Improving LLMs' Learning for Coreference Resolution

Yujian Gan[1] **Yuan Liang**[1] **Yanni Lin**[2] **Juntao Yu**[1] **Massimo Poesio**[1,3]
[1]Queen Mary University of London
[2]Guangxi Normal University
[3]University of Utrecht
y.gan@qmul.ac.uk yuan.liang@qmul.ac.uk linyn@mailbox.gxnu.edu.cn
juntao.yu@qmul.ac.uk m.poesio@qmul.ac.uk

Abstract

Coreference Resolution (CR) is crucial for many NLP tasks, but existing LLMs struggle with hallucination and under-performance. In this paper, we investigate the limitations of existing LLM-based approaches to CR—specifically the Question-Answering (QA) Template and Document Template methods—and propose two novel techniques: Reversed Training with Joint Inference and Iterative Document Generation. Our experiments show that Reversed Training improves the QA Template method, while Iterative Document Generation eliminates hallucinations in the generated source text and boosts coreference resolution. Integrating these methods and techniques offers an effective and robust solution to LLM-based coreference resolution [1].

1 Introduction

Coreference resolution involves detecting and clustering different mentions that refer to the same discourse world entity. As a task that requires linguistic and extra-linguistic understanding, it plays a crucial role for many downstream natural language processing tasks, such as information extraction, text summarization, chatbots, and dialogue systems. As a result, CR has garnered significant attention from the NLP community (Poesio et al., 2023).

The evolution of coreference resolution models can be divided from rule-based and statistical learning approaches to deep learning approaches (Liu et al., 2023; Poesio et al., 2023). In the past few years, Large Language Models (LLMs), which implicitly incorporate contextual information and commonsense knowledge, have revolutionized NLP by significantly improving performance across many tasks. This advancement has led researchers to investigate LLMs' potential for coreference resolution.

Recent studies have already proved the feasibility of prompting LLMs to resolve coreferences, with a special focus on zero- and few-shot applications (Yang et al., 2022; Agrawal et al., 2022; Le and Ritter, 2024; Zhu et al., 2024; Gan et al., 2024). Le and Ritter (2024) have shown that prompt-based LLMs surpass previous unsupervised systems but still perform worse than state-of-the-art supervised models. In their experiments, two prompt templates were used, namely the Question-Answering (QA) Template and the Document Template, as shown in Figure 1. Specifically, Le and Ritter (2024) have focused more on the Document Template, while Gan et al. (2024) have concentrated on the QA Template. Experimental results in our comparative study show that the performance of the Document Template is superior to that of the QA Template.

However, it is found that the Document Template method relies on a key assumption: LLMs do not generate hallucinations when resolving coreferences. But LLMs do produce hallucinations, which is challenging to match the generated document with the correct places in the original text. For example, in Figure 1, if the LLM wrongly generates "There are a candle a wall ..." as "There are a candle a candle a wall ..." with an unnecessary mention "a candle" repeated, it becomes much harder to align these mentions with the original document. Through studying the code given by Le and Ritter (2024), we have noticed that they removed examples where LLMs created hallucinations, only keeping cases where the original text matched correctly (i.e., without hallucinations). However, we have found that hallucinations are quite common, especially in linguistically complex texts, where the problem becomes even more noticeable. Although the QA Template method is less affected by hallucinations, its performance is weaker than the Document Template method, making it less reliable for coreference resolution.

[1]Our code is available here.

Proceedings of the 26th Annual Meeting of SIGDIAL, pages 311–321
Aug 25–27, 2025.

To overcome the limitations of both the QA and Document Template methods, we propose two new approaches: Reversed Training with Joint Inference, and Iterative Document Generation. Our experiments show that the Iterative Document Generation method not only completely removes hallucinations of LLMs but also improves the final CoNLL score. Meanwhile, the Reversed Training with Joint Inference method significantly improves the LLM's CR performance when using the QA Template. Ablation studies also show that removing any component of this solution reduces the CoNLL score.

Overall, our work makes two key contributions:

1. We propose the Reversed Training and Joint Inference methods for coreference resolution based on the QA Template. Ablation study shows that these methods effectively improve LLM performance.

2. We propose the Iterative Document Generation method for coreference resolution based on the Document Template. Experiments demonstrate that this method not only enhances LLM performance, but also effectively tackles the issue of hallucination in LLMs when using the Document Template.

2 Related Work

2.1 Traditional Coreference Resolution Systems

Typically, traditional coreference resolution systems have addressed the task through mention-ranking models. Early approaches, such as those by Wiseman et al. (2015) and Clark and Manning (2016), used a two-step process. First, a mention detector identified potential mentions in the text. Then, a separate clustering step grouped these mentions into coreference clusters. Lee et al. (2017) introduced a pivotal advancement with a deep learning system that jointly performed mention detection and clustering. This end-to-end approach proved to be both effective and simple, becoming the standard architecture for coreference resolution for a considerable period.

Building on this foundation, subsequent work explored various enhancements. Lee et al. (2018) and Kantor and Globerson (2019) incorporated contextual embeddings to improve performance. Yu et al. (2020) focused on addressing the challenges posed by singletons and non-referring expressions. Joshi et al. (2019, 2020) further refined the model by replacing LSTMs with fine-tuned BERT and SpanBERT, harnessing the power of transformer architectures.

While these advancements refined the core architecture, research has also began to shift towards exploring alternative approaches, such as reformulating the task using question-answering frameworks and document annotation templates.

2.2 Coreference Resolution as Question Answering

One line of studies have investigated the potential of framing coreference resolution as a question-answering task.

Wu et al. (2020) recast coreference resolution as the task of finding all other mentions (answers) that belong to the same cluster as a given mention (question). They utilized SpanBERT, pre-trained on Quoref and SQuAD 2.0, to encode the document and employed a BIO scheme to tag valid answers for each mention. Their findings highlighted the importance of considering scores from both directions (i.e., $S_{i,j}$ and $S_{j,i}$ for a mention pair *i*, *j*), which resonates with our observations on reversed training discussed later in this paper (Section 4.1). Aralikatte et al. (2021) adopted a BERT-based machine reading comprehension (MRC) approach, focusing on resolving ellipsis by identifying the answer span (antecedent) for a given mention. Yang et al. (2022) evaluated early GPT models (e.g., GPT-2) on the ECB+ corpus in a few-shot setting, using a QA Template for binary judgments on gold mention pairs. However, this early attempt did not achieve significant results. Bohnet et al. (2023) achieved state-of-the-art results by fine-tuning the mT5-XXL (13B) model to output candidate mentions and their corresponding clusters.

More recently, Gan et al. (2024) have demonstrated prompt-based instruction-tuned language models was feasible to resolve coreference. One of the two templates used in their study was the Question-Answering (QA) Template, where the language model generates the answer when given a passage and an open-ended *wh*-question. With further exploration on this QA Template, Gan et al. (2024) have conducted a comprehensive evaluation on the capabilities of different LLMs to resolve coreferences. In Gan et al. (2024)'s QA Template, the antecedent and its sentence ID was generated in a pair. In the same year, Zhu et al. (2024) have conducted a similar evaluation of several recent

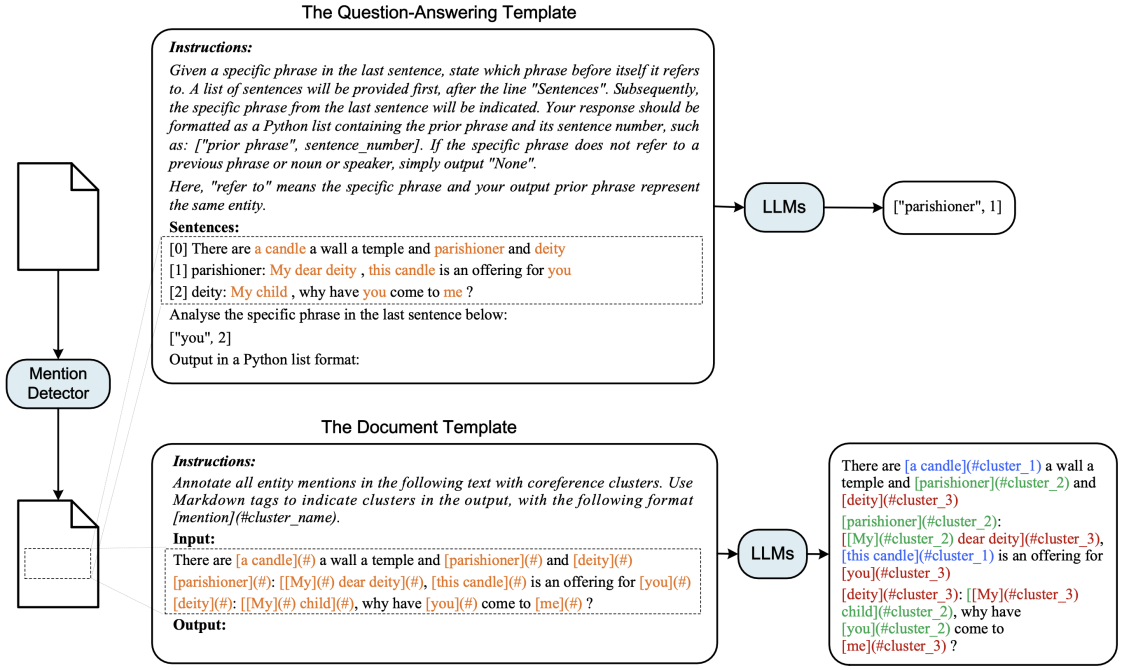

Figure 1: An example of the QA and Document Templates for Coreference Resolution. In the QA Template, LLMs generate answers based on input content and questions. In the Document Template, LLMs analyze the marked mention and assign a cluster ID to it.

LLMs on prompt-based coreference resolution as a component of context understanding. A key difference from Gan et al. (2024)'s work is that Zhu et al. (2024) formulated the task as a multiple-choice problem.

2.3 Document Annotation-based Coreference Resolution

Another line of research have explored the use of document annotation for coreference resolution.

Zhang et al. (2023) introduced a seq2seq approach using a prompt-fine-tuned T0 model. This system took a document as input and produces the same document with annotated mentions and clusters. Their results demonstrated that this approach can achieve performance comparable to that of Bohnet et al. (2023). Le and Ritter (2024) have proposed a prompt-based method where LLMs were provided with documents containing highlighted mentions (either predicted or gold) followed by a special cluster placeholder (#). The LLMs were then tasked with predicting and filling the placeholders with appropriate cluster IDs. However, our preliminary experiments with this method revealed issues including hallucination. Our solutions to this problem will be given in Section 5.

Our work mainly builds upon the approach of Le and Ritter (2024) and Gan et al. (2024), further refining the QA and Document Template to achieve significant improvements.

2.4 Reverse Training

The "Reversal Curse" describes LLMs' inability to understand the symmetric property of identify relation, i.e., LLMs trained on "A is B" may struggle to learn "B is A" (Berglund et al., 2024).

The Reversal Curse has been remedied by the recent attempts of reverse training. Yu et al. (2024) have demonstrated that LLMs can learn from modeling in both directions with comparable proficiency. In their study, a reverse training sample was constructed by directly reversing the original token sequence. Golovneva et al. (2024) have proved the effectiveness of training by reversing the ordering of segmented text chunks. In addition to positional relationship in linear sequences, the concept of "reverse" can be generalized to more types of association in many cases, where information presented in the reverse order is no less valuable than that in the original order. In multi-modal tasks, for example, Gul and Artzi (2024) have reversed the input of image description and the output of image classification.

Inspired by the previous work, we attempt to introduce reverse training to coreference resolution, as many coreferential relationships can be deemed as bidirectional, thus increasing the amount of useful information for model training.

3 LLM-Based Coreference Resolution

3.1 The Question-Answering (QA) Template

A Question-Answering (QA) Prompt Template is a structured format that guides LLMs in generating accurate responses to questions. It follows the structure $Instruction + Sentences + Question$ for coreference resolution. The first part $Instruction$ guides the LLM in resolving candidate mentions. The second part $Sentences$ provides the necessary context in a set of sentences, including the one containing the target mention. The last part $Question$ specifies the target mention to analyze to find its coreferential mention. An example of the QA Template can be seen in Figure 1.

3.2 The Document Template

The Document Template combines an instruction with a document for processing. The document marks candidate mentions and designates character positions where the LLM must output cluster IDs for these mentions. These cluster IDs group coreferential mentions together, with all mentions in a cluster referring to the same entity. This structure enables the LLMs to perform coreference resolution. An example of the Document Template can be seen in Figure 1.

3.3 Supervised Fine-Tuning for Coreference Resolution

To improve LLMs' performance in coreference resolution, we converted existing datasets into the two template formats described above for supervised fine-tuning (SFT). This process adapted open-source LLMs to the coreference resolution task.

Using the QA Template as an example, given a coreference dataset D, each document d contains multiple mentions $m_k, k = 1, 2, \ldots, n$. Each queried mention m_k has its corresponding resolved mention r_k as the answer. By inserting d and m_k into the QA Template, we create a prompt p_i and form a training data pair (p_i, rp_i), where rp_i equals r_k. This process transforms the dataset D into a training dataset $T = (p_i, rp_i)$. Using this training dataset T and a large language model M, SFT aims to minimize the loss to get the optimal M^*:

$$\min_{M} \Sigma_{i=1}^{T} L(M(pi), rp_i) \quad (1)$$

After fine-tuning, we use the optimal model M^* for inference. The inference data will also be converted into the corresponding template format.

4 The Refined QA Template Approach

4.1 Reversed Training for Conference Resolution

Based on the previous work, we reverse the pattern "A refers to B" in the QA Template (input A, output B) to "B refers to A" (input B, output A). We propose two training modes: forward training and backward training. In this context, forward training is exemplified in Figure 1. In the backward mode, however, as the position of the coreferential mention is uncertain, the backward prompt requires the entire document to be provided.

However, unlike other tasks, forward and backward training alone are not sufficient for the CR task, because it also involves singletons that do not refer to any other mentions. To address this issue, we designed a new question task specifically to identify mentions without antecedents:

> "List all phrases in the last sentence that do not refer to any previous phrases preceding them. Your response should be formatted as a Python list containing all phrases from the candidate list that do not refer to any preceding phrases. If no such phrases can be found, simply return 'None.'"

In summary, we designed three different QA tasks for reversed training: forward training, backward training, and finding all singletons.

4.2 Learning to Generate a Chain

As discussed in Section 4.1, one may assume a canonical case that the target mention refers to only one coreferential mention, be it the antecedent in the forward variation, or the anaphor in the backward variation. Actually, for a target mention, the number of coreferential mentions generated by LLM can be more than one $(n, n = 0, 1, 2, ..., i)$. Specially, when $n = 0$, the target mention is a singleton that fails to form a chain; when $n = 1$, the target mention and its coreferential mention form a coreference chain with the minimal length. Prompting LLMs to generate multiple coreferential mentions, if any, may provide more information that model training may benefit from. Thus, we advocate the method of learning to generate a coreference chain, rather than a single mention. In practice, we train the LLM to identify the two most recent mentions, as shown in the second and third panels of Figure 2.

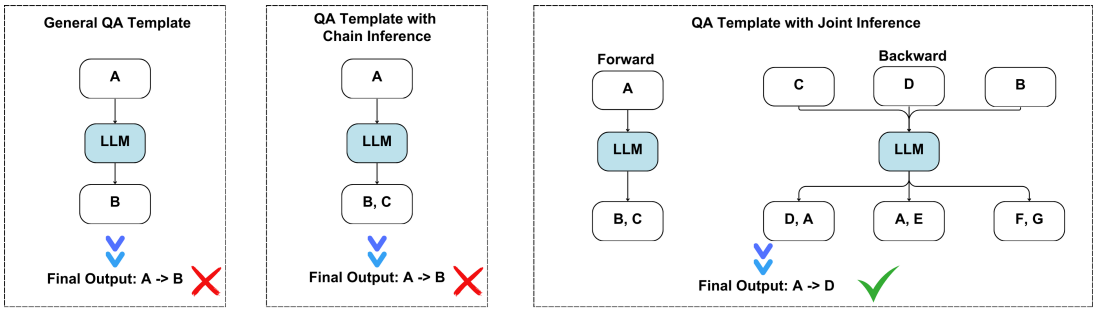

Figure 2: A comparison between different inference approaches.

4.3 Joint Inference

Joint Inference is based on reversed training and learning to generate a chain. After the model generates results for forward, backward, and singleton tasks, we compare the multiple outputs to obtain the final correct answer. Figure 2 illustrates an example of Joint Inference. In this case, assuming the LLM incorrectly infers that A refers to B, we can correct this mistake by comparing the outputs of backward tasks C, D, and B, ultimately correcting that A refers to D.

The joint inference algorithm takes two inputs: a set of forward pairs FP and a set of backward pairs BP. The algorithm first assigns initial weights to pairs in both groups, then updates these weights and stores the sum of weights for pairs appearing in both BP and FP in W. Pairs with weights exceeding 2 are used to create longer chains C, which are considered correct. Finally, the algorithm combines the pair weights W and chains to generate the final output $(anaphor, antecedent)$.

How to combine the pair weights W and chains C to generate the final output? The process iterates through all mentions in the weight list W. For each mention a, if any candidate referent $b, b \in W[a]$ has a weight $W[a][b] \geq 2$, the referent of a is considered found. Otherwise, the algorithm examines pairs of candidate referents $b, b \in W[a]$ and $d, d \in W[a]$. If both b and d co-occur in any chain C, their respective weights in W are incremented by 1. This process updates all pair weights using the chain information. After updating is complete, the algorithm selects the referent phrase with the highest weight for each mention to create the final pair $(anaphor, antecedent)$.

5 Iterative Document Generation

Le and Ritter (2024)'s Document Template provides the complete document with marked candidate mentions and designated character positions.

Input: There are [a candle](#
Prediction: cluster_1

Input: There are [a candle](#cluster_1) a wall a temple and [parishioner](#
Prediction: cluster_2

Input: There are [a candle](#cluster_1) a wall a temple and [parishioner](#cluster_2) and [dety](#
Prediction: cluster_3

Input: There are [a candle](#cluster_1) a wall a temple and [parishioner](#cluster_2) and [dety](#cluster_3). [parishioner](#
Prediction: cluster_2

Figure 3: An example of the iterative document generation process of the refined Document Template. For each iteration, the input consists of a text segment containing a marked mention and its designated position at the end. The LLMs then predict the cluster ID for the marked mention. This predicted ID and its following content are appended to the end, creating a new prompt for analyzing the next mention. This sequential process continues till all mentions have been analyzed.

The LLMs must input cluster IDs for these candidate mentions simultaneously at the designated positions. However, we discovered that processing the entire document and analyzing all mentions at once led to hallucination—where the LLM would generate irrelevant words not present in the document instead of completing the intended task. This issue significantly reduced the model's UASA score from approximately 89 (when excluding hallucination results from evaluation) to 59 (when including hallucination results).

To address this issue, we modified the full document generation process to generate only one mention's cluster ID at a time, as shown in Figure 3. The advantage of generating just one cluster ID per step is that it significantly reduces the difficulty of content generation for the LLM, helping to avoid hallucinations. The process begins with a text seg-

ment containing the document's first marked mention and its designated position. The LLM assigns a cluster ID to this mention. After filling this predicted cluster ID into the designated position, a new text segment is created for analyzing the next mention. This sequential analysis continues until all mentions in the document are processed. During training, the cluster IDs are filled with correct values. During inference, they are filled with IDs generated by the LLMs.

In prompt design, since only a cluster ID is predicted, using just the partial sentence from Figure 3 may not be enough for accurate prediction. Considering the context is crucial for understanding the coreference relationship of the mention. Therefore, based on the Document Template in Figure 1, we removed the full text from the input and instead added the relevant sentence containing the last mention. Unlike in Figure 1, this relevant sentence does not include the mention's position annotation.

6 Experiment

6.1 Experimental Setup

Dataset. We used the same datasets as those of Gan et al. (2024) and Le and Ritter (2024): OntoNotes 5.0 (Pradhan et al., 2012) and CODI/CRAC (Poesio et al., 2018; Yu et al., 2022). We utilize three subsets of CODI/CRAC: LIGHT, Penn Treebank, and TRAINS. The training set sizes vary significantly across datasets, with OntoNotes having the largest set, while LIGHT, Penn Treebank, and TRAINS comprise 2.73%, 24.95%, and 10.21% of OntoNotes' training set size, respectively. Consistent with the experimental setup of previous work, we used gold mentions. It should be noted that the QA Template's prompt does not specify the exact mention in the question, making it slightly more challenging compared to the Document Template.

To ensure consistency with prior work, we use a subset of the test set for comparison with Gan et al. (2024), and the full test set for comparison with Le and Ritter (2024). The four datasets used in this study are shown as follows:

- **OntoNotes**: We extracted 2,516 training documents from its training set, resulting in 128,319 training examples for the QA Template. We randomly selected 55 test documents from its test set, yielding 3,624 test examples for the QA Template.
- **LIGHT**: We used the full LIGHT training set, resulting in 3,499 training examples for the QA Template, and randomly selected 4 test documents, yielding 872 test examples.
- **Penn Treebank**: We used the full Penn Treebank training set, resulting in 31,979 training examples for the QA Template, and randomly selected 5 test documents, yielding 1067 test examples.
- **TRAINS**: We used the full Penn Treebank training set, resulting in 13,087 training examples for the QA Template, and randomly selected 4 test documents, yielding 757 test examples.
- **Combination**: We combined the training sets from LIGHT, Penn Treebank, and TRAINS. No test set was used.

Models. To validate the feasibility of our method, we fine-tuned two different large language models (LLMs) as base models: LLama 3.1 (Grattafiori et al., 2024) and Qwen 2.5 (Qwen et al., 2025). All experiments were conducted using LLAMA Factory (Zheng et al., 2024), with LoRA-based (Hu et al., 2021) fine-tuning utilizing the bf16 precision. The relevant hyperparameters were configured as follows:

- preprocessing_num_workers: 16
- per_device_train_batch_size: 1
- gradient_accumulation_steps: 4
- learning_rate: 1.0e-4
- num_train_epochs: 3.0
- lr_scheduler_type: cosine
- warmup_ratio: 0.1

Other parameters were set to the default values provided by LLAMA Factory (Zheng et al., 2024). The experiments were conducted on single or multiple A100 GPUs (40GB/80GB). For multi-GPU setups, DeepSpeed was used to optimize parallel training.

The symbols associated with the models in this section are defined as follows:

- **LLAMA** The LLama 3.1 8B Instruct model.
- **QWEN** The Qwen 2.5 7B Instruct model.
- $\mathbb{R}$ indicates that the model is trained using reversed learning based on the QA Template.
- $\mathbb{S}$ indicates that the model is trained using standard fine-tuning based on the QA Template.
- $\mathbb{C}$ represents that the model is trained on the Combination set.
- $\mathbb{OC}$ indicates that the model is trained on the OntoNotes, and then evaluated separately on CODI/CRAC and OntoNotes.
- $\mathbb{L}$ indicates that the model is trained to predict

	LIGHT		Penn Treebank		TRAINS		OntoNotes	
Approach	**UA**	**-S -SA**	**UA**	**-S -SA**	**UA**	**-S -SA**	**UA**	**-S -SA**
LLAMA $\mathbb{S}^\dagger$	84.02	70.39	83.83	70.60	78.63	75.03	82.86	84.05
LLAMA $\mathbb{R}$	85.77	73.20	86.86	77.12	79.20	74.68	81.23	82.58
LLAMA $\mathbb{SL}$	86.64	75.69	86.09	74.47	79.58	75.07	81.80	82.83
LLAMA $\mathbb{RL}$	88.20	77.81	**87.99**	**78.56**	81.76	78.36	**83.04**	**84.60**
LLAMA $\mathbb{SC}$	87.25	75.54	85.98	74.90	78.18	73.38		
LLAMA $\mathbb{RC}$	87.30	76.18	85.78	74.84	81.97	77.92		
LLAMA $\mathbb{SCL}$	87.20	77.09	85.46	74.25	82.69	78.37		
LLAMA $\mathbb{RCL}$	**89.29**	**80.40**	87.46	77.86	**83.67**	**81.02**		
QWEN $\mathbb{SL}$	82.66	68.35	85.72	73.66	80.79	76.48		
QWEN $\mathbb{RL}$	**87.08**	**76.49**	**86.19**	**75.10**	**82.62**	**79.64**		
QWEN $\mathbb{S}^\dagger$	82.48	67.50	84.22	71.54	78.14	72.48		
QWEN $\mathbb{R}$	86.35	75.19	85.98	74.93	79.54	74.59		

Table 1: Comparison of the training effects between Reversed Learning and fine-tuning based on original data. † denotes results reproduced from Gan et al. (2024). Our reproduced results outperform those reported by Gan et al. (2024) for two main reasons: (1) we fixed an evaluation issue in their original setup, where non-referring expressions were not excluded during scoring; and (2) our model architecture differs from theirs, which also contributes to the performance gains. To facilitate a lightweight comparison, we conducted testing on a randomly selected subset of the test dataset only, which is sufficient to verify the effectiveness of our method.

a coreference chain (i.e., two mentions instead of just one) based on the QA Template. If the model also uses reversed learning (denoted by $\mathbb{RL}$), joint inference will automatically be applied in this case.

- $\mathbb{F}$ indicates that the model is trained to predict a full document based on the Document Template.
- $\mathbb{I}$ indicates that the model is trained using the iterative document generation method based on the Document Template.

Metrics. For the QA Template, we use the Universal Anaphora (UA) scorer (Poesio et al., 2024) to evaluate different approaches, where UA represents the full score, and '-S -SA' indicate the CoNLL score (i.e., without singletons and split antecedents). For the Document Template, consistent with the experiment setup of Le and Ritter (2024)[2], we use '-SA' to indicate UA without split antecedents for evaluation and comparison across different methods. We also calculate the Pass score and Exact Match (EM) score to quantify the extent of hallucinations. "Pass" denotes the percentage of documents that passed the alignment check as defined in Le and Ritter (2024)'s work. "EM" refers to the exact match rate between the generated text and the original reference text.

Baselines. For both the QA and the Document Templates, we adopted the approaches proposed in Le and Ritter (2024)'s work as our baseline methods. However, their experiments were conducted using LLama 2, which is now somehow outdated. Recently, the LLama 3.1 8B and Qwen 2.5 have been released, featuring architectural improvements and training on a significantly larger dataset, resulting in noticeably enhanced performance.

To establish updated baseline results for fair comparison, we fine-tuned LLama 3.1 8B and Qwen 2.5 using the same methods described by Le and Ritter (2024). In our experiments, the fine-tuned models are denoted with $\mathbb{S}$ for the QA Template and $\mathbb{F}$ for the Document Template.

6.2 QA Template Results

From the results shown in Table 1, we observe that models trained with reversed learning generally outperformed those trained using standard fine-tuning across most datasets, in terms of both the full score (UA) and the CoNLL score excluding singletons and split antecedents (denoted as -S -SA). This held

[2]The Reference Coreference Scorer (Pradhan et al., 2014) does not score singletons (-S -SA). However, Le and Ritter (2024) retrained singletons in their evaluation. This may not be a big problem when evaluating on corpora without singleton mentions (e.g., OntoNotes), as singletons will only appear in system clusters. However, in corpora with singleton mentions (e.g., CODI/CRAC), this difference in setting can result in a change of more than 10% in CoNLL F_1. To make a fair comparison, we followed the same settings when compared to their system.

	LIGHT		Penn Treebank		TRAINS	
Approach	**UA**	**-S -SA**	**UA**	**-S -SA**	**UA**	**-S -SA**
LLAMA $\mathbb{RL}$	88.20	77.81	87.99	78.56	81.76	78.36
w/o Join Inference	87.95	77.74	87.56	77.83	81.45	77.46
w/o $\mathbb{R}$	86.64	75.69	86.09	74.47	79.58	75.07
w/o $\mathbb{L}$	84.02	70.39	83.83	70.60	78.63	75.03
LLAMA $\mathbb{RCL}$	89.29	80.40	87.46	77.86	83.67	81.02
w/o Join Inference	88.46	78.87	86.79	76.55	83.08	79.88
w/o $\mathbb{R}$	87.20	77.09	85.46	74.25	82.69	78.37
w/o $\mathbb{L}$	87.25	75.54	85.98	74.90	78.18	73.38
QWEN $\mathbb{RL}$	87.08	76.49	86.19	75.10	82.62	79.64
w/o Join Inference	86.59	76.66	85.99	74.46	81.13	77.52
w/o $\mathbb{R}$	82.66	68.35	85.72	73.66	80.79	76.48
w/o $\mathbb{L}$	82.48	67.50	84.22	71.54	78.14	72.48

Table 2: Ablation Analysis of Different QA Template Approaches.

true for both the LLama 3.1 and Qwen 2.5 models.

For both LLMs, reversed learning consistently yielded higher performance across all datasets, except for the comparison between LLAMA $\mathbb{S}$ and LLAMA $\mathbb{R}$ on the OntoNotes dataset. This suggests that when the training dataset is sufficiently large (128,319 training examples in OntoNotes) and diverse, it is enough for the models to learn the task. After adding Joint inference, the performance of LLAMA $\mathbb{RL}$ successfully surpassed that of LLAMA $\mathbb{SL}$ and LLAMA $\mathbb{S}$. This indicates that the method proposed in this paper indeed enhances the model's ability to solve CR under the QA Template.

On the other hand, for smaller datasets like LIGHT, Penn Treebank, TRAINS, and their combination (which vary in size from 3,499 to 48,565 examples), reversed learning consistently outperformed standard fine-tuning, highlighting its advantage in situations where training data is limited. For Qwen 2.5, similar trends can be observed. Among all the comparisons, the largest improvement was seen when transitioning from QWEN $\mathbb{SL}$ to QWEN $\mathbb{RL}$. Specifically, the 'UA' score increased by 4.56 (from 82.66 to 87.08) and the '-S -SA' score improved by 8.14 (from 68.35 to 76.49).

Ablation Study. To understand the contributions of each component, we conducted an ablation study by removing key elements from the model. We examined three components: Joint Inference, Reversed Learning ($\mathbb{R}$), and coreference chain prediction ($\mathbb{L}$). Results in Table 2 show their impact on performance across the LIGHT, Penn Treebank, and TRAINS datasets.

It is clear that the best performance is achieved when all components are present. Table 2 confirms that each component clearly contributes to the model's overall performance. Removing any of these components leads to a reduction in performance, highlighting the effectiveness of the proposed approach in solving the coreference resolution task using the QA Template.

6.3 Document Template Results

From Table 3, we observe that hallucinations were severe in full document generation, as indicated by significantly lower EM scores and alignment check pass rates in the $\mathbb{OCF}$ setting. In contrast, our proposed iterative generation method ($\mathbb{OCI}$) not only eliminated hallucinations—achieving a 100% pass rate on the alignment check across all datasets—but also significantly improved the Exact Match (EM) scores.

The EM metric provides a stricter evaluation of output fidelity, measuring whether the generated text exactly matches the gold reference. Our method achieved 100% EM across all datasets, underscoring its ability to produce highly faithful outputs. In comparison, full document generation often failed to match reference texts exactly, with EM scores ranging from 62.50% to 92.11% depending on the dataset and model. This further supports our claim that iterative generation produces more precise and consistent outputs.

Moreover, even when focusing solely on documents that passed the alignment check, our method still consistently outperformed full generation in EM, revealing that alignment check alone may not capture finer-grained errors that EM can detect. This discrepancy is especially apparent in the TRAINS dataset. Although the alignment pass rate was 100% for both $\mathbb{OCF}$ and $\mathbb{OCI}$, EM for $\mathbb{OCF}$ remained low (62.50%–81.25%), suggesting that its outputs, while structurally aligned, still contained subtle inconsistencies or content deviations.

These results collectively demonstrate that full document generation lacks robustness and struggles to achieve both structural alignment and content fidelity. Our iterative approach not only ensures alignment but also guarantees high-fidelity generation, as reflected in both pass rates and exact match accuracy.

7 Discussion

7.1 Why does the Document Template Work Better?

The Document Template excels because it explicitly marks all mentions, guiding the LLM to focus only on relevant parts of the text. This structured approach reduces the complexity, as the model does

	LIGHT			Penn Treebank			TRAINS			OntoNotes		
Approach	**-SA**	**Pass**	**EM**	**-SA**	**Pass**	**EM**	**-SA**	**Pass**	**EM**	**-SA**	**Pass**	**EM**
LLAMA $\mathbb{OCF}^{\dagger}$	86.6	92.11%	81.58%	82.5	100%	70%	73.4	100%	62.50%	**95.3**	99.71%	97.41%
LLAMA $\mathbb{OCI}$	**87.8**	100%	100%	**85.2**	100%	100%	**76.4**	100%	100%	94.9	100%	100%
QWEN $\mathbb{OCF}^{\dagger}$	**87.0**	97.37%	92.11%	**83.2**	98.33%	71.67%	77.2	100%	81.25%	**95.0**	99.14%	97.13%
QWEN $\mathbb{OCI}$	85.7	100%	100%	80.6	100%	100%	**79.6**	100%	100%	94.8	100%	100%

Table 3: Comparison of Training Outcomes Between Iterative and Full Document Generation on the full test dataset. † denotes results reproduced from Le and Ritter (2024). Tables 1 and 2 are based on a subset of the test set, whereas this table uses the full test set to ensure a fair comparison with Le and Ritter (2024), which uses the complete test set.

not need to identify appropriate mentions within the entire document, as it does with the QA Template. Essentially, it simplifies the task to something akin to a classification problem, where the LLM identifies coreference relations within predefined segments, making it easier and more accurate.

7.2 Do We Still Need the QA Template?

From a purely CR perspective, the Document Template is indeed better because the task is simpler and easier to train. However, the goal of CR research is not just to resolve coreferences but also to enhance the model's understanding of logical relationships in a language. While the Document Template performs better, it may only train a classification model that relies on mention annotations or detection, which limits its integration with other tasks. We consider it essentially a classification model because, in the Document Template task, the large language model only needs to generate the cluster ID to which a given mention belongs. In other words, it simply has to decide whether the mention belongs to a new cluster (a new cluster ID) or an existing one (an existing cluster ID). In contrast, the QA Template requires no mention annotations in the prompt, and the trained model is likely to have a better understanding of language, making it easier to integrate with other tasks. Therefore, in the long run, learning from the QA Template remains irreplaceable, particularly for cross-task applications and improving language comprehension.

8 Conclusion

In this paper, we have analyzed the limitations of current approaches to using LLMs for coreference resolution, particularly the QA and Document Templates, and then proposed two novel methods: Reversed Training with Joint Inference, and Iterative Document Generation. Our findings can be summarized as follows. First, Reversed Training enhances the QA Template method. Second, Iterative Document Generation eliminates hallucinations with an improvement in task performance. These methods and techniques proposed in our study form an integrated solution to enhance model learning in LLM-based coreference resolution.

Limitations

Although the methods proposed in this paper have been experimentally validated on English coreference resolution tasks, they have not been extensively tested on coreference tasks in other languages. Therefore, we are currently unable to assess the applicability of these methods to other languages. Additionally, due to limited computational resources, the methods have not been tested on larger-scale LLMs (e.g., models with 70B parameters or more), so we are uncertain about their performance of such models.

Acknowledgments

We thank the anonymous reviewers for their helpful comments. Yujian Gan, Juntao Yu and Massimo Poesio acknowledge financial support from the UK EPSRC under grant EP/W001632/1.

References

Monica Agrawal, Stefan Hegselmann, Hunter Lang, Yoon Kim, and David Sontag. 2022. Large language models are few-shot clinical information extractors. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1998–2022, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Rahul Aralikatte, Matthew Lamm, Daniel Hardt, and Anders Søgaard. 2021. Ellipsis resolution as question answering: An evaluation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 810–817, Online. Association for Computational Linguistics.

Lukas Berglund, Meg Tong, Max Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak,

and Owain Evans. 2024. The reversal curse: Llms trained on "a is b" fail to learn "b is a". *Preprint*, arXiv:2309.12288.

Bernd Bohnet, Chris Alberti, and Michael Collins. 2023. Coreference resolution through a seq2seq transition-based system. *Transactions of the Association for Computational Linguistics*, 11:212–226.

Kevin Clark and Christopher D. Manning. 2016. Improving coreference resolution by learning entity-level distributed representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 643–653, Berlin, Germany. Association for Computational Linguistics.

Yujian Gan, Massimo Poesio, and Juntao Yu. 2024. Assessing the capabilities of large language models in coreference: An evaluation. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 1645–1665, Torino, Italia. ELRA and ICCL.

Olga Golovneva, Zeyuan Allen-Zhu, Jason Weston, and Sainbayar Sukhbaatar. 2024. Reverse training to nurse the reversal curse. *Preprint*, arXiv:2403.13799.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Mustafa Omer Gul and Yoav Artzi. 2024. CoGen: Learning from feedback with coupled comprehension and generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 12966–12982, Miami, Florida, USA. Association for Computational Linguistics.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Mandar Joshi, Omer Levy, Luke Zettlemoyer, and Daniel Weld. 2019. BERT for coreference resolution: Baselines and analysis. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5803–5808, Hong Kong, China. Association for Computational Linguistics.

Ben Kantor and Amir Globerson. 2019. Coreference resolution with entity equalization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 673–677, Florence, Italy. Association for Computational Linguistics.

Nghia T. Le and Alan Ritter. 2024. Are language models robust coreference resolvers? In *First Conference on Language Modeling*.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen, Denmark. Association for Computational Linguistics.

Kenton Lee, Luheng He, and Luke Zettlemoyer. 2018. Higher-order coreference resolution with coarse-to-fine inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 687–692, New Orleans, Louisiana. Association for Computational Linguistics.

Ruicheng Liu, Rui Mao, Anh Tuan Luu, and Erik Cambria. 2023. A brief survey on recent advances in coreference resolution. *Artificial Intelligence Review*, 56(12):14439–14481.

Massimo Poesio, Yulia Grishina, Varada Kolhatkar, Nafise Moosavi, Ina Roesiger, Adam Roussel, Fabien Simonjetz, Alexandra Uma, Olga Uryupina, Juntao Yu, and Heike Zinsmeister. 2018. Anaphora resolution with the ARRAU corpus. In *Proc. of the NAACL Worskhop on Computational Models of Reference, Anaphora and Coreference (CRAC)*, pages 11–22, New Orleans.

Massimo Poesio, Maciej Ogrodniczuk, Vincent Ng, Sameer Pradhan, Juntao Yu, Nafise Sadat Moosavi, Silviu Paun, Amir Zeldes, Anna Nedoluzhko, Michal Novák, Martin Popel, Zdeněk Žabokrtský, and Daniel Zeman. 2024. Universal anaphora: The first three years. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 17087–17100, Torino, Italia. ELRA and ICCL.

Massimo Poesio, Juntao Yu, Silviu Paun, Abdulrahman Aloraini, Pengcheng Lu, Janosch Haber, and Derya Cokal. 2023. Computational models of anaphora. *Annual Review of Linguistics*, 9:561–587.

Sameer Pradhan, Xiaoqiang Luo, Marta Recasens, Eduard Hovy, Vincent Ng, and Michael Strube. 2014. Scoring coreference partitions of predicted mentions: A reference implementation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 30–35, Baltimore, Maryland. Association for Computational Linguistics.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 1–40, Jeju Island, Korea. Association for Computational Linguistics.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Sam Wiseman, Alexander M. Rush, Stuart Shieber, and Jason Weston. 2015. Learning anaphoricity and antecedent ranking features for coreference resolution. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1416–1426, Beijing, China. Association for Computational Linguistics.

Wei Wu, Fei Wang, Arianna Yuan, Fei Wu, and Jiwei Li. 2020. CorefQA: Coreference resolution as query-based span prediction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6953–6963, Online. Association for Computational Linguistics.

Xiaohan Yang, Eduardo Peynetti, Vasco Meerman, and Chris Tanner. 2022. What GPT knows about who is who. In *Proceedings of the Third Workshop on Insights from Negative Results in NLP*, pages 75–81, Dublin, Ireland. Association for Computational Linguistics.

Juntao Yu, Sopan Khosla, Ramesh Manuvinakurike, Lori Levin, Vincent Ng, Massimo Poesio, Michael Strube, and Massimo Poesio. 2022. The CODI/CRAC 2022 shared task on anaphora resolution, bridging and discourse deixis in dialogue. In *Proc. of CODI/CRAC Shared Task*.

Juntao Yu, Alexandra Uma, and Massimo Poesio. 2020. A cluster ranking model for full anaphora resolution. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 11–20, Marseille, France. European Language Resources Association.

Sicheng Yu, Yuanchen Xu, Cunxiao Du, Yanying Zhou, Minghui Qiu, Qianru Sun, Hao Zhang, and Jiawei Wu. 2024. Reverse modeling in large language models. *ArXiv*, abs/2410.09817.

Wenzheng Zhang, Sam Wiseman, and Karl Stratos. 2023. Seq2seq is all you need for coreference resolution. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 11493–11504, Singapore. Association for Computational Linguistics.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.

Yilun Zhu, Joel Ruben Antony Moniz, Shruti Bhargava, Jiarui Lu, Dhivya Piraviperumal, Site Li, Yuan Zhang, Hong Yu, and Bo-Hsiang Tseng. 2024. Can large language models understand context? In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 2004–2018, St. Julian's, Malta. Association for Computational Linguistics.