Low-Hallucination and Efficient Coreference Resolution with LLMs

Yujian Gan[1,2], Yuan Liang[2], Jinxia Xie[3],
Yanni Lin[4*], Juntao Yu[2], Massimo Poesio[2,5]
[1]Guangxi University of Finance and Economics [2]Queen Mary University of London
[3]Nanjing University of Science and Technology
[4]Guangxi Normal University [5]University of Utrecht
y.gan@qmul.ac.uk yuan.liang@qmul.ac.uk jinxia_xie@njust.edu.cn
linyn@mailbox.gxnu.edu.cn juntao.yu@qmul.ac.uk m.poesio@qmul.ac.uk

Abstract

Large Language Models (LLMs) have shown promising results in coreference resolution, especially after fine-tuning. However, recent generative approaches face a critical issue: hallucinations—where the model generates content not present in the original input. These hallucinations make evaluation difficult and decrease overall performance. To address this issue, we analyze the underlying causes of hallucinations and propose a low-hallucination and efficient solution. Specifically, we introduce Efficient Constrained Decoding for Coreference Resolution, which maintains strong robustness while significantly improving computational efficiency. On the English OntoNotes development set, our approach achieved slightly better performance than previous state-of-the-art methods, while requiring substantially fewer parameters [1].

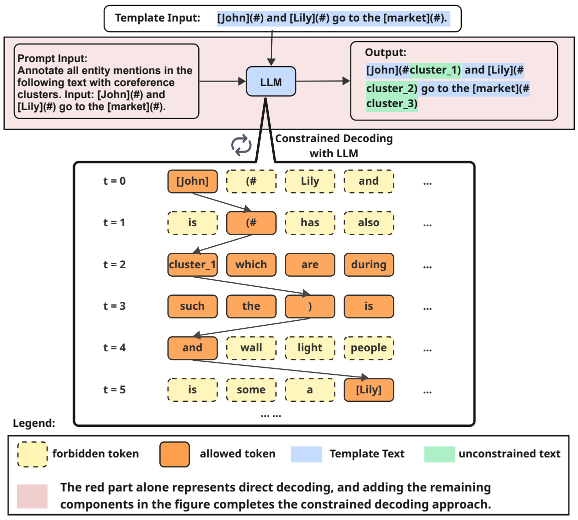

Figure 1: An example of constrained decoding. During decoding, the model generates content only at designated positions while copying the template text elsewhere.

1 Introduction

Coreference resolution (CR) is the task of detecting and grouping mentions that refer to the same entity in text. This task requires deep linguistic and contextual understanding, which makes it a vital component for many downstream NLP applications, including information extraction, text summarization, chatbots, and dialogue systems. As a result, CR has attracted significant attention from the NLP community (Poesio et al., 2023).

Most recently, the rise of Large Language Models (LLMs) has transformed NLP by providing strong contextual understanding and knowledge integration across tasks. Recent studies have demonstrated LLMs' effectiveness at coreference resolution, especially in zero- and few-shot scenarios (Yang et al., 2022; Agrawal et al., 2022; Le and Ritter, 2024; Zhu et al., 2024; Gan et al., 2024). Le and Ritter (2024) achieved superior performance among these approaches with their document template method. This method used Markdown-style formatting to highlight candidate mentions and places at the designated character positions as cluster ID placeholders. The model then generated the original document while inserting cluster IDs at these designated positions. These IDs served to indicate which mentions refer to the same entity, as illustrated by the red portion in Figure 1.

However, we observed that asking the model to reproduce the full text often introduced hallucinations—generating content not present in the input—making it not robust and difficult to evaluate the accuracy of predicted cluster IDs. For example, the LLM incorrectly generated "There are a candle a candle a wall ..." by adding the redundant mention "a candle". In such cases, aligning these mentions with the original document becomes significantly more challenging. Ideally, the output should be identical to the input except at the des-

[*]Corresponding author
[1]Our code is available here.

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 17243–17256
November 4-9, 2025 ©2025 Association for Computational Linguistics

ignated positions where cluster IDs are inserted. However, we find that it is non-trivial for LLMs to perform this task reliably, as hallucinations are widespread across models. Appendix C presents examples of hallucinations produced by different LLMs.

We attribute one of the main causes of hallucinations to LLMs' inability to strictly adhere to the original content in the prompt—particularly when processing long and structurally complex input texts. Building on this observation, our previous method restricted the model to output cluster IDs for one mention at a time rather than reproducing the full document (Gan et al., 2025). While this approach proved effective in mitigating hallucinations, its training efficiency was relatively low. Motivated by this limitation, we propose Efficient Constrained Decoding for Coreference Resolution, which preserves robustness while significantly improving computational efficiency. On the English OntoNotes development set, our method achieved slightly better performance than previous state-of-the-art methods, despite using substantially fewer parameters.

2 Related Work

Our work primarily builds on two studies by Le and Ritter (2024) and Zhang et al. (2023). In addition, we draw inspiration from the use of constrained decoding in named entity recognition (De Cao et al., 2021), and adapt this approach to the task of coreference resolution. Since both document-annotation-based coreference resolution and constrained decoding are central to our method, we discuss them in this section. Other related work is discussed in Appendix A.

2.1 Document Annotation-based Coreference Resolution

The traditional document annotation-based method framed coreference resolution as a generation task, where the input document was returned with mentions and cluster information explicitly annotated.

Zhang et al. (2023) proposed a seq2seq framework built on a prompt-fine-tuned T0 model. Their system took a full document as input and outputs the same document annotated with mentions and cluster information. This method achieved performance comparable to that of Bohnet et al. (2023).

Le and Ritter (2024) have introduced a prompt-based approach where LLMs were presented with documents containing highlighted mentions (either predicted or gold), each followed by a special cluster placeholder (#). The model was tasked with filling in these placeholders with the appropriate cluster IDs.

2.2 Constrained Decoding

Constrained Decoding is a technique used in language models (LLMs) to ensure that the generated output adheres to a specific constraint (Geng et al., 2023). This approach first appeared in the form of a grammar decoder, designed to ensure that the outputs generated by the decoder are syntactically correct (Kusner et al., 2017), such as guaranteeing the validity of SQL syntax (Gan et al., 2020). It is now particularly valuable for tasks requiring syntactic or semantic correctness, such as code generation and symbolic math reasoning (Banerjee et al., 2025).

Several approaches to constrained decoding exist, including constrained decoding with lookahead heuristics (CDLH) (Nakshatri et al., 2025), speculative lookaheads (Anderson et al., 2017), and grammar-aligned methods (Geng et al., 2023).

A key challenge in constrained decoding is striking the right balance between constraint satisfaction and LLM expressivity. While strict grammatical constraints can inhibit an LLM's ability to generate novel and creative responses (Banerjee et al., 2025), insufficient constraints may lead to syntactically or semantically incorrect output (Geng et al., 2023). Researchers have developed adaptive constrained decoding techniques to address this challenge. For instance, Yuan et al. (2024) introduced Contextual Information-Entropy Constrained Decoding (COIECD), an adaptive method that detected and handled knowledge conflicts during generation.

Constrained decoding remains a crucial technique for ensuring LLMs generate syntactically and semantically correct output (Geng et al., 2023). Through careful balancing of constraint satisfaction and expressivity, researchers continue to develop more effective and efficient algorithms applicable across various domains.

3 Hallucination Analysis and Causal Hypotheses

We argue that one of the primary causes of hallucinations in coreference resolution is LLMs' inability to strictly follow the original content in the prompt, particularly with long and structurally complex input text. This complexity often stems from missing punctuation, ambiguous pronouns, or omitted contextual information (e.g., dialogues lacking background) —factors that significantly increase the risk of hallucination during generation.

One possible underlying reason is attention dilution in Transformer-based models (Qin et al., 2022). As input length grows, the attention weight for each token in the softmax distribution decreases. This dilution makes it harder for the model to focus on crucial input elements, causing it to rely more heavily on pre-trained language patterns and potentially deviate from the source text. Moreover, Liu et al. (2024) have demonstrated that LLMs tend to focus disproportionately on the beginning and end of long inputs while neglecting important middle content, further increasing the hallucination risk.

Based on the above analysis, we believe that preventing hallucinations in coreference resolution requires constraining the model from generating incorrect tokens when processing long and complex input texts.

4 Hypothesis-Driven Methods

To test our hypothesis, we use two methods for preventing incorrect token generation: Iterative Document Generation (our previous work) and Constrained Decoding for Coreference Resolution.

4.1 Iterative Document Generation

The core idea of Iterative Document Generation is to avoid direct full-text generation by the model. Instead, the original one-shot generation task is decomposed into multiple steps, where the model generates a cluster ID for one mention at a time. The process works step by step: we input a short text snippet ending with the mention to be predicted and prompt the LLM to output only the cluster ID of that mention. We then combine the original input with the predicted cluster ID and continue appending the remaining content up to the next mention. This updated text is used to form a new prompt, which is fed to the LLMs to predict the cluster ID for the next mention. Through these iterative steps, we obtain cluster IDs for all mentions across the document without requiring the LLMs to generate the original text. For further implementation details and illustrations, please refer to Appendix B.1, which provides a reformulated description based on our previous work (Gan et al., 2025).

4.2 Efficient Constrained Decoding for Coreference Resolution

Constrained Decoding. Unlike Iterative Document Generation, which completely avoids generating original text, Constrained Decoding prevents incorrect text fragments through guided generation. The model uses both a structured prompt—similar to that used by Le and Ritter (2024)—and a template that constrains the output sequence. This template ensures certain parts of the model's output match the prompt exactly.

As shown in Figure 1, the LLM strictly follows the template when generating initial tokens. When it emits a special marker ('**(#**'), it enters an unconstrained free-form generation phase. This continues until another marker ('**)**') appears, after which constraint-guided decoding resumes. The process repeats until the model produces complete and correct text with corresponding cluster IDs.

Grammatical Framework. Previous applications of Constrained Decoding primarily focused on code generation tasks like SQL, where well-defined grammars ensured syntactic correctness. However, in CR, there is no universally accepted grammar comparable to programming languages. In our framework, we treat the original text as a task-specific 'grammar' or template. Because of natural language's inherent variability, this template must be explicitly provided to the LLM during each decoding step to guide generation.

A significant challenge during decoding relates to tokenization. Common tokenization schemes like byte pair encoding (BPE) (Sennrich et al., 2016) allow multiple tokenization possibilities for a single string. For example, the template marker '**(#)**' may be tokenized as '(#)', '(#)', or '(#)', which are all detokenized to the same string. This variability creates ambiguity in constrained decoding. To address this challenge, we replaced commonly ambiguous tokens with special tokens.

Efficient Constrained Decoding. In contrast to tasks like SQL generation that produce sparse grammar tokens, our coreference resolution setting requires generating text that largely replicates the input, with minimal insertions like cluster IDs (see Figure 1). Leveraging this characteristic, we introduce a fast decoding strategy that moves beyond token-by-token generation. For grammar regions defined by the template, we perform multi-token decoding in a single step. For instance, in Fig-

System	CoNLL F_1
SpanBERT+e2e (Joshi et al., 2020a)	91.1
Llama2 (7B) (Le and Ritter, 2024)	91.2
Llama2 (13B) (Le and Ritter, 2024)	92.8
Llama2 (70B) (Le and Ritter, 2024)	93.6
coref-T0 (11B) (Zhang et al., 2023)	94.8
LLAMA3 (8B) $\mathbb{I}$ (**Ours**)	94.7
LLAMA3 (8B) $\mathbb{E}$ (**Ours**)	**95.1**

Table 1: Coreference resolution performance on the English OntoNotes dev set.

ure 1, tokens at time steps $t = 0$ and $t = 1$ can be generated simultaneously during the first decoding step, eliminating repeated GPU invocations. This approach improves decoding efficiency without sacrificing accuracy, as the grammar tokens are deterministically constrained by the template. See Appendix B.2 for details and examples.

5 Experiments

5.1 Experimental Setup

Datasets. We used the same datasets as Le and Ritter (2024): English OntoNotes 5.0 (Pradhan et al., 2012) and CRAC (Poesio et al., 2018; Yu et al., 2022). We utilized three subsets of CRAC: LIGHT, Penn Treebank, and TRAINS. Consistent with the experimental setup of previous work, we used gold mentions.

Models. To validate the feasibility of our method, we fine-tuned different LLMs as base models: Llama 2, Llama 3.1, Llama 3.2 (Grattafiori et al., 2024) and Qwen 2.5 (Qwen et al., 2025). The symbols associated with the models are defined as follows:

- **LLAMA3 (8B)** indicates Llama 3.1 8B Instruct.
- **LLAMA3 (1B)** indicates Llama 3.2 1B Instruct.
- **LLAMA2 (7B)** indicates Llama 2 7B.
- **QWEN (7B)** indicates Qwen 2.5 7B Instruct.
- $\mathbb{D}$: full-document training and direct decoding.
- $\mathbb{E}$: indicates full-document training and efficient constrained decoding.
- $\mathbb{I}$: indicates iterative document generation for both training and inference.

For details of implementation and metrics, please refer to Appendix D.1.

5.2 Overall Performance

As shown in Table 1, our proposed models based on Llama 3.1 8B achieved state-of-the-art results on the English OntoNotes 5.0 development set. Specifically, the $\mathbb{E}$ variant of our model reached an F_1 score of **95.1**, slightly surpassing previous best-performing systems such as coref-T0 (94.8) and significantly outperforming other LLaMA2 baselines with comparable or larger parameter sizes. Notably, our best model attained this performance using only 8B parameters, whereas prior top-performing models typically required larger architectures (e.g., LLaMA2 70B or coref-T0 11B). While our current results establish a new performance benchmark, we anticipate further gains with larger model scales. However, due to computational limitations, we did not explore scaling beyond the 8B parameter range. It is worth noting that the LLAMA3 (8B) $\mathbb{D}$ variant encountered several unalignable examples during inference on the development set, preventing us from computing an exact CoNLL F_1 score. Therefore, we omit direct comparisons involving LLAMA3 $\mathbb{D}$ in this section. For a more comprehensive comparison including this variant, please refer to the next section, which presents evaluations on the test set.

5.3 Hallucination Analysis

Table 2 presents CoNLL F_1 scores and the percentage of unaligned samples for each approach. We fine-tuned all models on OntoNotes and tested them on both OntoNotes and CRACs datasets to evaluate their cross-domain robustness. Using the alignment check ("Pass") from Le and Ritter (2024), we measured hallucination rates by identifying unaligned outputs as instances of severe hallucination. We also conducted exact match ("EM") evaluation, which directly measured the proportion of the output faithfully reproduced by the model from the original input. See Appendix C for hallucination examples, and Appendix D.2 for results fine-tuned on CRACs.

Both $\mathbb{I}$ and $\mathbb{E}$ methods achieved 100% rates in both Pass and EM alignment checks, suggesting that they effectively eliminated hallucinations resulted from textual alignment. Manual inspection further confirmed that there were no instances of such hallucinations. However, in the TRAINS subset, hallucinations persisted with $\mathbb{D}$ despite its perfect pass rate. The notably low EM score in this case indicates that the alignment check may overestimate reliability, as we observed clear cases of hallucinated content not flagged. This discrepancy stems from the overly lenient nature of the check proposed by Le and Ritter (2024), which assessed only the number of predicted clusters without considering their semantic correctness. As a result,

	LIGHT			Penn Treebank			TRAINS			OntoNotes		
Approach	**CoNLL F_1**	**Pass**	**EM**	**CoNLL F_1**	**Pass**	**EM**	**CoNLL F_1**	**Pass**	**EM**	**CoNLL F_1**	**Pass**	**EM**
LLAMA2 (7B) $\mathbb{D}$	86.7	76.32%	50%	81.7	85.00%	65%	42.5	100%	43.75%	93.6	98.28%	93.97%
LLAMA2 (7B) $\mathbb{I}$	87.1	100%	100%	84.1	100%	100%	75.4	100%	100%	94.2	100%	100%
LLAMA2 (7B) $\mathbb{E}$	87.2	100%	100%	80.8	100%	100%	68.1	100%	100%	93.5	100%	100%
LLAMA3 (1B) $\mathbb{D}$	84.6	94.74%	78.95%	77.7	93.33%	70%	54.0	100%	50.0%	91.6	98.56%	94.54%
LLAMA3 (1B) $\mathbb{I}$	85.7	100%	100%	82.8	100%	100%	72.3	100%	100%	92.1	100%	100%
LLAMA3 (1B) $\mathbb{E}$	84.5	100%	100%	80.0	100%	100%	75.9	100%	100%	91.9	100%	100%
LLAMA3 (8B) $\mathbb{D}$	86.6	92.11%	81.58%	82.5	100%	70%	73.4	100%	62.50%	95.3	99.71%	97.41%
LLAMA3 (8B) $\mathbb{I}$	87.8	100%	100%	85.2	100%	100%	76.4	100%	100%	94.9	100%	100%
LLAMA3 (8B) $\mathbb{E}$	86.5	100%	100%	83.3	100%	100%	78.0	100%	100%	95.4	100%	100%
QWEN (7B) $\mathbb{D}$	87.0	97.37%	92.11%	83.2	98.33%	71.67%	77.2	100%	81.25%	95.0	99.14%	97.13%
QWEN (7B) $\mathbb{E}$	85.7	100%	100%	80.6	100%	100%	79.6	100%	100%	94.8	100%	100%

Table 2: Test set results comparing different approaches fine-tuned on the OntoNotes training set. “Pass” denotes the proportion of documents that passed the alignment check according to Le and Ritter (2024). “EM” (Exact Match) denotes the proportion of the output that exactly matches the original text.

System	Training	Inference
LLAMA2(7B) $\mathbb{D}$	1:35:04 (**100%**)	0:37:59 (338%)
LLAMA2(7B) $\mathbb{I}$	50:36:17 (3194%)	0:22:40 (202%)
LLAMA2(7B) $\mathbb{E}$	1:35:24 (**100%**)	0:11:14 (**100%**)

Table 3: Comparison of runtime for different methods in the same environment and on the same dataset.

invalid outputs could still be accepted, giving a misleadingly high pass rate. Consequently, $\mathbb{I}$ and $\mathbb{E}$ showed better overall performance than $\mathbb{D}$ on TRAINS. Since hallucinated examples in $\mathbb{D}$ were not filtered out, they were included in the F_1 computation, ultimately dragging down the final score.

Besides, $\mathbb{D}$ models—particularly smaller or older variants such as LLaMA3 1B and LLaMA2—produced more hallucinated outputs. However, our $\mathbb{I}$ and $\mathbb{E}$ methods remained robust in all domains. These results support the hypotheses presented in Section 3. The F_1 score, being calculated only on aligned samples, may overestimate the performance of weaker models by excluding difficult cases. Nevertheless, the overall F_1 scores across $\mathbb{D}$, $\mathbb{I}$, and $\mathbb{E}$ remain comparable.

5.4 Computational Efficiency

Table 3 compares runtime performance across methods. Other models exhibited similar trends, so we omit their results for brevity. $\mathbb{E}$ demonstrated superior efficiency, achieving the least inference time while maintaining training costs comparable to $\mathbb{D}$. In contrast, $\mathbb{I}$ was substantially more resource-intensive—requiring 30× longer for training and 2× longer for inference—due to its iterative approach. While $\mathbb{D}$ matched $\mathbb{E}$ in training time, it performed 3× slower during inference and exhibited hallucination problems, making it less practical than $\mathbb{E}$.

6 Conclusion

Hallucinations are a key challenge for LLM-based coreference resolution, arising from models’ inability to faithfully follow the input text. To verify this hypothesis, we revisited Iterative Document Generation from our previous work, which demonstrated the benefit of constraining the output space. Building on this insight, this paper introduces Efficient Constrained Decoding for coreference resolution, which not only eliminated hallucinations but also achieved substantially better efficiency, surpassing previous state-of-the-art results on OntoNotes while using fewer parameters.

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, ARCIDUCA. Yuan Liang’s work was supported by the UKRI Biotechnology and Biology Sciences Research Council [BB/X511833/1], Digital Environment and Research Institute (DERI), the Queen Mary University of London, and Recursion Pharmaceuticals Inc. Yanni Lin acknowledges the financial support from the Office of Philosophy and Social Sciences of Guangxi Zhuang Autonomous Region [22CYY005], and Department of Education of Guangxi Zhuang Autonomous Region [2022KY0040]. We also thank Apocrita (King et al., 2017) and its ITS team for providing and maintaining the HPC resources.

Ethical Considerations

All datasets for our experiments are publicly available and fully cited. We acknowledge that the cited datasets may contain biases and errors, and the systems could reflect these issues. This work is for research use only. AI writing assistants were used to aid in editing and proofreading this manuscript.

Limitations

Due to limited computational resources, we did not evaluate our methods on larger-scale language models. However, given the consistent results across different model families and parameter sizes, we believe our findings are expected to generalize across scales. In theory, our approach may still produce hallucinations in rare cases—for example, when the model fails to correctly generate a cluster ID. Such failures are more likely to occur with extremely small models. However, models of such scales typically fall beyond the scope of discussion on LLMs, thus not deemed as the focus of this study.

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.

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2017. Guided open vocabulary image captioning with constrained beam search. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 936–945, Copenhagen, Denmark. 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.

Debangshu Banerjee, Tarun Suresh, Shubham Ugare, Sasa Misailovic, and Gagandeep Singh. 2025. Crane: Reasoning with constrained llm generation. *Preprint*, arXiv:2502.09061.

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.

Anthony Chen, Panupong Pasupat, Sameer Singh, Hongrae Lee, and Kelvin Guu. 2023. Purr: Efficiently editing language model hallucinations by denoising language model corruptions. *Preprint*, arXiv:2305.14908.

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.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021. Autoregressive entity retrieval. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2024. Chain-of-verification reduces hallucination in large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3563–3578, Bangkok, Thailand. Association for Computational Linguistics.

Yujian Gan, Yuan Liang, Yanni Lin, Juntao Yu, and Massimo Poesio. 2025. Improving llms' learning for coreference resolution. *Preprint*, arXiv:2509.11466.

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.

Yujian Gan, Matthew Purver, and John R. Woodward. 2020. A review of cross-domain text-to-SQL models. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: Student Research Workshop*, pages 108–115, Suzhou, China. Association for Computational Linguistics.

Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. 2023. Grammar-constrained decoding for structured NLP tasks without finetuning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10932–10952, Singapore. Association for Computational Linguistics.

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.

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. 2020a. Spanbert: Improving pre-training by representing and predicting spans. *Preprint*, arXiv:1907.10529.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020b. 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.

Thomas King, Simon Butcher, and Lukasz Zalewski. 2017. *Apocrita - High Performance Computing Cluster for Queen Mary University of London*.

Matt J. Kusner, Brooks Paige, and José Miguel Hernández-Lobato. 2017. Grammar variational autoencoder. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1945–1954. PMLR.

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.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Potsawee Manakul, Adian Liusie, and Mark Gales. 2023. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9004–9017, Singapore. Association for Computational Linguistics.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12076–12100, Singapore. Association for Computational Linguistics.

Nishanth Sridhar Nakshatri, Shamik Roy, Rajarshi Das, Suthee Chaidaroon, Leonid Boytsov, and Rashmi Gangadharaiah. 2025. Constrained decoding with speculative lookaheads. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4681–4700, Albuquerque, New Mexico. Association for Computational Linguistics.

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, 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, 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.

Zhen Qin, Weixuan Sun, Hui Deng, Dongxu Li, Yunshen Wei, Baohong Lv, Junjie Yan, Lingpeng Kong, and Yiran Zhong. 2022. cosformer: Rethinking softmax in attention. In *International Conference on Learning Representations*.

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.

Vipula Rawte, S.m Towhidul Islam Tonmoy, Shravani Nag, Aman Chadha, Amit Sheth, and Amitava Das. 2025. FACTOID: FACtual enTailment fOr hallucInation detection. In *Proceedings of the 5th Workshop on Trustworthy NLP (TrustNLP 2025)*, pages 599–617, Albuquerque, New Mexico. Association for Computational Linguistics.

Pranab Sahoo, Prabhash Meharia, Akash Ghosh, Sriparna Saha, Vinija Jain, and Aman Chadha. 2024. A comprehensive survey of hallucination in large language, image, video and audio foundation models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 11709–11724, Miami, Florida, USA. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Neeraj Varshney, Wenlin Yao, Hongming Zhang, Jianshu Chen, and Dong Yu. 2023. A stitch in time saves nine: Detecting and mitigating hallucinations of llms by validating low-confidence generation. *Preprint*, arXiv:2307.03987.

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.

Ziwei Xu, Sanjay Jain, and Mohan Kankanhalli. 2025. Hallucination is inevitable: An innate limitation of large language models. *Preprint*, arXiv:2401.11817.

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.

Xiaowei Yuan, Zhao Yang, Yequan Wang, Shengping Liu, Jun Zhao, and Kang Liu. 2024. Discerning and resolving knowledge conflicts through adaptive decoding with contextual information-entropy constraint. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3903–3922, Bangkok, Thailand. Association for Computational Linguistics.

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.

A Detailed Related Work

A.1 Coreference Resolution

A.1.1 Traditional Coreference Resolution Systems

Traditional coreference resolution systems have largely relied on mention-ranking models. Early approaches, such as those by Wiseman et al. (2015) and Clark and Manning (2016), followed a two-step pipeline: first detecting potential mentions in the text, and then clustering them into coreference groups using a separate model. A significant advancement came with the work of Lee et al. (2017), who introduced an end-to-end neural model that jointly performed mention detection and clustering. This elegant architecture has become a standard in the field due to its effectiveness and conceptual simplicity.

Subsequent research built on this foundation by incorporating richer contextual representations and addressing more nuanced challenges. For instance, Lee et al. (2018) and Kantor and Globerson (2019) enhanced performance by leveraging contextual embeddings. Yu et al. (2020) tackled the problem of singletons and non-referring expressions, which had previously been overlooked. Later, Joshi et al. (2019, 2020b) improved the model's representational power by replacing LSTMs with pre-trained transformer models such as BERT and SpanBERT.

A.1.2 Coreference Resolution as Question Answering

Recent work has explored reframing coreference resolution as a question-answering (QA) task, enabling models to approach the problem through more interpretable and flexible interfaces.

Wu et al. (2020) proposed CorefQA, which reformulated coreference resolution by treating each mention as a question and identifying all other mentions that belonged to the same cluster as answers. Using SpanBERT pre-trained on datasets like Quoref and SQuAD 2.0, they employed a BIO tagging scheme for span prediction. A key insight from their work was the importance of bidirectional scoring between mention pairs (i.e., $S_{i,j}$ and $S_{j,i}$).

This QA formulation has been extended by other research in various ways. Aralikatte et al. (2021) addressed ellipsis resolution using a BERT-based machine reading comprehension (MRC) model that identified antecedents for incomplete mentions. Yang et al. (2022) evaluated early GPT models on the ECB+ dataset, using few-shot QA templates to classify mention pairs as coreferent or not, though results were limited. Bohnet et al. (2023) achieved state-of-the-art performance by fine-tuning the mT5-XXL (13B) model to generate both candidate mentions and their associated clusters.

Le and Ritter (2024) have demonstrated that prompt-based, instruction-tuned language models performed coreference resolution effectively using a QA-style template. In this setup, models were given open-ended *wh*-questions to extract coreferent entities from passages. Building on this idea, Gan et al. (2024) have conducted a comprehensive evaluation of LLMs' ability to identify antecedents and corresponding sentence IDs. Zhu et al. (2024) also adopted a prompt-based setting but employed a multiple-choice formulation to evaluate LLMs' decisions over candidate coreference links.

A.2 Hallucination in LLMs

Hallucination in Large Language Models (LLMs) has emerged as a critical challenge, particularly as these models see increasing deployment in high-stakes domains. This phenomenon occurs when LLMs generate fluent but factually incorrect, unsupported, or misleading content. Recent surveys Sahoo et al. (2024) have categorized LLM hallucinations into distinct types, including factual inaccuracies, semantic distortions, and contextual disconnections, revealing their varied manifestations and effects.

To combat this issue, researchers have developed numerous detection and mitigation methods. Detection approaches include probabilistic self-evaluation (e.g., SelfCheckGPT; Manakul et al., 2023), automatic evaluation (e.g., FACTOID; Rawte et al., 2025), and evidence retrieval (e.g., FACTSCORE; Min et al., 2023). These methods can identify hallucinated content without always needing external knowledge bases. For mitigation, promising strategies include fine-tuning with factual supervision (e.g., PURR; Chen et al., 2023), chain-of-thought and reasoning enhancement (e.g., CoVe; Dhuliawala et al., 2024), and prompt engineering and instructional techniques (e.g., Instructional Prompting; Varshney et al., 2023).

Despite these advances, hallucination remains stubbornly persistent. Xu et al. (2025) have suggested that hallucination might be inherent to LLMs' generative mechanisms and offered a formal framework for understanding its origins relative to ground truth. This insight has driven research into proactive solutions like retrieval-augmented generation, instruction tuning, and reinforcement learning with factual constraints.

In summary, research on LLM hallucinations represents a dynamic field that seeks both to enhance model reliability and to grasp fundamental limitations from theoretical and practical standpoints. These findings are crucial for ensuring LLMs can be deployed safely and effectively in real-world applications.

B Further Details of the Method

B.1 Iterative Document Generation

We illustrate the *Iterative Document Generation* framework in Figure 3. Both sub-figures depict the same core process: rather than generating full text, the model incrementally predicts *cluster IDs* for each entity mention in a document.

Iterative Process. At the heart of this approach is a step-by-step mention-level resolution process. Instead of prompting the model to generate the document all at once, we proceed as follows:

- At each iteration, we provide the model with a short text segment ending in the target mention.

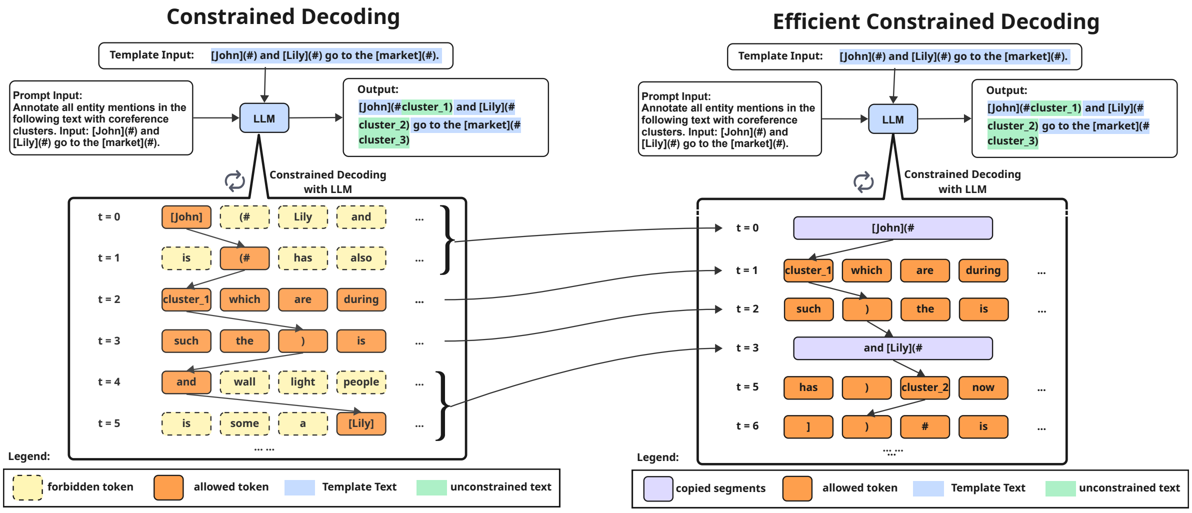

Figure 2: An example of efficient constrained decoding. During decoding, the model copies entire segments of the original text at once, rather than processing individual words.

- The model is prompted to generate **only** the cluster ID corresponding to this mention.
- The predicted cluster ID is appended after the mention in the context.
- We then extend the prompt with additional text, continuing until the next mention appears.

This process continues sequentially until all mentions in the document have been assigned cluster IDs. By restricting generation to structured identifiers rather than open-ended language, the method significantly reduces hallucinations and enhances control over the output format.

Training vs. Inference. A key distinction in this framework lies in how model inputs differ between training and inference:

- **During training**, the model receives gold-standard cluster IDs. All previous cluster IDs in the prompt are correct, allowing the model to learn from error-free context.
- **During inference**, gold labels are unavailable. If the model makes an incorrect prediction at any step, that incorrect cluster ID is still injected into the prompt. Subsequent predictions must be made based on potentially noisy and incorrect prior context.

This discrepancy may introduce error propagation during inference. However, it mirrors real-world usage, where ground truth annotations are not available at test time.

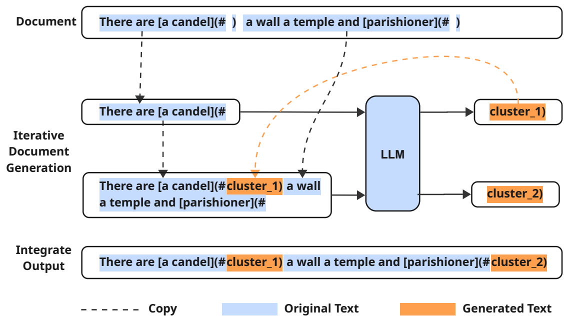

Figure 3: An example of the iterative document generation. Only the cluster ID is generated, while the original text is reused.

Limitations and Challenges. While the iterative document generation framework offers improved robustness and reduced hallucination rates, the approach is computationally inefficient. Since the model processes one mention at a time, the number of required forward passes scales linearly with the number of mentions in a document. This repeated invocation of the model significantly increases both training and inference time, making the method less practical for large-scale applications.

B.2 Constrained Decoding for Coreference Resolution

Figure 2 illustrates the transition from standard constrained decoding to our proposed efficient variant. As shown in the figure, efficient constrained decoding allows the model to generate all constrained-specified tokens in a single step. For example, decoding that originally completed at time step $t = 6$

under standard decoding can now be completed as early as $t = 3$. This results in a substantial reduction in decoding steps. Moreover, the efficiency gain becomes more pronounced as the number of constrained tokens increases. The detailed algorithm is provided in Algorithm 1.

As discussed in the main text, using predefined markers such as '(#)' can introduce complications during BPE tokenization. This is particularly problematic when the original text contains similar symbols, increasing the risk of tokenization mismatches and decoding errors. To address this, we experimented with replacing such ambiguous markers with special tokens. As different LLMs support different special tokens, we take LlamA 3 as an example: the opening marker '(#' is replaced with `<|reserved_special_token_87|>` and the closing marker ')' with `<|reserved_special_token_88|>`. However, this approach has a key limitation. Although these special tokens are reserved by the model, LlamA 3 does not allow them to be generated during decoding. As such, this technique is not applicable under direct decoding settings.

Under constrained decoding, however, this limitation is less severe. Even if the special tokens cannot be output directly by the model, the decoding constraints can still enforce their correct placement. Empirically, we observed that using special tokens yielded similar performance to using the original marker '(#)', confirming the feasibility of this strategy. Nonetheless, due to the lack of output support and inconsistent special token definitions across different LLMs, we did not adopt this approach in our main experiments. Instead, we validated its viability solely under Llama 3. Notably, when the model was forced to generate such special tokens, it often substituted them with a fixed token. In Llama 3, this corresponds to token index 124, which can be detokenized to the character 'À'.

Importantly, constrained decoding was only applied at inference time. The training procedure remains identical to that of direct decoding. We also experimented with modifying the training process by masking the loss for grammar-constrained portions, that is, setting their labels to -100 so that the loss was computed only on cluster ID spans. However, across multiple models, we observed no significant performance improvements compared to standard training. Therefore, all experiments in this paper used the same training set up as direct decoding, with constrained decoding applied only during inference.

Generating meaningless gibberish or labels

Output:

[cat]\ufffdcluster_18\ufffd said: To [me]\ufffdcluster_18\ufffd, [I]\ufffdcluster_18\ufffd m just a simple cat.
[a shape - shifting cat]\ufffdcluster_20\ufffd said: A simple cat [you]\ufffdcluster_18\ufffd may be, but [that]\ufffdcluster_25\ufffd is the form [I]\ufffdcluster_20\ufffd enjoy.

Analysis:

In this example, Llama3 generated meaningless garbled characters "\ufffd" in the output.

Figure 4: Example of generating meaningless gibberish or labels

Producing repetitive expressions or labels

Output:

[it](#cluster_56) is [the wardrobe](#cluster_56) [it](#cluster_56) is [the wardrobe](#cluster_56) [it](#cluster_56) is [the wardrobe](#cluster_56)

Analysis:

In this example, Llama2 repeated the same phrase along with its cluster tag "[it](#cluster_56) is [the wardrobe](#cluster_56)".

Figure 5: Example of producing repetitive expressions or labels

C Observed Hallucination

The cases where LLM hallucinations occur typically fall into four main categories: (1) generating meaningless gibberish or labels which do not exist in the original text; (2) copying unnecessary textual expressions or labels repetitively; (3) failing to detect a mention, or missing the coreference link; (4) incorrectly identifying a mention or its boundaries. We provide examples to illustrate these categories, which can be seen in Figure 4, 5, 6, and 7.

D Supplementary Experimental Content

D.1 Additional Experimental Setup

Implementation. All the results in this paper were based on a single run. 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 mul-

Algorithm 1 Efficient Constrained Decoding

```
Input: input_ids, template_ids, model, tokenizer
Output: input_ids                                                    // Generated token sequence
Initialize:
  followup ← True,   followup_slow ← True,   offset ← 0
  last_token ← '',   end_token ← 0,   end_token_str ← ''

while has_unfinished_sequences(cur_len, max_length) do
    model_inputs ← prepare_model_inputs(input_ids)
    if not followup then                       // Normal decoding mode using sampling or argmax
        outputs ← model_forward_pass(model, model_inputs)
        next_token_scores ← process_logits(outputs)
        if do_sample then
            next_tokens ← sample_from_probs(next_token_scores)
        else
            next_tokens ← argmax(next_token_scores)
        end if
    else                                                         // Constrained decoding mode
        current_template_token ← template_ids[0][offset]
        next_tokens ← force_token(current_template_token)
        current_token_str ← decode_token(current_template_token)
        if is_start_token(current_token_str, special_tokens) then
            handle_start_token_case()
            followup ← False                                  // Change to normal generation mode
            update_end_token()
            adjust_offset()
        else if is_middle_token(current_token_str, last_token, special_tokens) then
            followup ← False
            update_end_token()
            offset ← offset + 1                   // Update the current template position marker
        else if is_end_token(current_token_str, end_token_str, special_tokens) then
            handle_end_token_case()
            followup ← True                                     // Keep constrained decoding mode
            offset ← offset + 1
            last_token ← current_token_str
        end if
    end if
    input_ids ← append_token(input_ids, next_tokens)             // Update the generated sequence
    if not followup_slow then                                     // Efficient constrained decoding
        model_kwargs ← update_model_kwargs(outputs)
        clear_outputs()
    else                                                             // Normal constrained decoding
        update_attention_mask()
        update_cache_position()
        followup_slow ← followup
    end if
    cur_len ← cur_len + 1
    offset ← offset + 1
    if offset ≥ len(template_ids[0]) then
        break
    end if
end while
Return input_ids =0
```

Failing to detect a mention or a coreference link

Output:

cat said: [To](#cluster_26) me, [I m just](#cluster_27) a [simple cat.](#cluster_28)

Analysis:

In this example, the mention "cat" in the original text was missed by Llama2.

Figure 6: Example of failing to detect a mention or a coreference link

Incorrectly identifying a mention or its boundaries

Output:

cat said: [To](#cluster_25) me, [I m just](#cluster_15) a [simple cat.](#cluster_26)
[a shape-shifting cat said:](#cluster_17) A simple cat you [may](#cluster_27) be [, but
[that](#cluster_28) is](#cluster_29) the [form I](#cluster_15) enjoy [.

Analysis:

In this example, some mentions and their boundaries were incorrectly identified. Expressions beyond noun phrases, such as "to" and "may", were wrongly recognized as mentions. Additionally, boundary detection involving punctuation marks (e.g. "[," and "[.") was inaccurate.

Figure 7: Example of incorrectly identifying a mentions or its boundaries

tiple A100 GPUs (40GB/80GB). For multi-GPU setups, DeepSpeed was used to optimize parallel training.

Metrics. We directly used the CoNLL F1 evaluation script provided by Le and Ritter (2024). This script retains singleton mentions during evaluation. For corpora that did not include singleton mentions (e.g., OntoNotes), this has minimal impact, as singletons only appeared in the system-generated clusters. However, for corpora that contained singleton mentions (e.g., CRAC), this difference in setting can lead to changes of over 10% in CoNLL F1 scores. To ensure a fair comparison, we followed

	LIGHT		Penn Treebank		TRAINS	
Approach	**CoNLL F_1**	**Pass**	**CoNLL F_1**	**Pass**	**CoNLL F_1**	**Pass**
LLAMA3 (1B) $\mathbb{D}$	87.8	84.21%	86.7	95.0%	61.8	100%
LLAMA3 (1B) $\mathbb{E}$	88.5	100%	87.8	100%	82.9	100%
LLAMA3 (8B) $\mathbb{D}$	91.4	97.37%	90.3	95.0%	83.8	100%
LLAMA3 (8B) $\mathbb{E}$	92.0	100%	92.0	100%	87.1	100%
QWEN (7B) $\mathbb{D}$	92.1	94.74%	90.1	96.67%	82.7	100%
QWEN (7B) $\mathbb{E}$	91.6	100%	93.4	100%	86.3	100%

Table 4: Comparison of different approaches fine-tuned on the CRACs training set. "Pass" denotes the proportion of documents that pass the alignment check according to Le and Ritter (2024).

the same evaluation settings when comparing with their system.

D.2 Additional Experimental Results

In the main body of this paper, we presented results based on models fine-tuned on the OntoNotes dataset. To further assess the generalizability of our approach, we also conducted experiments using models fine-tuned on the CRACs training set. Table 4 summarizes the performance of various models across three benchmark test sets: LIGHT, Penn Treebank, and TRAINS.

The experimental conclusions are consistent with those in the main text: our $\mathbb{E}$ method continued to demonstrate robust performance across all evaluation settings. Notably, when both training and evaluation were conducted on CRACs, the performance gains of the $\mathbb{E}$ approach were even more pronounced than those reported in the main body.

E Prompts

E.1 Prompt for Iterative Document Generation

We provide the prompt for Iterative Document Generation along with an example that pairs an input document template with its output, which can be seen in Figure 8.

E.2 Prompt for Constrained Decoding for Coreference Resolution

We also provide the prompt for Constrained Decoding for coreference resolution along with an example that pairs an input document template with its output, which can be seen in Figure 9.

Prompt for Iterative Document Generation

Prompt:

Instruction:
Annotate the last mention in the based on the original text with coreference clusters. Use Markdown tags to indicate clusters in the output, with the following format [mention](\#cluster_name)

Input:
a baptismal an offering plate a pew a baptismal font a candle a stained glass a bible a paper a Cross parishioner petitioner parishioner said: My crops are withering without rain, what sins have I committed for God to have forsaken me ? petitioner said: I would ask the same, for I am sick and it is most likely that I will die soon. parishioner said: What sickness befalls one such as yourself ? parishioner said: I have heard that the water of Saint Marthas will heal all who drink of it. petitioner said: The doctors have been unable to come to a real conclusion as to what it is, that is why I came to seek prayers.

Annotate the last mention:[a baptismal](\#

LLM:

Output:
cluster_0)

Figure 8: Prompt for Iterative Document Generation

Prompt for Constrained Decoding

Prompt:

Instruction:
Annotate all entity mentions in the following text with coreference clusters. Use Markdown tags to indicate clusters in the output, with the following format [mention](\#cluster_name)

Input:
[a baptismal](#) [an [offering](#) plate](#) [a pew](#) [a [baptismal](#) font](#) [a candle](#) [a stained glass](#) [a bible](#) [a paper](#) [a Cross](#) [parishioner](#) [petitioner](#) [parishioner](#) said: [[My](#) crops](#) are withering without [rain](#), [what sins](#) have [I](#) committed for [God](#) to have forsaken [me](#) ? [petitioner](#) said: [I](#) would ask [the same](#), for [I](#) am sick and it is most likely that [I](#) will die soon.

LLM:

Output:
[a baptismal](#cluster_0) [an [offering](#cluster_1) plate](#cluster_2) [a pew](#cluster_3) [a [baptismal](#cluster_4) font](#cluster_5) [a candle](#cluster_6) [a stained glass](#cluster_7) [a bible](#cluster_8) [a paper](#cluster_9) [a Cross](#cluster_10) [parishioner](#cluster_11) [petitioner](#cluster_12) [parishioner](#cluster_11) said: [[My](#cluster_11) crops](#cluster_13) are withering without [rain](#cluster_14), [what sins](#cluster_15) have [I](#cluster_11) committed for [God](#cluster_16) to have forsaken [me](#cluster_11) ? [petitioner](#cluster_12) said: [I](#cluster_12) would ask [the same](#cluster_17), for [I](#cluster_12) am sick and it is most likely that [I](#cluster_12) will die soon.

Figure 9: Prompt for Constrained Decoding for Coreference Resolution