

HCRE: LLM-based Hierarchical Classification for Cross-Document Relation Extraction with a Prediction-then-Verification Strategy

Guoqi Ma^{1,2*} Liang Zhang^{1*} Hongyao Tu¹ Hao Fu³ Hui Li¹ Yujie Lin¹

Longyue Wang⁴ Weihua Luo⁴ Jinsong Su^{1,2†}

¹School of Informatics, Xiamen University

²National Institute for Data Science in Health and Medicine, Xiamen University

³Li Auto Inc.

⁴Alibaba International Digital Commerce Group

{guoqima,lzhang}@stu.xmu.edu.cn, jssu@xmu.edu.cn

Abstract

Cross-document relation extraction (RE) aims to identify relations between the head and tail entities located in different documents. Existing approaches typically adopt the paradigm of “*Small Language Model (SLM) + Classifier*”. However, the limited language understanding ability of SLMs hinders further improvement of their performance. In this paper, we conduct a preliminary study to explore the performance of Large Language Models (LLMs) in cross-document RE. Despite their extensive parameters, our findings indicate that LLMs do not consistently surpass existing SLMs. Further analysis suggests that the underperformance is largely attributed to the challenges posed by the numerous predefined relations. To overcome this issue, we propose an LLM-based Hierarchical Classification model for cross-document RE (HCRE), which consists of two core components: 1) an LLM for relation prediction and 2) a *hierarchical relation tree* derived from the predefined relation set. This tree enables the LLM to perform hierarchical classification, where the target relation is inferred level by level. Since the number of child nodes is much smaller than the size of the entire predefined relation set, the hierarchical relation tree significantly reduces the number of relation options that LLM needs to consider during inference. However, hierarchical classification introduces the risk of error propagation across levels. To mitigate this, we propose a *prediction-then-verification* inference strategy that improves prediction reliability through multi-view verification at each level. Extensive experiments show that HCRE outperforms existing baselines, validating its effectiveness. We release our code at <https://github.com/XMUDeepLIT/HCRE>.

*Equal contribution.

†Corresponding author.

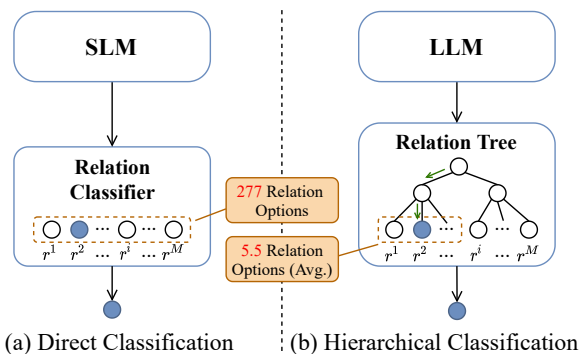


Figure 1: Comparison between the paradigm of existing cross-document RE model (Direct Classification) and ours (Hierarchical Classification). Green arrows indicate hierarchical classification based on the relation tree.

1 Introduction

As a fundamental NLP task, relation extraction (RE) aims to extract structured relational triples from unstructured texts. In this aspect, early studies (Zeng et al., 2014; Cai et al., 2016; Yao et al., 2019; Zhang et al., 2023c; Zhang et al., 2024; Zhang et al., 2025b), mainly focus on identifying target relations between the head and tail entities from a single sentence or document. However, numerous relational facts involve head and tail entities that do not co-occur in the same sentence or document. For instance, Yao et al. (2021) indicates that over half of Wikidata’s relational facts span multiple documents. Consequently, many researchers have shifted their attention to cross-document RE. Unlike conventional RE, cross-document RE requires the comprehensive analysis of multiple lengthy documents to predict relations between entities located in different documents.

Existing approaches to cross-document RE can be roughly divided into two categories. The first category of studies (Yao et al., 2021; Wang et al., 2022; Lu et al., 2023; Son et al., 2023; Na et al., 2024) primarily explore methods for extracting evidential context from lengthy documents, enabling

the Small Language Models (SLMs) to effectively identify cross-document relations. The second category of studies (Yao et al., 2021; Wu et al., 2023; Yue et al., 2024) mainly focus on modeling long-range dependencies between head and tail entities that are connected through relevant entities within the documents. Despite their progress, these approaches remain confined to the paradigm of “*SLM + Classifier*”. As illustrated in Figure 1(a), this paradigm requires models to directly select the target relation from an extensive predefined relation set, which poses significant challenges for SLMs due to their limited language understanding capability, thereby hindering further performance improvement.

Given the strong capabilities demonstrated by Large Language Models (LLMs) in various NLP tasks (Wang et al., 2023a; Wadhwa et al., 2023; Li et al., 2023; Zhou et al., 2024; Shao et al., 2024; Tu et al., 2025; Lian et al., 2026), applying LLMs to cross-document RE appears to be a promising research direction. Driven by this goal, we conduct a preliminary study to explore the performance of LLMs in cross-document RE. Our findings indicate that the performance of LLMs is limited and sometimes even lower than that of strong SLMs. Further analysis reveals that this suboptimal performance is largely attributable to the difficulty LLMs face when handling a large number of predefined relations in cross-document RE. On the one hand, an excessive number of relations hinders the model’s ability to distinguish between semantically similar relations. On the other hand, listing all predefined relations unnecessarily increases the context length, diverting the attention of LLMs away from critical details in documents.

Based on this insight, we propose an LLM-based Hierarchical Classification model for cross-document RE (HCRE). It consists of two core components: 1) an LLM used for relation prediction and 2) a *hierarchical relation tree* that narrows down the relation options during prediction. We carefully design a pipeline to construct a hierarchical relation tree, where each leaf node corresponds to a predefined relation and intermediate nodes represent higher-level concepts of their respective children. Guided by this tree, our LLM performs *hierarchical classification*, inferring a target relation in a top-down, level-by-level manner, as shown in Figure 1(b). In this way, we present the LLM with only a small-scale set of tree nodes as relation options at each level, thereby alleviating the challenge

posed by a large relation set for the LLM.

Although hierarchical classification mitigates the issue of excessive predefined relations, it also introduces the issue of error propagation across levels. To alleviate this issue, we propose a level-wise *prediction-then-verification* inference strategy. This strategy consists of two steps: 1) *Prediction step*. The LLM selects the best node and a sub-optimal node from the given options based on the input instance. 2) *Verification step*. Through replacing selected nodes with their children, we further construct multiple verification option sets. These sets are then fed back to the LLM to obtain additional predictions, which are used to verify the result from the prediction step. By iterating the above two steps, our strategy eliminates error-prone options and obtains a more accurate prediction at each level, effectively mitigating error propagation during hierarchical classification.

To summarize, our contributions are threefold:

- To the best of our knowledge, our work is the first to explore the LLM-based hierarchical classification for cross-document RE. To investigate the performance of LLMs in this task, we identify a critical issue: the excessive number of relation options is the primary bottleneck limiting LLM performance in cross-document RE. This finding not only motivates our framework design but also provides a research direction for the community.
- To address the crucial issue, we propose a LLM-based *hierarchical classification* model for cross-document RE, which reduces the number of relation options via a hierarchical relation tree, effectively improving the performance of LLMs in cross-document RE.
- To alleviate error propagation during hierarchical classification, we propose a level-wise *prediction-then-verification* inference strategy, which improves prediction reliability through multi-view verification at each tree level.

2 Preliminary Study

In this section, we conduct a preliminary study on the CodRED dataset to investigate the challenges of directly applying LLMs to cross-document RE, including evaluation metrics and model performance.

Baselines	max F1	micro F1	binary F1
End-to-End	49.07	33.33	41.76
ECRIM	61.64	39.25	42.98
NEPD	63.53	25.77	32.00

Table 1: Comparison among maximum F1, micro F1 and binary F1.

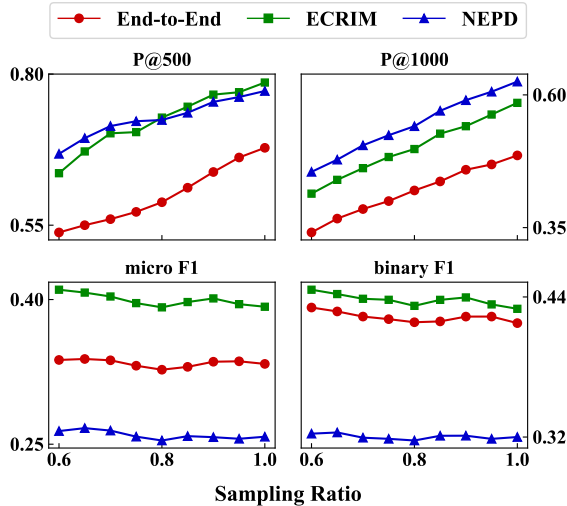


Figure 2: Comparison among P@K, micro F1, and binary F1 under varying dataset sizes.

2.1 Challenges in Evaluation Metrics

To analyze the reliability of commonly used metrics, we conduct a series of experiments on the CodRED benchmark (Yao et al., 2021) using several representative RE models, including End-to-End (Yao et al., 2021), ECRIM (Wang et al., 2022), and NEPD (Yue et al., 2024). To be specific, we investigate four evaluation metrics: 1) **Maximum F1**. This metric denotes the maximum F1 score on the Precision-Recall curve. 2) **P@K**. It measures model precision among the K relation triples with the highest confidence. 3) **Micro F1**. This classic metric provides an accurate measure for the trade-off between precision and recall across different relation classes. 4) **Binary F1**. A variant of micro F1, this metric treats all positive relations as a single positive class and considers the NA (Not Available) label as the negative class, aiming to assess the ability of RE models in discriminating whether any relation exists between target entities.

First, we review the calculation of **maximum F1**, which requires adjusting the decision threshold to achieve the best balance between precision and recall. Since the optimal threshold is unavailable

Task Instruction
You are an expert in the Cross-document Relation Extraction task. Given the context, consider what's the most precise relation between target entities from the given options.
Query
Context: <i>[Preprocessed Text Path]</i> Given the above documents, please tell me the relation between "[<i>head entity</i>]" and "[<i>tail entity</i>]" according to the following options.
Options: <i>[Options]</i>
Relation:

Figure 3: Our prompt template. Each prompt contains the preprocessed text path, head and tail entities, and relation options.

in real-world scenarios, maximum F1 fails to reflect the actual model ability and instead only reflects the ideal performance. Therefore, relying on maximum F1 may lead to an overestimation of the model's practical performance. From Table 1, we can observe that maximum F1 scores consistently exceed micro F1 and binary F1 across all models, further demonstrating its tendency to overestimate model performance.

Second, to analyze the effect of evaluation dataset scale on **P@K**, we compare the model performance on subsets of different sizes sampled from the development set of CodRED. From Figure 2, we observe that: 1) P@K increases with larger dataset size, indicating that P@K is sensitive to dataset scale; 2) P@K (e.g., P@500) may yield inconsistent model performance rankings under varying evaluation dataset scales. These results suggest that P@K is not sufficiently reliable for real-world scenarios, where the number of test instances is not fixed. In contrast, micro F1 and binary F1 remain consistently stable across all dataset scales, demonstrating greater reliability for evaluation.

Third, maximum F1 and P@K require assigning scores to all predefined relations simultaneously, whereas LLMs are trained to generate relation labels token by token. This mismatch makes such metrics unsuitable for LLMs.

Based on the above analysis, we adopt **micro F1** and **binary F1** as our primary evaluation metrics in subsequent experiments for the following reasons: 1) Both metrics avoid the flaws discussed earlier. 2) Given the large proportion of negative instances in cross-document RE (Yao et al., 2021; Yue et al., 2024), it is crucial for RE models to

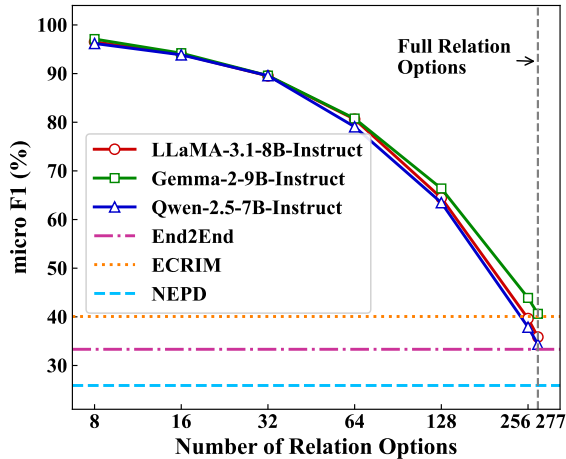


Figure 4: Effect of the number of relation options on LLMs. Note that CodRED involves 277 predefined relations. The horizontal dashed lines denote SLM performance serving as baselines.

distinguish positive instances from negative ones, which motivates our use of binary F1.

2.2 Challenges in Model Performance

Here, we explore the performance of applying an LLM directly to cross-document RE. To this end, we choose LLaMA-3.1-8B-Instruct (Meta, 2024), Gemma-2-9B-Instruct (Gemma, 2024), and Qwen-2.5-7B-Instruct (Yang et al., 2024) as the representative LLMs and compare them with the previous SLM-based baselines. All models are fine-tuned on the training set of CodRED dataset and micro F1 scores on the development set are reported. As shown in Figure 3, we employ a prompt template to transform each input instance into a text prompt, thereby guiding LLMs to directly generate the target relation.

As depicted by the gray dashed line in Figure 4, the performance of directly fine-tuned LLMs is unsatisfactory, sometimes even underperforming strong SLM-based baselines. We argue that the sub-optimal performance of LLMs is attributed to the massive predefined relations in cross-document RE, which makes it difficult for LLMs to distinguish similar relations and leads to an overlong input prompt, thereby interfering with model prediction.

To verify our hypothesis, we further assess their performance with varying numbers of relation options. Specifically, during both training and inference, we randomly remove a certain number of relation options from the original prompt for each instance. As illustrated in Figure 4, reducing the number of relation options leads to a significant

performance improvement in LLMs. This motivates us to explore methods that reduce the number of relation options to enhance model predictions.

3 Our Model

Based on the findings from our preliminary study, we propose HCRE, an LLM-based hierarchical classification model for cross-document RE. In this section, we provide a comprehensive description of our hierarchical classification model, which consists of an LLM \mathcal{M}_1 and a hierarchical relation tree. Concretely, we first introduce a pipeline to construct a *hierarchical relation tree* in Section 3.1. Based on this tree, we elaborate on the LLM-based *hierarchical classification* process in Section 3.2. Finally, we detail the model inference and training strategies in Sections 3.3 and 3.4, respectively.

3.1 Hierarchical Relation Tree

To address the performance degradation in LLM-based cross-document RE caused by numerous predefined relations, we construct a *hierarchical relation tree* to effectively organize these relations. To achieve this, we design a pipeline that employs an advanced LLM \mathcal{M}_2 (such as GPT-4o) to construct the tree based on the semantics of predefined relations. In this pipeline, we initialize the hierarchical relation tree with a root node encompassing all predefined relations, and then recursively partition upper-level nodes to obtain lower-level nodes, constructing the hierarchy level by level.

To enhance the distinctiveness of nodes at each level, we prompt the LLM \mathcal{M}_2 to generate a textual partitioning criterion C_l for each level l . At the l -th level, we prompt \mathcal{M}_2 with C_l to partition the relations contained in each current-level node, thereby producing child nodes at the $(l+1)$ -th level. Then, \mathcal{M}_2 generates a node name for each child node, reflecting the common aspects of relations it contains under the criterion C_l . For example, using the criterion “*domain*” enables us to split each node according to the domains of its corresponding relations, forming child nodes such as “*politics*” and “*finance*”. This partitioning process continues until a maximum tree depth L is reached.

In addition, to alleviate the label imbalance in cross-document RE, we further impose a constraint on the second level of the tree, limiting it to two nodes: “*valid relations*”, which includes all positive relations, and “*no valid relation*”, which contains only NA. This design explicitly separates pos-

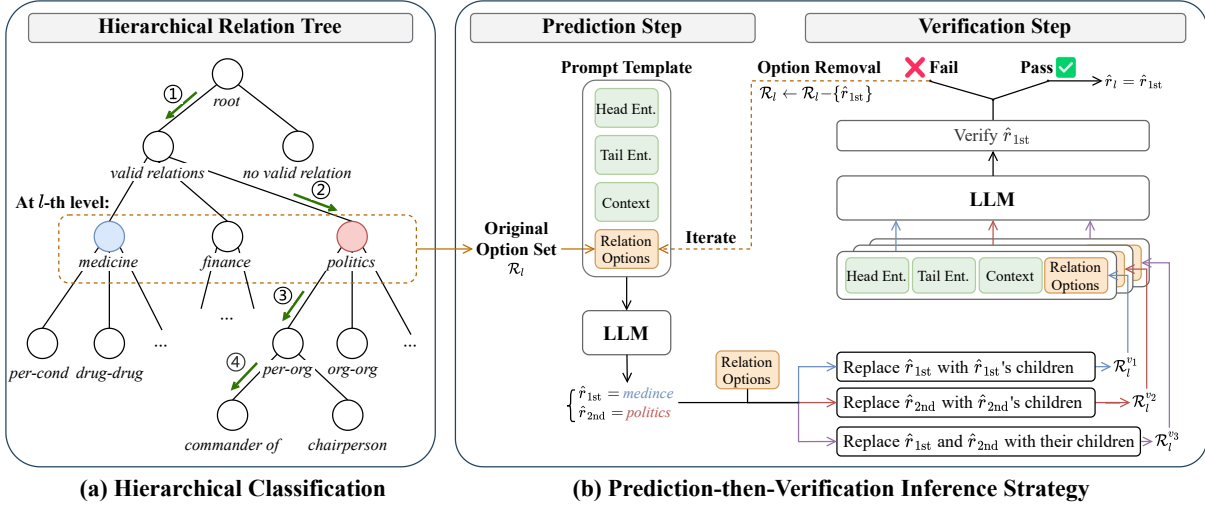


Figure 5: Overview of HCRE. (a) Guided by the hierarchical relation tree, the LLM \mathcal{M}_1 performs *hierarchical classification* in a top-down manner (See green arrows). (b) At each level, we refine model prediction by *prediction-then-verification* inference strategy, which is composed of the *prediction step* and the *verification step*.

itive relations from NA, enabling \mathcal{M}_1 to better distinguish between these two kinds of instances and reducing the model’s bias towards predicting NA due to its overabundance in the training data.

Following the above pipeline, we obtain a hierarchical relation tree, where leaf nodes correspond to predefined relations and intermediate nodes represent higher-level concepts of their respective child nodes. More details regarding the tree are provided in **Appendix A**.

3.2 LLM-based Hierarchical Classification

After obtaining the hierarchical relation tree, for each instance x (context, head entity, and tail entity), we guide the LLM \mathcal{M}_1 to infer its target relation at a leaf node by performing a top-down, level-by-level *hierarchical classification*, as shown in Figure 5(a). At each level, \mathcal{M}_1 only needs to consider a small set of relation options, effectively avoiding suboptimal performance caused by an excessive number of relation options.

Since \mathcal{M}_1 performs similar computations at each level of the tree, we illustrate this procedure in detail using the l -th level as an example. Here, \hat{r}_{l-1} refers to the node chosen by \mathcal{M}_1 at the $(l-1)$ -th level, and \mathcal{R}_l denotes the set of its child nodes at the l -th level. Next, we use \mathcal{R}_l as the set of relation options to instantiate the prompt template in Figure 3, thereby prompting \mathcal{M}_1 to select the optimal node \hat{r}_l from \mathcal{R}_l for the next level. This process is repeated until a leaf node corresponding to the target relation of instance x is reached. Crucially, the number of child nodes $|\mathcal{R}_l|$ for any parent is much

smaller than the total number of predefined relations, significantly reducing the number of options \mathcal{M}_1 must consider during inference.

3.3 Prediction-then-Verification Inference Strategy

Although hierarchical classification effectively reduces the number of relation options considered by \mathcal{M}_1 per level, errors occurring at earlier levels may propagate and affect predictions at subsequent levels. To address the issue of error propagation, we propose a *prediction-then-verification* inference strategy to refine model prediction at each level through multi-view verification. As illustrated in Figure 5(b), our strategy enhances the reliability of model prediction at each level by iterating the following two steps:

Prediction Step. At the l -th level of the hierarchical relation tree, we first adopt the children of the node \hat{r}_{l-1} predicted at the $(l-1)$ -th level as the relation option set \mathcal{R}_l . Then, \mathcal{M}_1 selects the best and suboptimal nodes from \mathcal{R}_l , denoted as \hat{r}_{1st} and \hat{r}_{2nd} , respectively. Intuitively, \mathcal{M}_1 tends to confuse \hat{r}_{1st} and \hat{r}_{2nd} , which may lead to an incorrect prediction.

Verification Step. To mitigate the confusion between \hat{r}_{1st} and \hat{r}_{2nd} , we validate the reliability of \hat{r}_{1st} using multiple *verification option sets*. By treating child nodes as a finer-grained alternative view of their parent node, we replace each node with its children to construct these sets. Specifically, we replace \hat{r}_{1st} and \hat{r}_{2nd} with their respective child nodes in the original relation option set \mathcal{R}_l ,

forming two verification option sets \mathcal{R}_l^{v1} and \mathcal{R}_l^{v2} . Meanwhile, the third verification option set \mathcal{R}_l^{v3} is obtained by simultaneously replacing both best and suboptimal nodes with their child nodes in \mathcal{R}_l .

In this way, we obtain three alternative views of \mathcal{R}_l , which allow us to verify whether \mathcal{M}_1 consistently predicts nodes that are semantically aligned with the best node \hat{r}_{1st} (i.e., exactly matches \hat{r}_{1st} or one of its children). Concretely, \mathcal{M}_1 is prompted to select the best node from each verification option set to serve as an *auxiliary verification node*. If more than half of the auxiliary verification nodes are semantically aligned with \hat{r}_{1st} , \hat{r}_{1st} is considered as a reliable and final prediction \hat{r}_l at the current level, and then we proceed to the next level of the hierarchical relation tree. Otherwise, \hat{r}_{1st} is deemed incorrect and removed from \mathcal{R}_l , after which we repeat the above two steps.

It should be noted that \mathcal{R}_l^{v1} , \mathcal{R}_l^{v2} and \mathcal{R}_l^{v3} are essentially three equivalent views of \mathcal{R}_l at a finer granularity. During the verification step, by incorporating next-level nodes, the verification option sets provide finer-grained semantic information compared to the original relation option set \mathcal{R}_l . This finer granularity enables \mathcal{M}_1 to effectively discern subtle differences between \hat{r}_{1st} and \hat{r}_{2nd} , ultimately resulting in a more reliable prediction. Further elaboration on the prediction-then-verification inference strategy is provided in **Appendix J**.

3.4 Model Training

To effectively train our LLM for hierarchical classification in cross-document RE, we reconstruct the training samples accordingly.

For each training sample (x, \mathcal{R}, r) , we begin by identifying a path r_0, r_1, \dots, r_{L-1} from the root node to the leaf node $r_{L-1} = r$ within the hierarchical relation tree. Here, x , \mathcal{R} , r , and L represent the input instance, predefined relation set, target relation, and tree depth, respectively. Utilizing this path, we extend (x, \mathcal{R}, r) to $L-1$ level-wise training samples $\{(x, \mathcal{R}_l, r_l)\}_{l=1}^{L-1}$, where \mathcal{R}_l denotes the relation option set containing r_l and its siblings. This process is repeated for all training samples to form a new dataset \mathcal{D}_1 .

On top of that, to simulate the verification step, we construct another dataset \mathcal{D}_2 . For each training sample (x, \mathcal{R}_l, r_l) in \mathcal{D}_1 , three new training samples are derived: $\{(x, \mathcal{R}_l^{v1}, r_{l+1}), (x, \mathcal{R}_l^{v2}, r_l), (x, \mathcal{R}_l^{v3}, r_{l+1})\}$. During the construction process, the ground-truth node is treated as the best node r_{1st} , and a randomly selected sibling node serves as

	Closed		Open
	Train	Dev	Dev
#Text Path (Pos.)	8,263	2,558	4,558
#Text Path (NA)	120,925	38,182	15,072
#Tokens / Doc	4,938.6	5,031.6	62,863

Table 2: Statistics of CodRED. (Pos.: Positive; NA: Not Available.)

the suboptimal node r_{2nd} . These two nodes are then replaced with their respective child nodes to form the verification option sets \mathcal{R}_l^{v1} , \mathcal{R}_l^{v2} , and \mathcal{R}_l^{v3} . Finally, our LLM \mathcal{M}_1 is fine-tuned on the combined dataset $\mathcal{D}_1 \cup \mathcal{D}_2$.

4 Experiments

4.1 Experiment Setup

Datasets. We conduct our experiments on CodRED (Yao et al., 2021), a widely used benchmark for cross-document RE. CodRED provides two settings: the closed setting offers gold text paths in advance, whereas the open setting requires retrieving relevant text paths for relation prediction. Detailed statistics of CodRED are provided in Table 2.

Baselines. We primarily compare our model with two kinds of baselines: 1) *Cross-document RE Baselines*, including **End-to-End** (Yao et al., 2021), **ECRIM** (Wang et al., 2022), **MR.COD** (Lu et al., 2023), **KXDocRE** (Jain et al., 2024a), **REIC** (Na et al., 2024), and **NEPD** (Yue et al., 2024); and 2) *LLM-based Hierarchical Text Classification (HTC) Baselines*, including **Rs-ICL** (Chen et al., 2024), **DFS-L** (Yu et al., 2022) and **BFS-L** (Huang et al., 2022; Jain et al., 2024b). Moreover, we include a directly fine-tuned baseline, denoted as **Vanilla**. The details of these baselines are provided in **Appendix B**.

In addition, the implementation details of our model are provided in **Appendix C**.

4.2 Main Results

The experimental results on both closed and open settings of CodRED are shown in Table 3. Based on these results, we draw several key conclusions:

First, our model consistently outperforms all SLM-based baselines under both settings, demonstrating the potential of LLMs for cross-document RE. In particular, compared to the strongest SLM-based baseline, RoBERTa + NEPD, our model achieves gains of 2.39 and 5.52 points in micro

Backbone	Model	Closed		Open	
		micro F1	binary F1	micro F1	binary F1
<i>Cross-Document RE Baselines</i>					
BERT-base	End-to-End (Yao et al., 2021)	33.33	41.76	22.23	28.85
	ECRIM (Wang et al., 2022)	39.25	42.98	19.81	22.06
	KXDocRE (Jain et al., 2024a)	37.53	38.25	18.80	19.39
	REIC (Na et al., 2024)	38.75	46.25	19.50	23.77
	NEPD (Yue et al., 2024)	25.77	32.00	29.16	36.36
RoBERTa-large	End-to-End (Yao et al., 2021)	41.45	47.99	25.35	29.60
	ECRIM (Wang et al., 2022)	42.54	49.47	23.39	27.60
	KXDocRE (Jain et al., 2024a)	42.55	45.36	21.74	23.23
	REIC (Na et al., 2024)	40.17	48.74	22.49	27.97
	NEPD (Yue et al., 2024)	42.96	52.67	30.12	37.04
<i>LLM-based HTC Baselines</i>					
GPT-4o-mini	Rs-ICL (Chen et al., 2024)	11.22	39.26	10.55	39.86
LLaMA-3.1-8B	DFS-L. (Yu et al., 2022)	36.83	42.74	14.51	18.09
	BFS-L. (Jain et al., 2024b)	35.55	41.95	13.46	16.46
<i>Ours</i>					
LLaMA-3.1-8B	Vanilla	38.14	41.43	15.19	17.00
	HCRE	45.35₃‡	58.19₃‡	34.91₂‡	49.33₂‡

Table 3: Experimental results on CodRED under both closed and open settings. The subscript denotes the corresponding standard deviation (e.g., 45.35₃ represents 45.35 ± 0.3). ‡ indicates significance at $p < 0.01$ over the second-best baseline NEPD, based on 1,000 bootstrap tests (Tibshirani and Efron, 1993).

Model	micro F1	binary F1
Ours	45.35	58.19
w/o multi-view	39.37	49.63
w/o PtV	37.66	47.28
w/o LTC	43.18	56.60
w/o LTC, PtV	32.55	45.44
w/o HRT	38.14	41.43

Table 4: Ablation study of our model on CodRED under the closed setting. Note that LTC, PtV, and HRT denote level-wise tree construction, prediction-then-verification, and hierarchical relation tree, respectively.

F1 and binary F1 under the closed setting.

Second, in comparison with LLM-based HTC baselines, our model consistently achieves better performance. This demonstrates the effectiveness of our prediction-then-verification inference strategy for hierarchical classification by mitigating error propagation.

Third, while the Vanilla baseline does not outper-

form all the baselines, our model improves performance based on it and surpasses all the baselines. This suggests that reducing the number of relation options can effectively enhance the performance of LLMs in cross-document RE.

4.3 Ablation Study

In Table 4, we investigate the effects of each component in our model to verify their effectiveness. Specifically, we compare our model with the following variants:

(1) *w/o multi-view*. In this variant, instead of adopting three verification option sets $\{\mathcal{R}^1, \mathcal{R}^2, \mathcal{R}^3\}$, we only adopt the first option set \mathcal{R}^1 for verification. which leads to a notable performance drop. This result highlights the importance of incorporating fine-grained information from multiple views, which enables the LLM \mathcal{M}_1 to verify the reliability of predictions more effectively.

(2) *w/o PtV*. The removal of the prediction-then-verification (PtV) inference strategy from HCRE results in a substantial performance drop. This

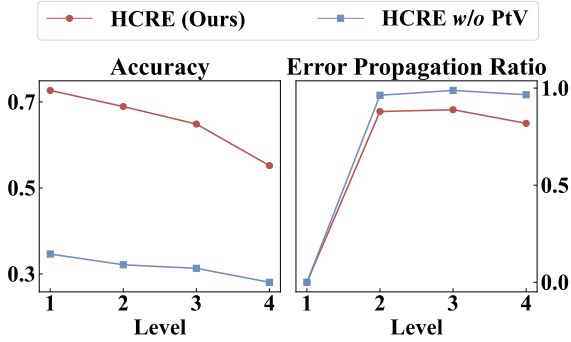


Figure 6: The accuracy and error propagation ratio of our model, with and without the PtV strategy.

underscores the critical role of the verification step in hierarchical classification, as it improves overall performance by mitigating errors at each level.

(3) *w/o LTC*. Here, we simply prompt \mathcal{M}_2 to directly generate the hierarchical relation tree based on the predefined relations, rather than constructing the relation tree level by level. In this variant, we observe a slight performance drop. This is potentially because the hierarchical relation tree constructed by our level-wise pipeline comprises more distinguishable nodes, allowing LLMs to identify target relations more accurately. More details about this variant are in **Appendix D**.

(4) *w/o LTC, PtV*. In this variant, we further remove the PtV strategy from the *w/o LTC* variant and observe a more significant performance degradation. We attribute this to the stronger semantic relevance between parent and child nodes in our tree, which enables child nodes to better represent their parent node during the verification step.

(5) *w/o HRT*. Different from the above variants, this variant requires \mathcal{M}_1 to select target relations from the entire predefined relation set. A slight performance decline is observed compared to the *w/o PtV* variant, suggesting \mathcal{M}_1 makes better relation prediction, benefiting from the hierarchical relation tree that reduces the number of relation options considered during inference.

4.4 Analysis of Error Propagation

To verify that the PtV strategy effectively mitigates error propagation during hierarchical classification, we evaluate the accuracy of our model at each level. Additionally, we focus on the misclassification instances caused by incorrect predictions at previous levels and define the proportion of such instances among all misclassifications as *error propagation ratio*. As illustrated in Figure 6, the PtV strategy

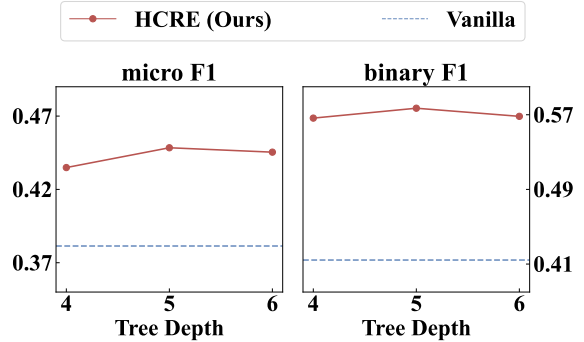


Figure 7: Model performance on the CodRED development set using trees with varying tree depth L .

not only consistently improves model performance but also mitigates error propagation at all levels.

4.5 Effect of Tree Depth L

To investigate the impact of predefined tree depth L , we evaluate our model using relation trees with $L \in \{4, 5, 6\}$. Figure 7 shows that HCRE is not sensitive to the predefined tree depth, exhibiting strong robustness. Since our model achieves the best performance at $L = 5$, we adopt 5 as the relation tree depth in all experiments.

We additionally present analysis of error propagation, bag-level evaluation, experiments on conventional evaluation metrics, studies using different backbone architectures, cross-dataset generalization results of HCRE, and more ablation experiments about PtV in **Appendices E, F, G, H, I, and J**, respectively.

5 Related Works

5.1 Cross-document Relation Extraction.

Cross-document RE aims to identify the relations between entities appearing in different documents, a task first introduced by Yao et al. (2021). Subsequent studies mainly advance this task along two directions: evidence retrieval and long-range relation representation. For evidence retrieval, prior work aims to reduce irrelevant context by selecting informative sentences or paths, such as document filtering based on entity co-occurrence (Wang et al., 2022), graph-based evidence path mining (Lu et al., 2023), text path expansion via bridge entities (Son et al., 2023), and reinforcement learning-based sentence selection (Na et al., 2024). Another line of research focuses on modeling long-range dependencies. Representative approaches include cross-path relation attention (Wang et al., 2022),

local-to-global causal reasoning over text paths (Wu et al., 2023), unified entity graph construction (Yue et al., 2024), and domain knowledge injection (Jain et al., 2024a). Despite their progress, their models still follow the “*SLM + Classifier*” paradigm, constrained by the limited language understanding capabilities of SLMs. In contrast, our work explores LLM-based hierarchical classification for cross-document RE and avoids this flaw.

5.2 LLM-based Hierarchical Text Classification.

In recent years, LLMs have shown strong capabilities in HTC, which aims to predict labels organized in a hierarchy. Some works (Wang et al., 2023b; Paletto et al., 2024; Zhang et al., 2025c) adopt LLMs as data annotators, highlighting the application of LLMs to generate high-quality data and enrich label taxonomies for HTC. Other works explore utilizing LLMs to enhance HTC at inference time, training models to perform hierarchical classification either by converting label hierarchies into sequences using depth-first (Yu et al., 2022) or breadth-first orders (Huang et al., 2022; Jain et al., 2024b) or by classifying level by level (Jain et al., 2024b; Chen et al., 2024; Tabatabaei et al., 2025). Despite the effectiveness of these methods, they often neglect error propagation during hierarchical classification. In contrast, our model mitigates this issue through multi-view verification at each level.

6 Conclusion and Future Work

In this paper, we have proposed an LLM-based hierarchical classification model for cross-document RE. Specifically, we utilize the predefined relations to construct a hierarchical relation tree, which guides our LLM to infer target relations level by level. Furthermore, we propose a prediction-then-verification inference strategy to refine model predictions at each level, thereby mitigating error propagation. Empirical results on the commonly used benchmark CodRED show the superiority of HCRE over existing baselines. In the future, we plan to investigate the generalization ability of our model on more information extraction tasks.

Limitations

Our study has two main limitations that warrant further investigation. First, due to the limitation of input context length, our LLM can only handle a single text path during each inference, which pre-

vents the model from leveraging cross-path dependency information that could potentially enhance its performance. Second, during the training of our model, we randomly sample a node as an alternative to the suboptimal node, which may not be the optimal strategy.

Acknowledgments

The project was supported by National Natural Science Foundation of China (No. 62276219), Natural Science Foundation of Fujian Province of China (No. 2024J011001), and the Public Technology Service Platform Project of Xiamen (No.3502Z20231043). We also thank the reviewers for their insightful comments.

References

- Rui Cai, Xiaodong Zhang, and Houfeng Wang. 2016. [Bidirectional recurrent convolutional neural network for relation classification](#). In *ACL 2016*.
- Huiyao Chen, Yu Zhao, Zulong Chen, Mengjia Wang, Liangyue Li, Meishan Zhang, and Min Zhang. 2024. [Retrieval-style in-context learning for few-shot hierarchical text classification](#). *TACL 2024*.
- Team Gemma. 2024. [Gemma 2: Improving open language models at a practical size](#). *Preprint*, arXiv:2408.00118.
- Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. 2023. [Reasoning with language model is planning with world model](#). In *EMNLP 2023*.
- 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.
- Wei Huang, Chen Liu, Bo Xiao, Yihua Zhao, Zhaoming Pan, Zhimin Zhang, Xinyun Yang, and Guiquan Liu. 2022. [Exploring label hierarchy in a generative way for hierarchical text classification](#). In *COLING 2022*.
- Monika Jain, Raghava Mutharaju, Kuldeep Singh, and Ramakanth Kavuluru. 2024a. [Knowledge-driven cross-document relation extraction](#). In *Findings of ACL 2024*.
- Vidit Jain, Mukund Rungta, Yuchen Zhuang, Yue Yu, Zeyu Wang, Mu Gao, Jeffrey Skolnick, and Chao Zhang. 2024b. [HiGen: Hierarchy-aware sequence generation for hierarchical text classification](#). In *EACL 2024*.
- Peng Li, Tianxiang Sun, Qiong Tang, Hang Yan, Yuanbin Wu, Xuanjing Huang, and Xipeng Qiu. 2023. [CodeIE: Large code generation models are better few-shot information extractors](#). In *ACL 2023*.

- Shuquan Lian, Juncheng Liu, Yazhe Chen, Yuhong Chen, and Hui Li. 2026. [Swe-agile: A software agent framework for efficiently managing dynamic reasoning context](#). *Preprint*, arXiv:2604.11716.
- Ilya Loshchilov and Frank Hutter. 2019. [Decoupled weight decay regularization](#). *Preprint*, arXiv:1711.05101.
- Keming Lu, I-Hung Hsu, Wenxuan Zhou, Mingyu Derek Ma, and Muhao Chen. 2023. [Multi-hop evidence retrieval for cross-document relation extraction](#). In *Findings of ACL 2023*.
- AI Meta. 2024. [The llama 3 herd of models](#). *Preprint*, arXiv:2407.21783.
- Byeonghu Na, Suhyeon Jo, Yeongmin Kim, and Il-chul Moon. 2024. [Reward-based input construction for cross-document relation extraction](#). In *ACL 2024*.
- Lorenzo Paletto, Valerio Basile, and Roberto Esposito. 2024. [Label augmentation for zero-shot hierarchical text classification](#). In *ACL 2024*.
- Liangying Shao, Liang Zhang, Minlong Peng, Guoqi Ma, Hao Yue, Mingming Sun, and Jinsong Su. 2024. [One2Set + large language model: Best partners for keyphrase generation](#). In *EMNLP 2024*.
- Junyoung Son, Jinsung Kim, Jungwoo Lim, Yoonna Jang, and Heuseok Lim. 2023. [Explore the way: Exploring reasoning path by bridging entities for effective cross-document relation extraction](#). In *Findings of EMNLP 2023*.
- Seyed Amin Tabatabaei, Sarah Fancher, Michael Parsons, and Arian Askari. 2025. [Can large language models serve as effective classifiers for hierarchical multi-label classification of scientific documents at industrial scale?](#) In *COLING 2025*.
- Robert J Tibshirani and Bradley Efron. 1993. An introduction to the bootstrap. *Monographs on statistics and applied probability*, 57(1).
- Hongyao Tu, Liang Zhang, Yujie Lin, Xin Lin, Haibo Zhang, Long Zhang, and Jinsong Su. 2025. [LLM-OREF: An open relation extraction framework based on large language models](#). In *EMNLP 2025*.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. [Attention is all you need](#). *Preprint*, arXiv:1706.03762.
- Somin Wadhwa, Silvio Amir, and Byron Wallace. 2023. [Revisiting relation extraction in the era of large language models](#). In *ACL 2023*.
- Fengqi Wang, Fei Li, Hao Fei, Jingye Li, Shengqiong Wu, Fangfang Su, Wenxuan Shi, Donghong Ji, and Bo Cai. 2022. [Entity-centered cross-document relation extraction](#). In *EMNLP 2022*.
- Xiao Wang, Weikang Zhou, Can Zu, Han Xia, Tianze Chen, Yuansen Zhang, Rui Zheng, Junjie Ye, Qi Zhang, Tao Gui, Jihua Kang, Jingsheng Yang, Siyuan Li, and Chunsai Du. 2023a. [Instructuie: Multi-task instruction tuning for unified information extraction](#). *Preprint*, arXiv:2304.08085.
- Yue Wang, Dan Qiao, Juntao Li, Jinxiong Chang, Qishen Zhang, Zhongyi Liu, Guannan Zhang, and Min Zhang. 2023b. [Towards better hierarchical text classification with data generation](#). In *Findings of ACL 2023*.
- Zilong Wang, Zifeng Wang, Long Le, Steven Zheng, Swaroop Mishra, Vincent Perot, Yuwei Zhang, Anush Mattapalli, Ankur Taly, Jingbo Shang, Chen-Yu Lee, and Tomas Pfister. 2025. [Speculative RAG: Enhancing retrieval augmented generation through drafting](#). In *ICLR 2025*.
- Haoran Wu, Xiuyi Chen, Zefa Hu, Jing Shi, Shuang Xu, and Bo Xu. 2023. [Local-to-global causal reasoning for cross-document relation extraction](#). *IEEE/CAA Journal of Automatica Sinica*, 10(7).
- Lilong Xue, Dan Zhang, Yuxiao Dong, and Jie Tang. 2024. [AutoRE: Document-level relation extraction with large language models](#). In *ACL 2024*.
- An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 40 others. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.
- Yuan Yao, Jiaju Du, Yankai Lin, Peng Li, Zhiyuan Liu, Jie Zhou, and Maosong Sun. 2021. [CodRED: A cross-document relation extraction dataset for acquiring knowledge in the wild](#). In *EMNLP 2021*.
- Yuan Yao, Deming Ye, Peng Li, Xu Han, Yankai Lin, Zhenghao Liu, Zhiyuan Liu, Lixin Huang, Jie Zhou, and Maosong Sun. 2019. [DocRED: A large-scale document-level relation extraction dataset](#). In *ACL 2019*.
- Chao Yu, Yi Shen, and Yue Mao. 2022. [Constrained sequence-to-tree generation for hierarchical text classification](#). In *SIGIR 2022*.
- Hao Yue, Shaopeng Lai, Chengyi Yang, Liang Zhang, Junfeng Yao, and Jinsong Su. 2024. [Towards better graph-based cross-document relation extraction via non-bridge entity enhancement and prediction debiasing](#). In *Findings of ACL 2024*.
- Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. [Relation classification via convolutional deep neural network](#). In *COLING 2014*.
- Fu Zhang, Hongsun Yu, Jingwei Cheng, and Huangming Xu. 2025a. [Entity pair-guided relation summarization and retrieval in LLMs for document-level relation extraction](#). In *Findings of NAACL 2025*.

- Kaizhong Zhang and Dennis Shasha. 1989. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6).
- Liang Zhang, Zijun Min, Jinsong Su, Pei Yu, Ante Wang, and Yidong Chen. 2023a. Exploring effective inter-encoder semantic interaction for document-level relation extraction. In *IJCAI 2023*.
- Liang Zhang, Jinsong Su, Yidong Chen, Zhongjian Miao, Min Zijun, Qingguo Hu, and Xiaodong Shi. 2022. Towards better document-level relation extraction via iterative inference. In *EMNLP 2022*.
- Liang Zhang, Jinsong Su, Zijun Min, Zhongjian Miao, Qingguo Hu, Biao Fu, Xiaodong Shi, and Yidong Chen. 2023b. Exploring self-distillation based relational reasoning training for document-level relation extraction. In *AAAI 2023*.
- Liang Zhang, Zhen Yang, Biao Fu, Ziyao Lu, Liangying Shao, Shiyu Liu, Fandong Meng, Jie Zhou, Xiaoli Wang, and Jinsong Su. 2024. Multi-level cross-modal alignment for speech relation extraction. In *EMNLP 2024*.
- Liang Zhang, Yang Zhang, Ziyao Lu, Fandong Meng, Jie Zhou, and Jinsong Su. 2025b. A self-denoising model for robust few-shot relation extraction. In *ACL 2025*.
- Liang Zhang, Chulun Zhou, Fandong Meng, Jinsong Su, Yidong Chen, and Jie Zhou. 2023c. HyperNetwork-based decoupling to improve model generalization for few-shot relation extraction. In *EMNLP 2023*.
- Yunyi Zhang, Ruozhen Yang, Xueqiang Xu, Rui Li, Jinfeng Xiao, Jiaming Shen, and Jiawei Han. 2025c. Teleclass: Taxonomy enrichment and llm-enhanced hierarchical text classification with minimal supervision. *Preprint*, arXiv:2403.00165.
- Sizhe Zhou, Yu Meng, Bowen Jin, and Jiawei Han. 2024. Grasping the essentials: Tailoring large language models for zero-shot relation extraction. In *EMNLP 2024*.

A Details of Hierarchical Relation Tree

A.1 Prompt Templates for Tree Construction

As illustrated in Figure 8, our tree construction pipeline begins by deriving a set of partitioning criteria to progressively partition the relations. Using these criteria, we then recursively construct child nodes and populate them with relevant relations, constructing the hierarchy level by level.

Specifically, we first adopt the following prompt template to generate partitioning criteria.

Prompt Template for Criterion Generation

Background:

You are an expert in cross-document relation extraction, which aims to identify predefined relations between entities that appear in different documents.

Task:

Your task is to analyze and provide several distinct classification criteria which is beneficial to group predefined relations based on:

- Homogeneity: Relations in the same category should be highly similar.
- Heterogeneity: Different categories should be as distinct as possible.

Requirements:

1. Provide 10-12 distinct classification criteria with concise names (1-2 words).
2. Choose the top 2 criteria that might yield the most effective classification results.
3. Return the result in a valid JSON format as shown below.

JSON Output Format:

```
```json
{
 "classification criteria": {
 "criterion name 1": {
 "explanation": "explanation of criterion 1",
 "possible category names": ["category name 1", "category name 2"]
 },
 "criterion name 2": {
 "explanation": "explanation of criterion 1",
 "possible category names": ["category name 1", "category name 2"]
 },
 ...
 },
 "top2 criteria": ["criterion name i", "crite-
```

```
 rion name j"]
}
```

##### Output:

Then, we employ the prompt template below to construct child nodes by dividing a parent node into multiple nodes. Here, **[CRITERION\_NAME]** refers to a textual criterion, **[CRITERION\_EXPLANATION]** denotes its corresponding description, and **[RELATION\_WITH\_DESC]** represents the relations contained in the parent node along with their descriptions.

#### Prompt Template for Node Name Generation

##### Task:

Analyze the given relation types and their descriptions, and categorize these relations into 10-12 clusters based on their **[CRITERION\_NAME]**s, where “[**CRITERION\_NAME**]” is defined as: “[**CRITERION\_EXPLANATION**]”.

##### Requirements:

1. Each **[CRITERION\_NAME]** should have a **CONCISE, CLEAR and STRUCTURED** name (1-2 words) reflecting its theme (e.g., **[CRITERION\_EXAMPLES]**).
2. Ensure that **[CRITERION\_NAME]**s do not overlap in meaning, with each covering a single **[CRITERION\_NAME]**.
3. Ensure that **[CRITERION\_NAME]**s cover **ALL** provided relation types.
4. Provide a brief yet precise description (~15 words) for each **[CRITERION\_NAME]**.
5. Return the result in a valid JSON format as shown below.

##### JSON Output Format:

```
```json
{
  "name of [CRITERION_NAME] 1": "description of [CRITERION_NAME] 1",
  "name of [CRITERION_NAME] 2": "description of [CRITERION_NAME] 2",
  ...
}
```

Relation Types and Descriptions:

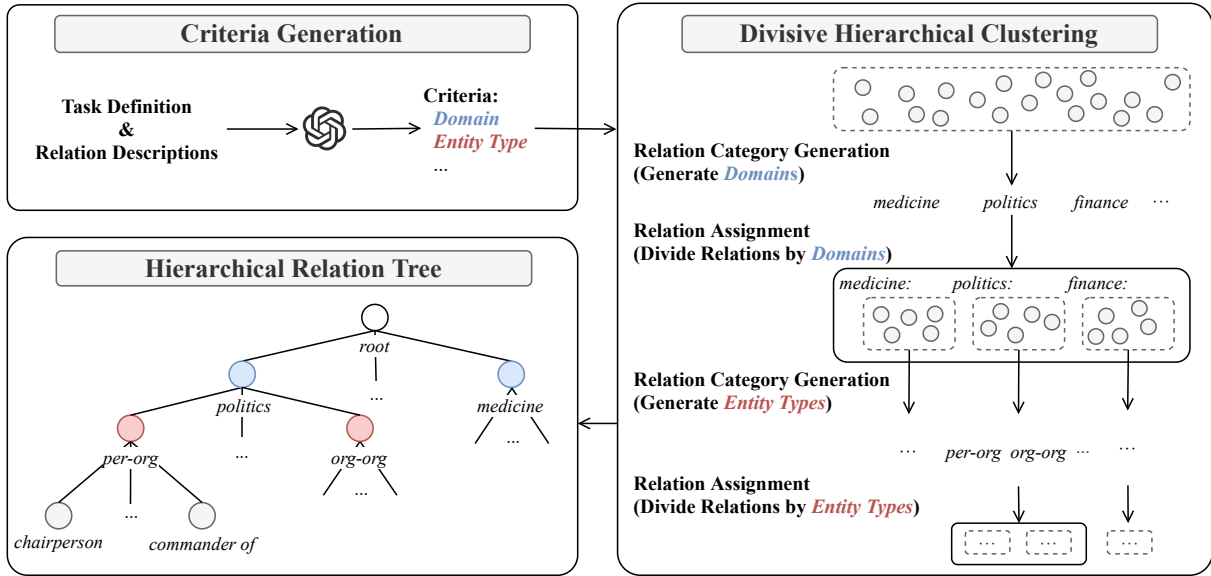


Figure 8: Illustration of the level-wise pipeline for hierarchical relation tree construction.

```

```json
[RELATION_WITH_DESC]
```
Output:

```

Next, we assign each relation to the most suitable tree nodes. In this template, `[CRITERION_NAME]` denotes a textual criterion, `[REL_EXPLANATION]` refers to a predefined relation, `[REL_DESC]` is its description, and `[CRITERION_INSTANCES]` corresponds to the node names generated in the previous step.

```

Prompt Template for Relation Assignment

Task:
You are tasked with analyzing the [CRITERION_NAME] of the relation label "[REL_NAME]" based on its description: "[REL_DESC]". Determine which [CRITERION_NAME](s) best align with "[REL_NAME]".
Available [CRITERION_NAME]s:
Below is a list of [CRITERION_NAME]s with their descriptions:
```json
[CRITERION_INSTANCES]
```
Output Format:
Provide your answer as a NON-EMPTY JSON array containing the [CRITERION_NAME](s):

```

```

```json
["[CRITERION_NAME] 1", "[CRITERION_NAME] 2", ...]
```
Output:

```

A.2 API Cost of Tree Construction

According to the cost analysis in Table 6, constructing one relation tree with GPT-4o incurs a total API cost of \$0.9641. Since tree construction is performed only once for each relation schema, the overall overhead is negligible, underscoring the economic efficiency of our proposed pipeline.

A.3 Statistics of Hierarchical Relation Tree

We provide the detailed statistics of our hierarchical relation tree in Table 8. It is noted that there are more leaf nodes than predefined relations in the hierarchical relation tree. This occurs because some predefined relations inherently correspond to multiple high-level concepts, and thus may be assigned to more than one node. For instance, it is reasonable for the relation "*significant person*" to be contained by both intermediate nodes "*politics*" and "*creative works*".

A.4 Robustness of Hierarchical Relation Tree

Here, we construct hierarchical relation trees with varying random seeds and criterion sets. We first assess the tree edit similarities (Zhang and Shasha, 1989) among trees generated with different random

| Tree Construction Parameters | Value | micro F1 | binary F1 |
|------------------------------|-------------------------|----------|-----------|
| Random Seed | 42 | 45.35 | 58.19 |
| | 666 | 44.50 | 57.42 |
| | 1024 | 44.55 | 57.98 |
| Criterion Set | (Domain, Entity Type) | 45.35 | 58.19 |
| | (Domain, Polarity) | 45.10 | 57.08 |
| | (Entity Type, Polarity) | 44.14 | 57.36 |

Table 5: Experiment results with varying tree construction parameters.

| #Input Tokens | #Output Tokens | Total |
|---------------|----------------|----------|
| 334,793 | 12,713 | 347,506 |
| Input Cost | Output Cost | Total |
| \$0.8370 | \$0.1271 | \$0.9641 |

Table 6: API cost of hierarchical relation tree construction.

| Seed | 42 | 666 | 1024 |
|------|-------|-------|-------|
| 42 | 100.0 | – | – |
| 666 | 46.67 | 100.0 | – |
| 1024 | 47.82 | 47.72 | 100.0 |

Table 7: The tree edit similarity (%) matrix between constructed tree pairs.

seeds, and then evaluate the overall performance of our model across all constructed trees. The results are reported in Table 7 and Table 5. While the relation trees exhibit some differences, the overall performance of our model remains consistently stable across different trees.

A.5 Quality of Hierarchical Relation Tree

To further assess the quality of our hierarchical relation tree, we provide both qualitative and quantitative evidence.

Qualitative Evidence. We conduct a case study to analyze the semantic coherence in intermediate nodes of the constructed tree. A partial view of the constructed tree is presented in Figure 9. Due to space limitations, only a representative subset of the full tree is shown.

For the example in Figure 9, we adopt {"domain", "entity types"} as the criterion set. As shown in the partial tree structure, using the "domain" criterion, the valid relations are first grouped into 12 high-level relation categories, such as "pol-

| Statistic | Value |
|--------------------|-------|
| Tree Depth | 5 |
| #Node at Level 0 | 1 |
| #Node at Level 1 | 2 |
| #Node at Level 2 | 12 |
| #Node at Level 3 | 136 |
| #Node at Level 4 | 675 |
| #Leaf Node | 676 |
| #Intermediate Node | 149 |
| #Child Node (Avg.) | 5.50 |

Table 8: The statistics of our hierarchical relation tree.

itics", "finance", and "healthcare". Within each domain, we further organize the relations according to the "entity types" criterion, resulting in multiple domain-specific subcategories. For instance, the domain "technology" contains 11 entity types, such as "object-software", "software-user", and "person-organization". At the last level, each predefined relation (e.g., "software engine" or "GUI toolkit or framework") is linked to its corresponding tree nodes (e.g. "object-software"). By progressively narrowing the semantic scope from general concepts to fine-grained relations, the resulting tree forms a reasonable hierarchical structure that helps the LLM to conduct hierarchical classification.

Quantitative Evidence. To provide the explicit quantitative evidence, we follow Hao et al. (2023); Wang et al. (2025) and conduct an automated semantic evaluation using an advanced LLM GPT-4o. We iterate through all parent-child pairs in our tree and query GPT-4o with the prompt:

Prompt Template for Coherence Evaluation

Is the relation node "[CHILD_NODE]" coherent with the node "[PARENT_NODE]"?
Output Yes/No only.

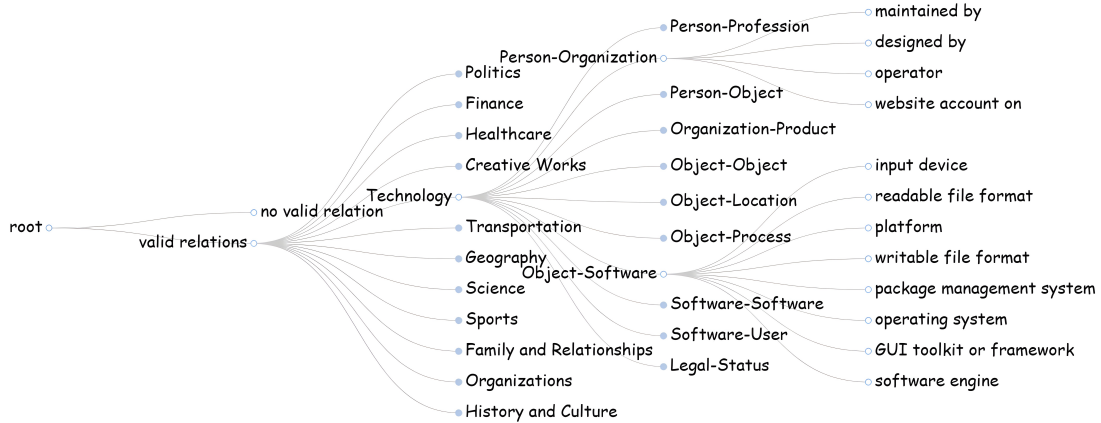


Figure 9: Visualization of a partial hierarchical relation tree.

| | Coherence (%) |
|-------------|---------------|
| Level 1 → 2 | 82.52 |
| Level 2 → 3 | 80.27 |
| Level 3 → 4 | 83.42 |

Table 9: The semantic coherence scores for each pair of adjacent tree levels.

We use the average probability of the “Yes” token output by GPT-4o as the *coherence* score, and evaluate coherence for each pair of adjacent tree levels.

As shown in Table 9, every pair of adjacent tree levels achieves a high coherence score, demonstrating that our hierarchical relation tree is not only effective for downstream performance but also linguistically interpretable and meaningful, validating the semantic quality of intermediate nodes.

B Baselines

We primarily compare our model with two categories of baselines:

(1) *Cross-Document RE Baselines*. **End-to-End** (Yao et al., 2021) is an Encoder-only model equipped with a selective attention mechanism to aggregate relation representations. **ECRIM** (Wang et al., 2022) introduces a cross-path entity relation attention mechanism to model the interaction among different text paths. **KXDocRE** (Jain et al., 2024a) enriches input text with domain knowledge to enhance relation representations. **REIC** (Na et al., 2024) designs a reinforcement learning-based sentence selector to identify relation evidence. **NEPD** (Yue et al., 2024) integrates entity graph encoding with ECRIM and calibrates the prediction distribution.

(2) *LLM-based Hierarchical Text Classification (HTC) Baselines*. Since our model involves hierarchical classification during inference, we also reproduce several representative LLM-based HTC baselines for cross-document RE to provide a comprehensive comparison. **Rs-ICL** (Chen et al., 2024) uses a hierarchy-aware indexer to retrieve demonstrations for in-context learning in HTC. **DFS-L** (Yu et al., 2022) and **BFS-L** (Huang et al., 2022; Jain et al., 2024b) train models to perform hierarchical classification by converting label hierarchies into sequences following depth-first and breadth-first search orders, respectively.

In addition to the above baselines, we also compare our model with a baseline referred to as **Vanilla**, which is fine-tuned to directly select the target relation from the full predefined relation set given the input instance.

C Implementation Details

We choose LLaMA-3.1-8B-Instruct (Meta, 2024) and GPT-4o as \mathcal{M}_1 and \mathcal{M}_2 , respectively. During training, we adopt an AdamW optimizer (Loshchilov and Hutter, 2019) with a learning rate of $5e-5$ and a total batch size of 32. Our model is trained for 6,400 steps using LoRA (Hu et al., 2021) with $r = 64$ and $\alpha = 128$. All experiments are conducted on 4 NVIDIA A100 80G GPUs. To ensure fair comparisons, we adopt the document-context filter of ECRIM (Wang et al., 2022) to preprocess all text paths.

D The *w/o* LTC Variant

Different from the level-wise pipeline of tree construction, in this variant, we first prompt the LLM \mathcal{M}_2 with predefined relations to directly generate

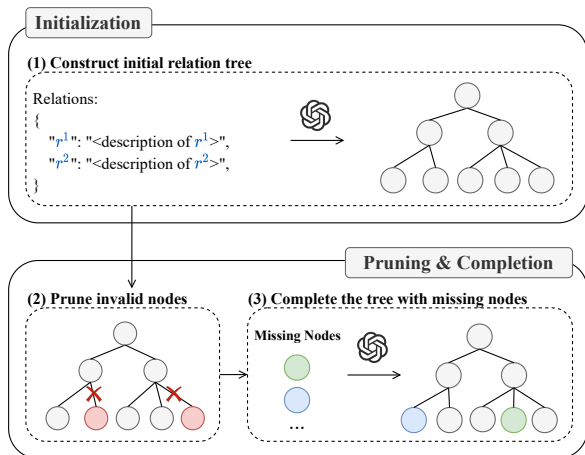


Figure 10: The tree construction pipeline for the *w/o* LTC variant.

a hierarchical relation tree in JSON format. Then, we ensure the completeness and validity of this tree step by step, as shown in Figure 10. Concretely, we construct the hierarchical relation tree of the *w/o* LTC variant via the following steps.

Step 1: We begin by concatenating the names and descriptions of all predefined relations into a JSON string, and then input it into \mathcal{M}_2 to cluster and summarize the relations, producing an initial hierarchical relation tree.

Step 2: However, since LLMs are prone to hallucinations, the initial tree often contains invalid relations that are not involved in the predefined relations. For the invalid relations, we simply remove them from the relation tree.

Step 3: Besides, some predefined relations may be absent in the initial relation tree. For each missing relation, we prompt \mathcal{M}_2 to place it at a suitable position within the current relation tree.

Following the above pipeline, we can easily obtain an LLM-generated hierarchical relation tree. Nevertheless, since the main tree structure is generated by \mathcal{M}_2 in a single run, the resulting trees often exhibit limited reasonableness. This inherent limitation accounts for its worse performance compared with our level-wise pipeline.

E Error Types in Error Propagation

To gain deeper insights into how errors propagate through the hierarchical classification process, we categorize error cases into two primary types: “Wrong Parent” and “Sibling Confusion”. The detailed error transfer statistics are summarized in Table 10. As shown in the table, PtV greatly reduces errors of the “Wrong Parent” type. Since an in-

| Level | Model | %CP | %WP | %SC |
|-------|---------------------|-------|-------|-------|
| 1 | HCRE <i>w/o</i> PtV | 34.64 | 0.00 | 65.36 |
| | HCRE (Ours) | 72.67 | 0.00 | 27.33 |
| 2 | HCRE <i>w/o</i> PtV | 32.13 | 65.36 | 2.51 |
| | HCRE (Ours) | 68.92 | 27.33 | 3.75 |
| 3 | HCRE <i>w/o</i> PtV | 31.31 | 67.87 | 0.82 |
| | HCRE (Ours) | 64.86 | 31.08 | 4.06 |
| 4 | HCRE <i>w/o</i> PtV | 28.07 | 68.69 | 3.24 |
| | HCRE (Ours) | 55.20 | 35.14 | 0.67 |

Table 10: The error transfer details for HCRE *w/o* PtV and HCRE. Note that **CP**, **WP**, and **SC** refer to “Correct Prediction”, “Wrong Parent”, and “Sibling Confusion”, respectively.

correct parent node almost guarantees an incorrect prediction, reducing this type of error effectively alleviates error propagation during hierarchical classification.

F Bag-level Evaluation

To further align the evaluation protocols with prior works, we perform bag-level evaluation by applying a simple majority voting mechanism to aggregate the path-level results into bag-level (entity-pair-level) predictions. Specifically:

1. The LLM independently generates a relation prediction for every text path within a bag. The confidence scores of predicted relations are also recorded for tie-breaking.
2. For each bag, the most frequent positive relation is selected as the bag-level prediction.
3. If multiple relations are tied for the highest frequency, we choose the relation with the highest confidence as the bag-level prediction.
4. If all paths within the bag are predicted as NA, the bag-level prediction is NA.

Notably, the bag-level evaluation is essentially another form of path-level evaluation, requiring no additional model training or inference. Therefore, the two evaluation protocols are fundamentally consistent.

As shown in Table 11, HCRE still significantly outperforms the second-best baseline (RoBERTa+NEPD) and Vanilla at bag-level evaluation. This is consistent with our path-level eval-

| Model | Closed | | Open | |
|--------------|--------------------|---------------------|--------------------|---------------------|
| | bag-level micro F1 | bag-level binary F1 | bag-level micro F1 | bag-level binary F1 |
| RoBERTa+NEPD | 54.17 | 72.04 | 50.39 | 64.38 |
| Vanilla | 47.53 | 52.47 | 47.67 | 54.49 |
| HCRE (Ours) | 64.82 | 86.18 | 58.98 | 79.90 |

Table 11: Bag-level evaluation results on CodRED under both closed and open settings.

| Setting | Model | Dev | | | | Test | | Avg. |
|---------|-------------------------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| | | F1 | AUC | P@500 | P@1000 | F1 | AUC | |
| Closed | End-to-End (Yao et al., 2021) | 51.26 | 47.94 | 62.80 | 51.00 | 51.02 | 47.46 | 49.42 |
| | ECRIM (Wang et al., 2022) | 61.12 | 60.91 | 78.89 | 60.17 | 62.48 | 60.67 | 61.30 |
| | KXDocRE (Jain et al., 2024a) | 64.97 | 64.30 | – | – | 66.30 | 65.55 | 65.28 |
| | REIC (Na et al., 2024) | 63.47 | 66.41 | 80.20 | 63.50 | 65.02 | 65.88 | 65.20 |
| | NEPD (Yue et al., 2024) | 63.63 | 65.01 | 77.84 | 64.03 | 64.41 | 66.23 | 64.82 |
| | HCRE (Ours) | 65.36 | 67.06 | 80.40 | 64.60 | 66.91 | 64.40 | 65.93 |
| Open | End-to-End (Yao et al., 2021) | 47.23 | 40.86 | 59.00 | 46.30 | 45.06 | 39.05 | 43.05 |
| | KXDocRE (Jain et al., 2024a) | 56.70 | 55.20 | – | – | 57.93 | 57.12 | 56.74 |
| | NEPD (Yue et al., 2024) | 54.29 | 54.92 | 68.66 | 53.84 | 56.68 | 55.87 | 55.44 |
| | HCRE (Ours) | 58.15 | 55.79 | 70.60 | 57.50 | 60.85 | 56.34 | 57.78 |

Table 12: Experiment results on conventional metrics under both closed and open settings. **F1** denotes maximum F1, and **Avg.** scores are computed based on F1 and AUC only. Baseline results are taken from their respective original papers.

uation, demonstrating that our evaluation ensures fairness and practical usability.

G Experiments on Conventional Metrics

Following prior work in cross-document RE (Yao et al., 2021; Wang et al., 2022; Yue et al., 2024), we evaluate our model on CodRED using conventional metrics, including P@K, AUC, and maximum F1. We compare HCRE with several representative cross-document RE models: End-to-End (Yao et al., 2021), ECRIM (Wang et al., 2022), KXDocRE (Jain et al., 2024a), REIC (Na et al., 2024), and NEPD (Yue et al., 2024). Details of these baselines are provided in Appendix B. For HCRE, to compute the conventional score-based metrics, we traverse the relation tree, obtain the score for each node and aggregate them to produce the relation score distribution.

As shown in Table 12, HCRE consistently outperforms all baselines across the conventional metrics under both closed and open settings. Furthermore, we submit the test results to the official competition leaderboard, where our model obtains 66.91 and 60.85 F1 points under the closed and

open settings, respectively.

H Experiments on Different Backbones

Due to the widespread use of LLaMA-3.1-8B-Instruct in the research community, we use it as our primary backbone for experiments. To examine the generality of HCRE across backbone architectures and model scales, we further report the results of experiments using Qwen2.5-0.5B-Instruct, Qwen2.5-7B-Instruct (Yang et al., 2024), and Gemma2-9B-Instruct (Gemma, 2024). As shown in Table 13, HCRE consistently yields performance gain, suggesting our PtV inference strategy is robust to both backbone architecture and model scale.

I Cross-Dataset Generalization of HCRE

To further validate the generalization of HCRE across different datasets, we examine the performance of HCRE on DocRED (Yao et al., 2019), a popular document-level RE benchmark.

Following the prior studies in document-level RE (Zhang et al., 2022; Zhang et al., 2023a; Zhang et al., 2023b; Zhang et al., 2025a), we adopt F1 and Ign F1 as our evaluation metrics. We compare

| Backbone | Model | Closed | | Open | |
|-----------------------|-------------|--------------|--------------|--------------|--------------|
| | | micro F1 | binary F1 | micro F1 | binary F1 |
| Qwen2.5-0.5B-Instruct | Vanilla | 18.63 | 26.41 | 7.00 | 10.15 |
| | HCRE (Ours) | 25.83 | 44.10 | 21.00 | 37.63 |
| Qwen2.5-7B-Instruct | Vanilla | 36.10 | 41.16 | 14.20 | 16.59 |
| | HCRE (Ours) | 37.29 | 52.35 | 34.40 | 53.43 |
| Gemma2-9B-Instruct | Vanilla | 37.34 | 41.77 | 14.53 | 17.04 |
| | HCRE (Ours) | 41.74 | 53.39 | 28.94 | 38.75 |

Table 13: Experiment results on CodRED with different backbones.

| Model | Dev | | Test | |
|---|--------------|--------------|--------------|--------------|
| | F1 | Ign F1 | F1 | Ign F1 |
| <i>LLM-based Doc-level RE Baselines</i> | | | | |
| AutoRE | 50.37 | 48.79 | 50.35 | 48.58 |
| EP-RSR | 56.17 | 53.74 | 56.47 | 53.22 |
| <i>Ours</i> | | | | |
| Vanilla | 61.30 | 60.21 | 60.77 | 59.63 |
| HCRE | 62.80 | 61.22 | 61.57 | 60.18 |

Table 14: Experiment results on DocRED.

| Model | Input Tok. | #LLM Calls | Latency |
|-------------|------------|------------|---------|
| Vanilla | 1,499.76 | 40,740 | 0.21s |
| HCRE (Ours) | 575.75 | 152,663 | 0.29s |

Table 15: Comparison of computation efficiency between Vanilla and HCRE. **Input Len.** refers to the average input tokens.

HCRE with the Vanilla baseline and two representative LLM-based document-level RE models: 1) **AutoRE** (Xue et al., 2024) proposes a LLM-based Relation-Head-Facts paradigm that enables the LLM to extract relations without the need to perceive the full predefined relation set. 2) **EP-RSR** (Zhang et al., 2025a) introduces an Entity-Pair-Relation-Fact paradigm and enhances the relevance between candidate relations and target entities via an entity pair-level relation filtering method. As shown in Table 14, HCRE consistently surpasses these baselines on both development and test sets across all metrics, demonstrating strong cross-dataset generalization.

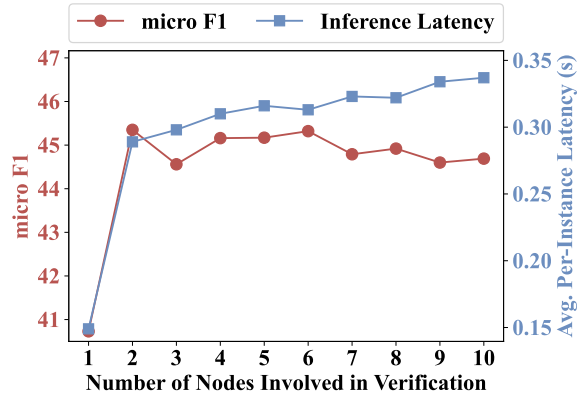


Figure 11: The micro F1 and per-instance inference latency using PtV with varying numbers of verification nodes.

J More Details about PtV

To further demonstrate our PtV inference strategy, we further describe its detailed procedure, analyze its computation efficiency, and conduct ablation experiments on the number of nodes involved in verification in this section.

J.1 Detailed Procedure

To describe the PtV inference strategy precisely, we present the complete PtV process in Algorithm 1.

J.2 Computation Efficiency

To analyze the computational efficiency of the PtV inference strategy, we compare the average per-instance input tokens, the number of LLM calls, and the average per-instance latency of Vanilla and HCRE on an NVIDIA A100 80G GPU. The statistics are presented in Table 15. Although HCRE triggers more LLM calls, its hierarchical classification paradigm substantially reduces the number of options during each inference, reducing the average number of input tokens by 61.6% (from 1,499.76

Algorithm 1 Prediction-then-Verification Inference Strategy

Input: LLM \mathcal{M}_1 , hierarchical relation tree \mathcal{T} , context c , head entity e_h , tail entity e_t , current level l , node predicted at $(l-1)$ -th level \hat{r}_{l-1} , and maximum PtV round M

Output: Final relation \hat{r}_l

$\mathcal{R}_l \leftarrow \mathcal{T}.\text{children_of}(\hat{r}_{l-1})$

$t \leftarrow 0$ // Number of PtV rounds

while True **do**

 // Step 1: Prediction step

$\hat{r}_{1\text{st}}, \hat{r}_{2\text{nd}} \leftarrow \mathcal{M}_1(c, e_h, e_t, \mathcal{R}_l)$

 // Step 2: Verification step

 // Sub-step 2.1: Replace nodes with their respective children

$\mathcal{R}_l^{v1} \leftarrow \mathcal{R}_l.\text{replace}(\hat{r}_{1\text{st}}, \mathcal{T}.\text{children_of}(\hat{r}_{1\text{st}}))$

$\mathcal{R}_l^{v2} \leftarrow \mathcal{R}_l.\text{replace}(\hat{r}_{2\text{nd}}, \mathcal{T}.\text{children_of}(\hat{r}_{2\text{nd}}))$

$\mathcal{R}_l^{v3} \leftarrow \mathcal{R}_l.\text{replace}(\hat{r}_{1\text{st}}, \mathcal{T}.\text{children_of}(\hat{r}_{1\text{st}})).\text{replace}(\hat{r}_{2\text{nd}}, \mathcal{T}.\text{children_of}(\hat{r}_{2\text{nd}}))$

 // Sub-step 2.2: Verify $\hat{r}_{1\text{st}}$

$\hat{r}^{v1}, _ \leftarrow \mathcal{M}_1(c, e_h, e_t, \mathcal{R}_l^{v1})$

$\hat{r}^{v2}, _ \leftarrow \mathcal{M}_1(c, e_h, e_t, \mathcal{R}_l^{v2})$

$\hat{r}^{v3}, _ \leftarrow \mathcal{M}_1(c, e_h, e_t, \mathcal{R}_l^{v3})$

if $\mathbb{1}_{\hat{r}^{v1} \in \mathcal{T}.\text{children_of}(\hat{r}_{1\text{st}})} + \mathbb{1}_{\hat{r}^{v2} = \hat{r}_{1\text{st}}} + \mathbb{1}_{\hat{r}^{v3} \in \mathcal{T}.\text{children_of}(\hat{r}_{1\text{st}})} \geq 2$ **then**

$\hat{r}_l \leftarrow \hat{r}_{1\text{st}}$

break

else

$\mathcal{R}_l.\text{remove}(\hat{r}_{1\text{st}})$

end if

 // Exceeds max round limitation

$t \leftarrow t + 1$

if $t > M$ **then**

$\hat{r}_l \leftarrow \hat{r}_{1\text{st}}$

break

end if

end while

return \hat{r}_l

to 575.75). Given the $\mathcal{O}(N^2)$ computation complexity of Transformer (Vaswani et al., 2023), this reduction accelerates the prefiling phase of LLM inference. Consequently, HCRE achieves latency on par with Vanilla.

J.3 Extending Verification to Top- k Nodes

To analyze the impact of the number of nodes involved in verification, we conduct an ablation experiment that extends verification to top- k nodes, with k ranging from 1 to 10.

As shown in Figure 11, the top-2 setting achieves a significant improvement over top-1, indicating that considering more than one candidate helps mitigate error propagation more effectively. However, further increasing k beyond 2 does not bring more performance gain while adding computational overhead. Therefore, our design choice maximizes the

performance-efficiency trade-off.