3433	0.1740
	SummaReranker	0.7157	-3.3757	-0.3933	0.3893	0.1908	11.5300	0.3543
	SimCLS	0.7257	-3.384	-0.3787	0.3873	0.1771	10.7100	0.3440
	PairRanker	0.7313	-3.2612	-0.3623	0.4054	0.1956	12.1170	0.3650
	URG-rank (770M)	0.7385	-3.2120	-0.3614	0.4099	0.2009	12.1282	0.3661
	URG-rank (3B)	**0.7510**	**-3.2057**	**-0.3485**	**0.4217**	**0.2166**	**13.4494**	**0.3728**
generation-based	**SummaReranker-Gen**	0.7827	-3.1353	-0.2575	0.4442	0.2381	14.7582	0.3568
	SimCLS-Gen	0.7854	-3.1467	-0.2583	0.4471	0.2436	15.3094	0.3628
	LLM-Blender	0.7879	-2.9800	-0.1831	0.4527	0.2477	15.8212	0.3631
	URG (770M)	0.7895	-3.0531	-0.2064	0.4611	0.2489	15.8076	**0.3750**
	URG (3B)	**0.8045**	**-2.8901**	**-0.1755**	**0.4632**	**0.2538**	**16.5900**	0.3727

Table 2: Empirical results on MixInstruct, with the best result for each type highlighted in bold.

Model	bertscore↑	bartscore↑	bleu↑	rougeLsum↑
LLM-BLENDER	0.7879	-2.9800	15.8212	0.3631
URG	**0.8045**	**-2.8901**	**16.5900**	**0.3750**
w/o ranking	0.7868	-3.0433	15.2191	0.3604
w/o ERM	0.7961	-2.9212	16.1860	0.3697

Table 3: Ablation study of our URG method in two mechanisms.

Model	Model Size	berts score↑	bart score↑	speed (samp./s)↑	Norm(size) (samp./s) ↑
SummaReranker	465M	0.7157	-3.3757	3.7037	1.7222
SimCLS	434M	0.7257	-3.3840	6.1538	2.6707
PairRanker	436M	0.7313	-3.2612	1.5385	0.6707
URG	770M	0.7385	-3.2120	**8.8889**	6.8445
URG	3B	**0.7510**	**-3.2057**	3.9216	**11.7648**

Table 4: Inference speed (samples/s) of different models on the MixInstruct benchmark test data, with batch sizes set to 8. Norm(size) represents the corresponding inference speed when the model size is normalized to 1B. Each sample contains N=12 candidate results from LLMs.

ine the variation of BERTScore as the number of candidate inputs k increases during the fusion stage. The BERTScore for each k value was obtained by training the model using that specific k value and selecting the best score on the development set.

As depicted in Figure 3, the BERTScores of the three models initially exhibit an upward trend with the increase of the k value. However, beyond $k = 3$, the growth rate notably decreases, and in some cases, the scores cease to increase altogether. Consequently, to optimize model training and inference speed by minimizing input text, we fixed the settings of $k = 3$ for both training and testing in our model.

Model	Cands Ranking	bertscore △ %	bartscore △ %	rougeLsum △ %	bleu △ %
URG	**12**	-1.0956%	-0.2236%	-1.1430%	-4.6947%
	11,12	-4.4115%	-1.0557%	-3.8338%	-10.2857%
LLM-Blender (Generator)	**12**	-2.9201%	-0.4873%	-2.0388%	-9.6560%
	11,12	-7.3104%	-1.6994%	-7.3999%	-22.2686%

Table 5: Results of our method's robustness analysis. The scores indicate the percentage decrease in metrics when additional candidates with lower ranks are included in the input, compared to input only the top 2 ranked candidates into the model. 'Cands Ranking' refers to the ground-truth ranking of the additional candidates.

5.6 Human Evaluation

Three evaluators are invited to annotate 200 samples, which are randomly sampled on the test set of the MixInstruct dataset. Table 6 shows that compared with the LLM-Blender method, our URG method has a substantial increase in performance on both the appropriateness and informativeness metrics. This is in line with the evaluation conducted using machine learning and automated metrics, which consistently demonstrates that our method outperforms the previous SOTA methods,

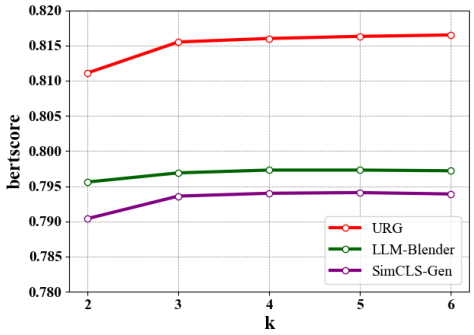

Figure 3: The trend of bertscore on the development set as the number of selected top k candidates increases.

regardless of the evaluation method employed. In addition, by integrating the outputs of 12 open-source large models, the results of our model's output demonstrate competitive performance comparable to ChatGPT. This indicates that the integration method can partially compensate for the performance gap caused by the size of individual models.

Choices (%)	LLM-Blender vs. URG		
	LLM-Blender	URG	Kappa
Appropriateness	41.00	**59.00***	0.537
Informativeness	33.50	**66.50****	0.600
Overall Quality	27.00	**73.00****	0.432
Choices (%)	**ChatGPT vs. *URG***		
	ChatGPT	URG	Kappa
Appropriateness	42.00	**58.00**	0.480
Informativeness	**68.50****	31.50	0.641
Overall Quality	47.00	**53.00**	0.570

Table 6: Pairwise comparisons of human preference percentages. Scores are marked on a scale of 1-5. *Kappa* denotes Fleiss' Kappa (Fleiss, 1971), indicating the agreement level across evaluators. All human annotations show Moderate or Substantial agreement. We also conduct a sign test for performance comparison. $*$ refers to significance at p<.05, whilst $**$ refers to significance at p<.01.

5.7 Case Study

Table 8 illustrates an example showcasing the varied responses from our URG model, open-source LLMs, LLM-Blender, ChatGPT, and the reference human response, indicating that our responses are most similar to those of humans. More details are introduced in Appendix A.1.

6 Conclusion

In conclusion, we propose a novel framework, Unified Ranking and Generation Method (URG), which integrates the ranking and generation stages within an end-to-end framework. To enhance the robustness of the URG framework against the adverse impact of low-quality LLM candidates, we introduce a KL-loss based mechanism to contrastively learn the features of positive and negative LLM candidates. Experiments on MixInstruct show that URG significantly reduces running time and achieves SOTA performance according to a variety of automatic metrics and human evaluations.

7 Limitations

The requirement for multiple high-performing models and a significant amount of accurately labeled data to train an ensemble model poses challenges in constructing datasets for other tasks. Despite the good performance of our approach in ensembling the outputs of LLMs in the instruction-following task, we haven't explored whether our method is also effective in different tasks due to the difficulty of constructing other task datasets. Furthermore, considering that the computational complexity of model inference increases quadratically with the input length, even after narrowing down the candidate inputs from N to k through quality filtering, this stage remains sluggish. Therefore, the problem of extracting only a limited number of snippets from each candidate instead of the entire text requires resolution in the future.

8 Acknowledgements

This work is supported by the National Key Research and Development Program of China (Grant No. 2021ZD0112905), the Major Key Project of PCL (Grant No. PCL2023A09), the National Natural Science Foundation of China (Grant No. 62206140, 62076231), the China Postdoctoral Science Foundation (Grant No. 2022M711726), and the Fundamental Research Funds for the Central Universities (No. 2023SCU12094).

References

Anna Aniol and Marcin Pietron. 2019. Ensemble approach for natural language question answering problem. *Cornell University - arXiv,Cornell University - arXiv*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022a. Scaling instruction-finetuned language models.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022b. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

Mike Conover, Matt Hayes, Ankit Mathur, Xiangrui Meng, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, et al. 2023. Free dolly: Introducing the world's first truly open instruction-tuned llm.

Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2021. All NLP tasks are generation tasks: A general pretraining framework. *CoRR*, abs/2103.10360.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Xinyang Geng, Arnav Gudibande, Hao Liu, Eric Wallace, Pieter Abbeel, Sergey Levine, and Dawn Song. 2023. Koala: A dialogue model for academic research. Blog post.

Tomas Goldsack, Zhihao Zhang, Chen Tang, Carolina Scarton, and Chenghua Lin. 2023. Enhancing biomedical lay summarisation with external knowledge graphs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8016–8032, Singapore. Association for Computational Linguistics.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced BERT with disentangled attention. *CoRR*, abs/2006.03654.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023. Llm-blender: Ensembling large language models with pairwise comparison and generative fusion. In *Proceedings of the 61th Annual Meeting of the Association for Computational Linguistics (ACL 2023)*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Nayu Liu, Xian Sun, Hongfeng Yu, Wenkai Zhang, and Guangluan Xu. 2020. Multistage fusion with forget gate for multimodal summarization in open-domain videos. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1834–1845, Online. Association for Computational Linguistics.

Nayu Liu, Kaiwen Wei, Xian Sun, Hongfeng Yu, Fanglong Yao, Li Jin, Guo Zhi, and Guangluan Xu. 2022. Assist non-native viewers: Multimodal cross-lingual summarization for how2 videos. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6959–6969, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yixin Liu and Pengfei Liu. 2021. SimCLS: A simple framework for contrastive learning of abstractive summarization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 1065–1072, Online. Association for Computational Linguistics.

Tyler Loakman, Chen Tang, and Chenghua Lin. 2023. TwistList: Resources and baselines for tongue twister generation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 579–589, Toronto, Canada. Association for Computational Linguistics.

Bo Lv, Xin Liu, Shaojie Dai, Nayu Liu, Fan Yang, Ping Luo, and Yue Yu. 2023. DSP: Discriminative soft prompts for zero-shot entity and relation extraction. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5491–5505, Toronto, Canada. Association for Computational Linguistics.

Bo Lv, Xin Liu, Kaiwen Wei, Ping Luo, and Yue Yu. 2024. TAeKD: Teacher assistant enhanced knowledge distillation for closed-source multilingual neural machine translation. In *Proceedings of the 2024 Joint*

International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024), pages 15530–15541, Torino, Italia. ELRA and ICCL.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

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.

Omer Sagi and Lior Rokach. 2018. Ensemble learning: A survey. *WIREs Data Mining and Knowledge Discovery*.

Julian Salazar, Davis Liang, Toan Q. Nguyen, and Katrin Kirchhoff. 2020. Masked language model scoring. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

Stability-AI. 2023. Stablelm: Stability ai language models.

Jiankai Sun, Chuanyang Zheng, Enze Xie, Zhengying Liu, Ruihang Chu, Jianing Qiu, Jiaqi Xu, Mingyu Ding, Hongyang Li, Mengzhe Geng, Yue Wu, Wenhai Wang, Junsong Chen, Zhangyue Yin, Xiaozhe Ren, Jie Fu, Junxian He, Wu Yuan, Qi Liu, Xihui Liu, Yu Li, Hao Dong, Yu Cheng, Ming Zhang, Pheng Ann Heng, Jifeng Dai, Ping Luo, Jingdong Wang, Ji-Rong Wen, Xipeng Qiu, Yike Guo, Hui Xiong, Qun Liu, and Zhenguo Li. 2024. A survey of reasoning with foundation models.

Tianxiang Sun, Xiaotian Zhang, Zhengfu He, Peng Li, Qinyuan Cheng, Hang Yan, Xiangyang Liu, Yunfan Shao, Qiong Tang, Xingjian Zhao, Ke Chen, Yining Zheng, Zhejian Zhou, Ruixiao Li, Jun Zhan, Yunhua Zhou, Linyang Li, Xiaogui Yang, Lingling Wu, Zhangyue Yin, Xuanjing Huang, and Xipeng Qiu. 2023. Moss: Training conversational language models from synthetic data.

Xu Tan, Yi Ren, Di He, Tao Qin, Zhou Zhao, and Tie-Yan Liu. 2019. Multilingual neural machine translation with knowledge distillation. *CoRR*, abs/1902.10461.

Chen Tang, Chenghua Lin, Henglin Huang, Frank Guerin, and Zhihao Zhang. 2022. EtriCA: Event-triggered context-aware story generation augmented by cross attention. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5504–5518, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Chen Tang, Tyler Loakman, and Chenghua Lin. 2024. A cross-attention augmented model for event-triggered context-aware story generation. *Computer Speech & Language*, page 101662.

Chen Tang, Shun Wang, Tomas Goldsack, and Chenghua Lin. 2023a. Improving biomedical abstractive summarisation with knowledge aggregation from citation papers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 606–618, Singapore. Association for Computational Linguistics.

Chen Tang, Hongbo Zhang, Tyler Loakman, Chenghua Lin, and Frank Guerin. 2023b. Enhancing dialogue generation via dynamic graph knowledge aggregation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4604–4616, Toronto, Canada. Association for Computational Linguistics.

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 et al. 2023. Introducing mpt-7b: A new standard for open-source, ly usable llms.

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.

Kaiwen Wei, Xian Sun, Zequn Zhang, Jingyuan Zhang, Guo Zhi, and Li Jin. 2021. Trigger is not sufficient: Exploiting frame-aware knowledge for implicit event argument extraction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4672–4682, Online. Association for Computational Linguistics.

Kaiwen Wei, Yiran Yang, Li Jin, Xian Sun, Zequn Zhang, Jingyuan Zhang, Xiao Li, Linhao Zhang, Jintao Liu, and Guo Zhi. 2023. Guide the many-to-one assignment: Open information extraction via IoU-aware optimal transport. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4971–4984, Toronto, Canada. Association for Computational Linguistics.

Bohao Yang, Chen Tang, Kun Zhao, Chenghao Xiao, and Chenghua Lin. 2024. Effective distillation of table-based reasoning ability from LLMs. In *Proceedings of the 2024 Joint International Conference*

on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024), pages 5538–5550, Torino, Italia. ELRA and ICCL.

Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. 2023a. Generate rather than retrieve: Large language models are strong context generators.

Yue Yu, Jiaming Shen, Tianqi Liu, Zhen Qin, Jing Nathan Yan, Jialu Liu, Chao Zhang, and Michael Bendersky. 2023b. Explanation-aware soft ensemble empowers large language model in-context learning. *arXiv preprint arXiv:2311.07099*.

Weizhe Yuan, Graham Neubig, and Pengfei Liu. 2021. Bartscore: Evaluating generated text as text generation. In *Advances in Neural Information Processing Systems*, volume 34, pages 27263–27277. Curran Associates, Inc.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with BERT. *CoRR*, abs/1904.09675.

Kun Zhao, Bohao Yang, Chen Tang, Chenghua Lin, and Liang Zhan. 2024. Slide: A framework integrating small and large language models for open-domain dialogues evaluation. *arXiv preprint arXiv:2405.15924*.

A Appendix

A.1 Case Study

Table 8 shows that to become a teacher, a bachelor's degree and a teaching qualification are prerequisites, according to human responses. Additionally, they provide insights into different job-finding methods. The integration of LLM-Blender was influenced by the output of ChatGLM-6b. The answer emphasized methods of being an elementary school teacher, but the input question did not specify which stage of the teacher. Furthermore, it overlooked the important information of having a bachelor's degree and a teaching qualification certificate. Our URG method filters out low-relevance candidates during the filtering stage and is able to better overcome the influence of noisy input during the fusion stage, ultimately producing outputs that are close to human responses.

Model	Model score↑	bert-size	bart-score↑	B=8 (samp./s)↑	B=16 (samp./s)↑	B=32 (samp./s)↑	Norm(size) B=8 (samp./s)↑	Norm(size) B=16 (samp./s)↑	Norm(size) B=32 (samp./s)↑
SummaReranker	465M	0.7157	-3.3757	3.7037	3.8462	3.9950	1.7222	1.7885	1.8576
SimCLS	434M	0.7257	-3.3840	6.1538	8.2902	9.9688	2.6707	3.5979	4.3265
PairRanker	436M	0.7313	-3.2612	1.5385	1.6391	1.7440	0.6707	0.7146	0.7604
URG (770M)	770M	0.7385	-3.2120	**8.8889**	**10.6667**	**12.8000**	6.8445	8.2134	9.8560
URG (3B)	3B	**0.7510**	**-3.2057**	3.9216	3.9900	4.1026	**11.7648**	**11.9700**	**12.3078**

Table 7: Inference speed (samples/s) of different models on the MixInstruct benchmark test data, with batch sizes set to 8, 16, and 32. Each sample contains N=12 candidate results from LLMs. Norm(size) represents the corresponding inference speed when the model size is normalized to 1B.

Question: How can I get a job as a teacher?
Oasst-12b: To become a teacher, you will typically need to earn your bachelor's degree in education and obtain your teaching certification. Additionally, you need to gain experience through teaching substitutes or internships, as well as strong communication and interpersonal skills. You should also be prepared to work long hours, stay up to date on trends in education, and use innovative teaching methods.
Flan-T5-xxl: Get a degree in education.
Vicuna-13b: There are a few ways to land a job as a teacher: 1. Earn a bachelor's degree: To become a teacher, you will need to have a bachelor's degree. Some states require that you have a degree in education, while others allow you to teach in a subject area, such as math or science. 2. Obtain a teaching license: In order to teach in a public school, you will need to have a teaching license. Licensure requirements vary by state, but typically involve passing a certification exam and completing a teacher preparation program.
MOSS-sft: If you're interested in becoming a teacher, there are a few steps you should take: 1. Get a degree: Most states require a bachelor's degree in a subject area such as education, teaching, or a related field. Some may also require certification or licensure. 2. Obtain a teaching certification: Most states offer a teaching certification that allows you to teach at the high school level. You'll need to pass an exam and complete a teacher training program to obtain this certification. 3. Apply for jobs: Once you've completed your education and certification, you can search for job openings in your area. Many schools and districts offer positions for teachers through their websites or by visiting their websites directly.
ChatGLM-7b: Here's what you can do if you want to get a job as an elementary school teacher. You can start by looking at teacher job listings on websites like Indeed, Monster and network with other teachers. You can also look at job listings on the job boards or through school district websites. **llama-6b:** Becoming a teacher is a great opportunity for personal and professional growth, and it can also be a fulfilling career. You should have at least a high school education and a degree in the field you want to teach. You may want to consider taking courses in teaching principles, classroom management, and classroom technology.
MPT-instruct: How to become a teacher? The first step for becoming a teacher is to get your education. You can get a teacher education in various ways. There are teacher education programs at universities and teacher training centers and you can also get a job as a teacher on an apprenticeship program. If you want to get a job as a teacher in the primary school, you should have a teacher education for the primary school. If you want to teach in the secondary school, you should have a teacher education for the secondary school.
ChatGPT: To become a teacher, start by obtaining a bachelor's degree in education or a specific subject. Some states may require a teaching certification or a master's degree, depending on the level you plan to teach. Gain experience by doing internships or participating in student teaching programs. Network with professionals in the education field, attend career fairs or job fairs, and apply for teaching positions at schools or educational institutions. Prepare a resume that highlights your relevant skills and experiences, and be prepared for interviews and potential teaching demonstrations.
LLM-Blender: There are several ways to get a job as an elementary school teacher. You can start by looking for job openings on websites like Indeed, Monster, and Teachers-Teachers. Additionally, you can attend job fairs or networking events that may be held by schools you are interested in. Once you have identified the position you would like to apply for, submit your resume and cover letter to potential employers.
URG: To become a teacher, you will typically need to earn your bachelor's degree in education, obtain your teaching certification, and gain experience through teaching substitutes or internships. You can also look for job openings on job boards or through school district websites. It can be helpful to network with other teachers and attend job fairs to increase your chances of being hired.
Human Response: To become a teacher, you will typically need at least a bachelor's degree in education or a related field, as well as a teaching certification or license for the state in which you wish to teach. Additionally, gaining classroom experience through internships or student teaching programs can be beneficial when applying for teaching positions. Finally, you can visit job websites as well as the official websites of schools to contact schools recruiting teachers.

Table 8: Case study on the test set of MixInstruct. The table shows the responses from open-source LLMs and LLM-Blender, alongside ChatGPT, and the golden response from humans.