Do not be greedy, THINK TWICE: Sampling and Selection for Document-level Information Extraction

Anonymous ACL submission

Abstract

Document-level Information Extraction (DocIE) aims to produce an output template with the entities, relations and events of interest occurring in the given document. Standard practices include prompting decoder-only LLMs using greedy decoding to avoid output variability. Rather than treating this variability as a limitation, we show that sampling can produce substantially better solutions than greedy decoding, especially when using reasoning models. We thus propose THINKTWICE, a sampling and selection framework in which the LLM generates multiple candidate templates for a given document, and a selection module chooses the most suitable one. We introduce both an unsupervised method that exploits agreement across generated outputs, and a supervised selection method using reward models trained on labeled DocIE data. To address the scarcity of golden reasoning trajectories for DocIE, we propose a rejection-sampling-based method to generate silver training data that pairs output templates with reasoning traces. Our experiments show the validity of unsupervised and supervised THINKTWICE, consistently outperforming greedy baselines and the supervised state-of-the-art. Our code is publicly available.[1]

1 Introduction

Information Extraction (IE) aims to convert natural occurring text into structured representations by identifying entities, attributes, and relations of interest. Traditionally, IE was conceived as a document-level task (muc, 1992; Ji et al., 2010), but with the popularization of statistical and machine-learning methods, the field shifted towards sentence-level datasets (Walker et al., 2006; Zhang et al., 2017; Grishman, 2019), which were simpler and easier to annotate. Still, many real-world scenarios require reasoning over entire documents, where relevant information is scattered across multiple sentences or sections. Document-level IE (DocIE) addresses this challenge by jointly extracting and aggregating information at the document scope, typically producing structured templates that depend on long-range context and cross-sentence interactions. With the rapid development of Large Language Models (LLMs), and their longer context, it has now become more natural to work on the document-level (Gatto et al., 2025) instead of analyzing each sentence independently and aggregating the results (Chen et al., 2023).

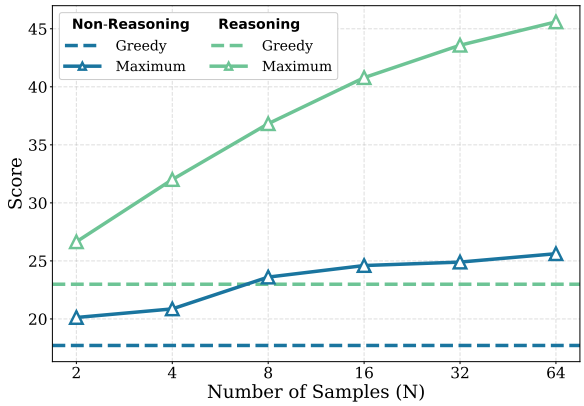


Figure 1: Sampling results on MUC-4, *maximum* reports the results of oracle selection among generated samples.

In contrast to encoder-based models, decoder-only LLMs can be directly prompted with some annotation guidelines to produce the annotations for a given document, enabling zero- and few-shot inferences without task-specific architectural modifications or fine-tuning. The generative nature of LLMs, however, presents a challenge when it comes to evaluating their performance, as each time they are prompted, a different solution could be generated. In order to produce reproducible comparisons, researchers leverage greedy decoding strategies to ensure the same output template is produced for a given pair of guidelines and docu-

[1] Upon acceptance

ment (Sainz et al., 2024; Li et al., 2023). Producing only a single answer—even though the most probable is chosen at each time—may not produce the best answer, hindering the evaluation and hiding the real performance of a given model. As we show in Figure 1, there exists an increasing gap between the best performing answer and the greedy decoded answer when the amount of samples increases, which further intensifies with reasoning models. This phenomenon is not tailored to IE only, but recently studied under the Test-Time Scaling (TTS) term, particularly for mathematical reasoning tasks (Wu et al., 2025; Snell et al., 2025).

The application of TTS techniques to IE remains largely unexplored. To address this gap, we propose THINKTWICE, a sampling and selection framework in which an LLM generates multiple candidate extractions for a document, and a dedicated selection function identifies the most reliable output. We show that greedy decoding fails to fully exploit the capabilities of LLMs, motivating the use of sampling-based decoding strategies. We further analyze how this effect manifests in both reasoning-oriented and non-reasoning models.

As our main **contributions**, we introduce a generation and selection framework, as well as two new selection strategies tailored to IE: an unsupervised **F1 Voting** method and a supervised **reward-based** selection strategy. Both substantially outperform standard TTS baselines and establish new supervised state-of-the-art results in document-level IE. In the supervised setting, we overcome the lack of golden trajectories for DocIE with a rejection-sampling approach for generating high-quality reasoning traces, which are used to produce silver training data. Our experimental results lead to the following key **findings**: 1) Reasoning models consistently outperform standard LLMs in document-level IE. 2) The proposed unsupervised sampling and selection framework yields consistent gains over the greedy decoding and standard TTS baselines in zero-shot settings. 3) Training on high-quality silver reasoning trajectories paired with output templates achieves state-of-the-art results in document-level IE when combined with a supervised selection strategy, both in the monolingual and cross-lingual settings.

2 Related Work

Document-level Information Extraction. Information Extraction, while originally conceived as a document-level task, was long constrained to the sentence level due to the limitations of earlier systems. With recent advances in the field of Natural Language Processing (NLP), however, DocIE has regained significant attention, on problems such as argument linking (Li et al., 2021), *N*-ary relation extraction (Jain et al., 2020), and, our primary focus, template extraction (Huang et al., 2021; Chen et al., 2023). This resurgence has led researchers to revisit classic datasets such as MUC (Grishman and Sundheim, 1996) and to develop new resources like BETTER (Mckinnon and Rubino, 2022) and the multilingual extension of MUC-4, MultiMUC (Gantt et al., 2024). The state-of-the-art in these datasets is ITERX (Chen et al., 2023), which formulates template extraction as a Markov decision process, generating templates iteratively using an imitation-learned policy, and introduces a new evaluation metric for MUC-4, CEAF-RME. However, ITERX relies on encoder-based backbones, which are unsuitable for zero-shot settings and often require truncating the document due to their shorter sequence length, resulting in lower performance. Motivated by these limitations, recent work has begun to explore the use of LLMs for Information Extraction.

LLMs for IE. The adoption of large language models in IE has grown substantially in recent years (Zhang et al., 2025c). In addition to achieving state-of-the-art performance in supervised settings (Wadhwa et al., 2023; Ding et al., 2024), LLMs have demonstrated strong zero-shot IE capabilities by leveraging dataset-specific guidelines (Sainz et al., 2024). This flexibility extends the Unified Information Extraction paradigm and opens new research directions. LLMs can also use in-context learning examples to improve their performance without further training (Wan et al., 2023; Zhang et al., 2024) or be decoded using constraint decoding to create always valid outputs (Sakota and West, 2025). However, the inherently stochastic nature of LLMs leads to variability across generations. Although greedy decoding is commonly used to reduce this instability, our results suggest that it is not the most effective decoding strategy. Motivated by this observation, we explore a sampling and selection approach inspired by recent test-time scaling methods.

Reasoning and Test-time Scaling. With the release of OpenAI's O1 model (Jaech et al., 2024), reasoning-focused models have become a central

topic of interest in the field. At the same time, Test-Time Scaling (TTS) has also gained attention (Zhang et al., 2025a). This technique increases computational effort during inference to achieve better results. To support this approach, several methods have emerged, such as Majority Voting (Lightman et al., 2023), which selects the most frequent answer among multiple sampled responses, and the use of reward models (Wu et al., 2025) to identify the highest-quality output. However, most existing work in this area has primarily focused on math and reasoning-intensive benchmarks (Zhang et al., 2025a). The methodology proposed in this work draws inspiration from these TTS approaches while adapting them to the IE task.

3 Methodology

While standard approaches rely on greedy decoding to produce a single output, Figure 1 shows that the highest-scoring outputs are not always obtained with this method. Generating multiple candidate templates and applying a selection strategy could potentially yield substantially better results. Moreover, the gap between greedy decoding performance and the maximum achievable score **is even more pronounced for reasoning models**. Training reasoning models, however, requires good-quality reasoning traces along with the gold standard templates, which are not available for our current datasets.

To address these issues, we propose THINKTWICE, a sampling and selection architecture, where the model first produces several outputs for a given input, and the selector then decides the final output. In addition, in order to properly train a model based on reasoning LLMs, we propose a rejection-sampling based method to generate good-quality reasoning traces for each training sample.

3.1 Task Representation

Following previous works on zero-shot IE, we adopted the annotation guidelines aware prompting strategy (Sainz et al., 2024) as our task representation. Formally, for a given document-level IE dataset $\mathcal{D}$, each example is a triplet (g, x_i, t_i) where g are the annotation guidelines for the given dataset, x_i is the text (or document) to analyze, and, t_i is the template with the structured information associated with x_i text. Current LLMs require following a predefined chat template that specifies how each message in a conversation is encoded and identifies the author role of each message (i.e., system, user, or assistant). The final instance representation is obtained by giving the annotation guidelines g encoded as Markdown text within the system prompt, and each text document x_i in the user prompt. The output template t_i is expected to be generated by the model under the assistant role following a predefined JSON schema. More information about the prompts we used can be found in Appendix A.

3.2 Inference

The inference of our method is described in Figure 2. Briefly, THINKTWICE consist in performing a two-step inference where, first, a model π_θ is prompted with task guidelines g and a document x_i to produce N possible template candidates $T_i = \{\hat{t}_{i,1}, \ldots, \hat{t}_{i,N}\}$. On a second step, a selector strategy $\mathcal{S}(g, x_i, T_i)$ is applied over the generated options to select the preferred output candidate. This selector $\mathcal{S}$ can be either an unsupervised or supervised method, as described later in Section 3.4.

Producing complex document-level IE output templates can be challenging, particularly in zero-shot scenarios. To that end, we sampled the candidates applying a constrained decoding strategy where the model was forced to follow a predefined JSON schema. In the case of reasoning models, we first let the model generate the reasoning traces, and then apply the same constrained decoding strategy to force the output meet the requirements. The JSON schemas leveraged during the inference are described in Appendix B.

3.3 Model training

Training standard instructed LLMs for the task can be accomplished by simply optimizing the cross-entropy loss for each training example (g, x_i, t_i) in the dataset $\mathcal{D}$. However, in order to fine-tune a reasoning LLMs, we also require good quality reasoning traces r_i tailored to each training example. To that end, we propose a rejection-sampling based method to generate those reasoning traces that we require to train our model. Figure 2 illustrates the overall workflow of the proposed iterative approach.

For each input $x_i \in \mathcal{D}$, we first generate N synthetic reasoning–template pairs $(\hat{r}_{i,j}, \hat{t}_{i,j})$ by sampling a reasoning model π_θ conditioned on the instance (g, x_i). We then rank these candidates using a scoring function $\phi(\hat{t}_{i,j}, t_i)$ based on the evaluation metric of the dataset (cf. Section 4.2), which compares each synthesized template to the

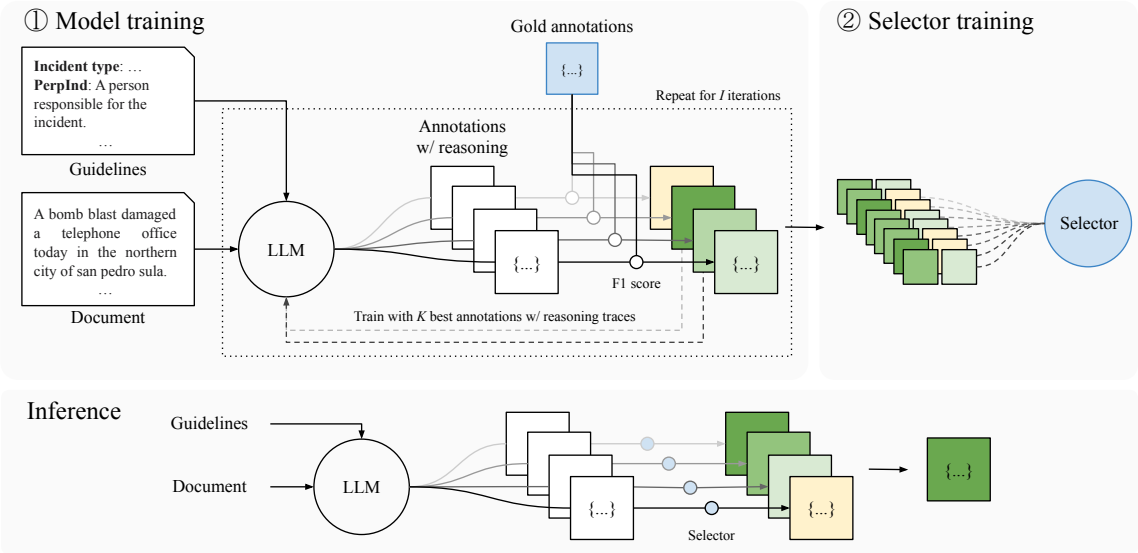


Figure 2: THINKTWICE architecture, with the inference process at the bottom. The supervised option includes two steps: ① The iterative procedure to generate the silver dataset with trajectories and to fine-tune the reasoning model; ② Training the selector with silver preference data that include trajectories.

gold annotation t_i, and retain the top K examples while discarding the remaining $N - K$. Using this ranking, we construct a silver dataset $\hat{\mathcal{D}} = \{(g, x_1, \hat{r}_{1,1}, \hat{t}_{1,1}), \ldots, (g, x_{|\mathcal{D}|}, \hat{r}_{|\mathcal{D}|,K}, \hat{t}_{|\mathcal{D}|,K})\}$, which is subsequently used to train the LLM. This iterative procedure is repeated until the maximum selection score no longer improves across successive iterations. After convergence, we perform a final aggregation step that collects candidates from the preceding iteration, selects the top R reasoning–template pairs, and fine-tunes a newly reinitialized model on this resulting dataset.

3.4 Selectors and their training

Selecting a single template $\hat{t}_{i,j}$ from a set of candidate templates $T_i = \{\hat{t}_{i,1}, \ldots, \hat{t}_{i,N}\}$ can be approached in multiple ways, either unsupervised or supervised. Standard TTS approaches typically rely on **Majority Voting** (Wang et al., 2023; Chen et al., 2024) to select the most frequent candidate. However, while effective for math-related tasks, this strategy often fails for IE, where complex outputs lead to the most common prediction being empty. To address this limitation, we **propose both an unsupervised and supervised selector**.

F1 Voting. This is an alternative to Majority Voting, specifically designed for IE, which uses F1-score-related metrics[2] as a continuous similarity measure instead of the binary match signal used in Majority Voting. Each candidate template is assigned a score based on its similarity to the remaining candidates. The candidate with the highest aggregate score is selected. Formally,

$$\hat{t}_i = \arg\max_{\hat{t}_{i,j} \in T_i} \sum_{k=1}^{N} \phi(\hat{t}_{i,j}, \hat{t}_{i,k}) \,, \qquad (1)$$

where ϕ denotes the scoring function (the F1-score in our case), T_i is the set of templates generated for instance i, and $N = |T_i|$.

Supervised selector (Reward model). Training a reward model requires outputs of varying quality, thus, we propose to train the selector based on the silver instances generated for model training, including poor-quality outputs, together with the gold annotations. As shown in Figure 2, the discarded noisy templates produced during training can be leveraged together with the selected examples to learn a preference-based reward model, which is subsequently used to rank candidate templates and select the most suitable one. Formally, we can construct a preference dataset $\mathcal{P}$ based on silver $T_i = \{\hat{t}_{i,1}, \ldots, \hat{t}_{i,N}\}$ and gold $\{t_i\}$ annotations.

$$\mathcal{P} = \{(g, x_i, t_i^c, t_i^r) \mid t_i^c, t_i^r \in T_i \cup \{t_i\}, \; \phi(t_i^c, t_i) > \phi(t_i^r, t_i)\} \qquad (2)$$

where t_i^c and t_i^r represent the selected and rejected solutions, respectively. Additionally, we can compute an informed margin m by considering the difference in the score between t_i^c and t_i^r:

[2] See Section 4.2 for the metric used in each dataset.

$$m = \lambda \cdot (\phi(t_i^c, t_i) - \phi(t_i^r, t_i)) \quad (3)$$

and, λ is a scaling factor which we set to 3 as suggested by Touvron et al. (2023). We can then train the reward model r_θ by optimizing the Bradley-Terry formula with a margin loss function (Bradley and Terry, 1952; Touvron et al., 2023):

$$\mathcal{L}_{\text{BT}} = \sum_{(g,x_i,t_i^c,t_i^r,m_i)\in\mathcal{P}} -\log\Big(\sigma\big(r_\theta(g,x_i,t_i^c) - r_\theta(g,x_i,t_i^r) - m_i\big)\Big) \quad (4)$$

Finally, we can use the trained reward model r_θ to rank and select the correct template $\hat{t}_i$ among the candidates as follows:

$$\hat{t}_i = \arg\max_{\hat{t}_{i,j}\in T_i} r_\theta\big(g, x_i, \hat{t}_{i,j}\big)\,, \quad (5)$$

4 Experimental Setup

4.1 Models

We leveraged two open-source model families as our backbone models: Llama and Qwen. In addition, we used an embedding model as initialization for our supervised selector approach.

Llama 3.3 70B Instruct (Grattafiori et al., 2024) is a well-established open-source LLM that achieves strong performance across a wide range of tasks. We use this model as our non-reasoning Llama-based LLM backbone.

DeepSeek Distill Llama R1 70B is a distilled version of the DeepSeek R1 600B model, trained using Llama 3.3 70B as base (DeepSeek-AI, 2025). Since no official reasoning-oriented models have been released within the Llama family, we adopt this model as our Llama-based reasoning LLM backbone.

Qwen3 32B is the largest dense model in the Qwen 3 family and supports both non-reasoning (*no-think*) and reasoning (*think*) inference modes (Yang et al., 2025). This model is particularly well suited to our experiments, as it allows inference with and without reasoning traces while achieving performance comparable to Llama 70B.

Qwen3 8B Embedding (Zhang et al., 2025b) is a pretrained embedding model that achieves state-of-the-art performance across a wide range of downstream tasks. We build our reward model on top of this backbone, leveraging its rich semantic representations for reward modeling. To adapt the model to our setting, we add a feed-forward projection head that maps the embedding representation to a single scalar value, which is used as the reward score during inference.

For all experiments, we fix the number of sampled generations per instance to $\mathbf{N} = \mathbf{64}$, which offers a favorable trade-off between exploration and computational cost. Additional hyperparameter details are provided in Appendix D.

To evaluate the stability of our methods and improve the reliability of the reported results, we perform bootstrapping using three different random seeds over the sampled generations. We report both the mean and standard deviation across these runs.

4.2 Evaluation Datasets

MUC-4 and MultiMUC: the MUC-4 dataset (muc, 1992) contains 1,700 English news articles on geopolitical conflicts, annotated with structured templates covering six event types: Attack, Arson, Bombing, Murder, Robbery, and Forced Work Stoppage. MUC-4 has served as a foundational benchmark for template-based event extraction. Regarding the evaluation metrics, we report results using the **CEAF-RME** metric proposed by Chen et al. (2023). Building on this benchmark, MultiMUC (Gantt et al., 2024) extends the dataset to a multilingual setting by providing parallel annotations in Arabic, Farsi, Korean, Russian, and Chinese alongside the original English corpus.

BETTER Granular (Mckinnon and Rubino, 2022) dataset further increases task difficulty by covering a broader range of domains and event types, including protests, epidemics, natural disasters, acts of terrorism, corruption incidents, and human migration. While it also focuses on six complex event categories, BETTER Granular features substantially longer documents and more intricate templates with finer-grained argument structures, being much more complex than the MUC-4 dataset. Due to the length of the documents and annotation guidelines, we used BETTER Granular exclusively for zero-shot evaluations. While we used the official scorer to compute the final results, we implemented a simplified F1 scorer in order to use it with the F1 Voting selector.

4.3 Baselines

We compared THINKTWICE against our baseline, **greedy decoding**, and previous zero-shot and supervised state-of-the-art methods:

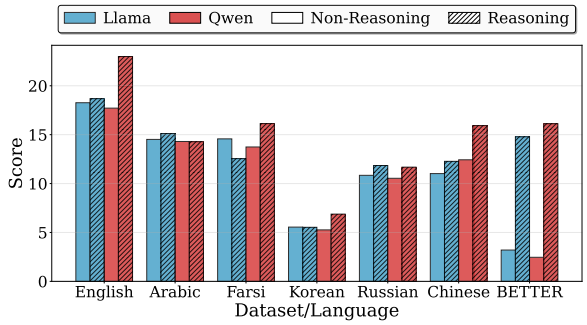


Figure 3: Zero-shot results for the reasoning and non-reasoning baselines of two LLMs on MUC (English), MultiMUC and BETTER.

Method	Selector	MultiMUC	BETTER
GPT $_{5.5}$	✗	19.84	27.95
Greedy $_{\text{Llama R1}}$	✗	12.67	14.78
THINKTWICE $_{\text{Llama R1}}$	Majority	13.83±0.66	10.51±2.30
	F1 Voting	14.41±0.33	15.29±0.88
	(oracle)	31.77	34.08
Greedy $_{\text{Qwen 3}}$	✗	14.65	16.12
THINKTWICE $_{\text{Qwen 3}}$	Majority	16.18±0.89	16.26±0.82
	F1 Voting	**16.56**±0.37	**19.45**±0.16
	(oracle)	35.32	36.74

Table 1: Zero-Shot results on MultiMUC and BETTER. Greedy results are compared to two unsupervised selectors. Underline and **bold** indicate the best result among greedy and the selectors per model and across all models, respectively.

Zero-shot methods. Prior work on these datasets under zero- or few-shot settings is limited. Therefore, we evaluate and report **GPT$_{5.5}$** results in a zero-shot setting on the MUC-4, MultiMUC, and BETTER datasets for reference.

Supervised methods. Our main points of comparison for the supervised setting comes from the results reported by Chen et al. (2023). We compare our system against ITERX (Chen et al., 2023), a T5$_{\text{large enc}}$-based system and current state-of-the-art; GTT (Du et al., 2021), a BERT encoder based approach; and, TEMPGEN (Huang et al., 2021), a BART encoder-decoder based approach.

5 Results

5.1 To think, or not to think?

As depicted by the preliminary results shown in Figure 1, the reasoning models not only perform better, but also allow for bigger *maximum scores*[3]. To properly answer this question, we evaluated both Llama and Qwen3 models with and without reasoning on all of our testing datasets. Figure 3 shows the **zero-shot** performance of each model-inference combination on each specific dataset. We leveraged *greedy decoding*, i.e. we did not use any sampling nor selector on this initial analysis to focus only on the reasoning/non-reasoning comparison.

Do reasoning models perform better than their non-reasoning counterparts? As shown in the Figure 3, reasoning models perform consistently better than their non-reasoning counterparts, with only a few exceptions. Interestingly, the performance gap between reasoning and non-reasoning models is especially pronounced on the BETTER dataset, which involves more complex guidelines and templates. Manual analysis revealed that constrained decoding is insufficient to enable non-reasoning models to produce good-quality output templates. We hypothesize that the superior performance of reasoning models stems from their specialized training on long-form sequential dependencies, which facilitates more coherent, structured outputs in document-level IE tasks, specifically in the BETTER dataset. Therefore, based on these results, the following analyses will focus exclusively on **reasoning models**.

5.2 THINKTWICE as zero-shot

Following the zero-shot experiments, we analyze the impact of THINKTWICE with different **unsupervised** selection strategies. Table 1 compares greedy decoding, our proposed THINKTWICE framework, and GPT$_{5.5}$.

Does THINKTWICE improve zero-shot performance? Table 1 shows that sampling with an explicit selection strategy consistently outperforms greedy decoding. On MultiMUC, F1 Voting improves performance by roughly two F1 points across both backbones. Gains are even larger on the more challenging BETTER benchmark, where Majority Voting struggles with the increased template complexity, while F1 Voting yields consistent improvements over greedy decoding, exceeding three points for Qwen3. Additionally, F1 Voting exhibits lower variance in all datasets and models, indicating greater robustness than Majority Voting.

Overall, F1 Voting is the most effective unsupervised selection strategy, consistently outperforming greedy decoding across backbones and datasets.

[3]We compute the maximum attainable score by selecting the best generated output for each instance, which serves as an oracle upper bound for the selector.

Method	Selector	MUC-4
TEMPGEN $_{\text{BART large}}$	✗	28.30
GTT $_{\text{BERT base}}$	✗	32.30
ITERX $_{\text{T5-enc large}}$	✗	35.20
Greedy $_{\text{Llama R1}}$	✗	28.52
THINKTWICE $_{\text{Llama R1}}$	Majority	28.41$_{\pm 1.49}$
	F1 Voting	36.56$_{\pm 0.22}$
	Reward	41.11$_{\pm 0.48}$
	(oracle)	56.60
Greedy $_{\text{Qwen 3}}$	✗	35.97
THINKTWICE $_{\text{Qwen 3}}$	Majority	35.67$_{\pm 0.59}$
	F1 Voting	40.77$_{\pm 0.53}$
	Reward	**42.51**$_{\pm 0.72}$
	(oracle)	58.32

Table 2: Supervised results for THINKTWICE, greedy baseline and the state-of-the-art on MUC-4.

How good are our current selection strategies? In addition to the results obtained with the current selection strategies, Table 1 also reports the scores that could be obtained when a perfect selector (oracle) is used. Comparing the oracle results with $GPT_{5.5}$[4] we can see that much smaller model can surpass proprietary state-of-the-art models if combined with a good selector. However, we observe that the unsupervised selectors remain far below the oracle upper-bound. This suggests that **there is still significant room for improvement through a more effective selection strategy**. In the following sections, on an attempt to reduce the gap, we will analyze the results obtained by a supervised selection approach.

5.3 Supervised THINKTWICE

All the results reported previously were obtained by an out-of-the-box LLM, without further training for the task. In this section, we explore the impact of supervised data along with our proposed rejection sampling method, and, the impact of supervised data to train the selector. We focus on MUC-4 for training and evaluation, and explore cross-lingual transfer in the following subsection.

Does THINKTWICE improve results in a supervised scenario? Table 2 reports the results for our supervised experiments on the MUC-4 dataset. As can be seen, similar conclusions are drawn compared to the zero-shot results: (1) THINKTWICE improves over the greedy baseline by a large margin; and, (2) F1 Voting performs consistently better than Majority Voting in all cases. Interestingly, when data is used to train the model, Majority Voting improvements are marginal or even prejudicial whereas F1 Voting results remain consistent.

While the performance gains over previous state-of-the-art methods are substantial, one might attribute this improvement to the superior inherent capabilities of the Llama R1 and Qwen 3 compared to their predecessor models. Our results demonstrate that simply applying greedy decoding on modern models does not unlock their full potential, sometimes achieving performance below alternatives that leverage older and less capable LLMs. In contrast, performance improves significantly when sampling is coupled with an effective selection strategy. Nevertheless, our approach still faces an important limitation, as it is constrained by the absence of gold-standard reasoning traces for supervised fine-tuning. Utilizing a rejection-sampling based strategy, we successfully obtained gold-quality reasoning traces for only 67.77% and 71.46% of the training data for Llama R1 and Qwen 3, respectively. A deeper analysis of the rejection-sampling fine-tuning is reported in Appendix G.

Is the supervised selector effective? Motivated by the intuition that classification is inherently easier than generation, we investigate the impact of training a selector on the data collected during the iterative rejection-sampling process. Table 2 compares the supervised selector against the unsupervised approaches and the oracle upper bound (i.e., the maximum achievable score under perfect selection). The supervised selector consistently surpasses all unsupervised methods across both Llama and Qwen variants, with especially pronounced gains for the Llama R1 model. Incorporating the supervised selector yields further improvements for both models and establishes a new state-of-the-art for the task. Nevertheless, a substantial gap to the oracle remains. Although matching the oracle is likely unattainable in practice, the remaining margin suggests considerable room for improving upon our relatively simple supervised selector.

5.4 Cross-lingual transfer

Finally, we evaluate the cross-lingual generalization of THINKTWICE. We first assess whether an LLM fine-tuned only on English data can transfer the learned knowledge to other languages when paired with the unsupervised selector. We then analyze the cross-lingual transferability of the learned

[4] Single sample results are reported due to the lack of control over inference hyperparameters such as temperature and the resultant poor diversity (see Appendix F).

Method	Selector	English	Arabic	Farsi	Korean	Russian	Chinese	Average
GPT $_{5.5}$	✗	29.50	24.60	22.03	07.59	17.70	17.59	19.84
Gantt et al. (2024) $_{\text{Supervised}}$	✗	35.20	21.46	20.66	**23.91**	23.77	21.93	24.49
THINKTWICE $_{\text{Zero-shot}}$	F1 Voting	24.30$_{\pm 0.57}$	17.66$_{\pm 0.32}$	19.62$_{\pm 0.44}$	07.22$_{\pm 0.37}$	14.34$_{\pm 0.16}$	16.20$_{\pm 0.16}$	16.56$_{\pm 0.37}$
	Reward	33.46$_{\pm 0.70}$	26.05$_{\pm 0.24}$	26.80$_{\pm 0.21}$	08.71$_{\pm 0.53}$	20.66$_{\pm 0.25}$	25.47$_{\pm 0.74}$	23.53$_{\pm 0.50}$
THINKTWICE $_{\text{English FT}}$	F1 Voting	40.77$_{\pm 0.53}$	22.66$_{\pm 0.44}$	23.61$_{\pm 0.77}$	08.73$_{\pm 0.75}$	20.53$_{\pm 1.18}$	22.15$_{\pm 0.92}$	23.07$_{\pm 0.80}$
	Reward	**42.51**$_{\pm 0.72}$	**30.27**$_{\pm 0.49}$	**30.16**$_{\pm 0.66}$	09.62$_{\pm 0.14}$	**29.95**$_{\pm 1.23}$	**29.75**$_{\pm 0.75}$	**28.71**$_{\pm 0.74}$

Table 3: Cross-lingual transfer performance of THINKTWICE. The Qwen3-based LLM and reward model are trained exclusively on English data (MUC-4) and evaluated on multiple languages. Results are compared against state-of-the-art models trained on the target language (Gantt et al., 2024), as well as $GPT_{5.5}$ under a zero-shot setting.

reward selector. Finally, we compare against the best previously reported multilingual MUC results from Gantt et al. (2024), which were obtained by fine-tuning language specific models for the task.

Cross-lingual generalization of THINKTWICE. Table 3 presents the cross-lingual transfer results obtained with Qwen3 under both zero-shot and fine-tuned settings. In the zero-shot setting, performance drops substantially for languages other than English, with the largest degradation observed for Korean. Notably, $GPT_{5.5}$ exhibits a similar trend, further underscoring the persistent imbalance in multilingual capabilities across current language models. Fine-tuning on English data alone yields an average improvement of 6.51 F1 points across languages, demonstrating strong cross-lingual transfer despite the absence of target-language supervision. Although gains for Korean remain limited, most languages benefit consistently from English-only training.

The introduction of a supervised selector leads to even larger improvements, providing nearly 7 additional F1 points for the zero-shot model and 5.64 F1 points for the English fine-tuned model. Interestingly, combining both forms of supervision enables our models to outperform systems trained directly on language-specific data (Gantt et al., 2024). Finally, compared to $GPT_{5.5}$, we find that applying supervision either at the model level or at the selector level is already sufficient to surpass its performance, highlighting the critical role of supervised data in information extraction tasks.

6 Conclusions

We introduce THINKTWICE, a sampling and selection framework for document-level Information Extraction. The proposed method leverages the inherent output variability of decoder-only LLMs to recover higher-quality solutions than those obtained via greedy decoding. We further propose two complementary sample selection strategies: F1 Voting, which is fully unsupervised, and a supervised reward-based selector. Through an extensive experimental evaluation, we show that test-time scaling techniques can be effectively adapted to document-level Information Extraction, surpassing greedy decoding and standard TTS baselines and achieving state-of-the-art results in supervised and cross-lingual settings.

We first demonstrate that reasoning-oriented LLMs consistently outperform non-reasoning models in zero-shot scenario, with particularly strong gains on complex extraction tasks such as those in the BETTER dataset. Motivated by this observation, we conduct a comprehensive analysis across multiple datasets and languages, showing that the sampling and selection technique of THINKTWICE consistently outperforms greedy decoding. Additionally, the F1 Voting selector also surpasses the standard TTS approach, Majority Voting, in terms of performance and stability.

In supervised experiments on MUC-4, we further showed that scaling model size alone is insufficient to produce substantial improvements when relying solely on greedy decoding. In contrast, combining stronger reasoning LLMs with THINKTWICE produces significant gains, highlighting the importance of effective test-time scaling strategies. We additionally demonstrated that the supervised selector can be trained entirely from automatically collected supervision, leading to further improvements beyond the unsupervised setting. Finally, our cross-lingual experiments revealed that both the English fine-tuned reasoning model and the reward-based selector transfer effectively across languages, outperforming systems trained directly on target-language supervision.

Limitations

To apply our approach in a supervised setting, we must train reasoning-oriented LLMs, which requires high-quality reasoning traces. Although the proposed rejection-sampling procedure generated a substantial number of gold-quality trajectories, coverage remained incomplete, leading to noisy training signals. Future work could investigate reinforcement learning methods, such as GRPO, to train reasoning-oriented LLMs that perform more effectively on this task, potentially providing improvements complementary to those achieved through sampling and selection.

THINKTWICE also introduces additional computational overhead at inference time. Generating multiple candidate templates increases both latency and decoding cost relative to standard greedy decoding. Nevertheless, the number of generated samples (e.g., 64) is a tuneable hyperparameter rather than a fixed constant, allowing a flexible trade off between computational cost and performance.

Finally, despite the strong improvements obtained with both F1 Voting and reward-based selection, a large gap remains between current selectors and oracle performance, suggesting that there is a promising future work direction in test-time scaling approaches for IE.

References

1992. *Fourth Message Understanding Conference (MUC-4): Proceedings of a Conference Held in McLean, Virginia, June 16-18, 1992*.

Ralph Allan Bradley and Milton E. Terry. 1952. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345.

Lingjiao Chen, Jared Quincy Davis, Boris Hanin, Peter Bailis, Ion Stoica, Matei Zaharia, and James Zou. 2024. Are more LLM calls all you need? towards the scaling properties of compound AI systems. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Yunmo Chen, William Gantt, Weiwei Gu, Tongfei Chen, Aaron Steven White, and Benjamin Van Durme. 2023. Iterative document-level information extraction via imitation learning. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1858–1874, Dubrovnik, Croatia. Association for Computational Linguistics.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Yuyang Ding, Juntao Li, Pinzheng Wang, Zecheng Tang, Yan Bowen, and Min Zhang. 2024. Rethinking negative instances for generative named entity recognition. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3461–3475, Bangkok, Thailand. Association for Computational Linguistics.

Xinya Du, Alexander Rush, and Claire Cardie. 2021. Template filling with generative transformers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 909–914, Online. Association for Computational Linguistics.

William Gantt, Shabnam Behzad, Hannah An, Yunmo Chen, Aaron Steven White, Benjamin Van Durme, and Mahsa Yarmohammadi. 2024. MultiMUC: Multilingual template filling on MUC-4. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 349–368, St. Julian's, Malta. Association for Computational Linguistics.

Joseph Gatto, Omar Sharif, Parker Seegmiller, and Sarah M. Preum. 2025. Document-level event-argument data augmentation for challenging role types. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 25109–25131, Vienna, Austria. Association for Computational Linguistics.

Aaron Grattafiori, . . . , and Zhiyu Ma. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Ralph Grishman. 2019. Twenty-five years of information extraction. *Natural Language Engineering*, 25(6):677–692.

Ralph Grishman and Beth Sundheim. 1996. Message Understanding Conference- 6: A brief history. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*.

Kung-Hsiang Huang, Sam Tang, and Nanyun Peng. 2021. Document-level entity-based extraction as template generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5257–5269, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

OpenAI: Aaron Jaech, . . . , and Zhuohan Li. 2024. Openai o1 system card. *Preprint*, arXiv:2412.16720.

Sarthak Jain, Madeleine van Zuylen, Hannaneh Hajishirzi, and Iz Beltagy. 2020. SciREX: A challenge dataset for document-level information extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7506–7516, Online. Association for Computational Linguistics.

Heng Ji, Ralph Grishman, Hoa Trang Dang, Kira Griffitt, and Joe Ellis. 2010. Overview of the tac 2010 knowledge base population track. In *Third text analysis conference (TAC 2010)*, volume 3, pages 3–3.

Guozheng Li, Peng Wang, and Wenjun Ke. 2023. Revisiting large language models as zero-shot relation extractors. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 6877–6892, Singapore. Association for Computational Linguistics.

Sha Li, Heng Ji, and Jiawei Han. 2021. Document-level event argument extraction by conditional generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 894–908, Online. Association for Computational Linguistics.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *Preprint*, arXiv:2305.20050.

Timothy Mckinnon and Carl Rubino. 2022. The IARPA BETTER program abstract task four new semantically annotated corpora from IARPA's BETTER program. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 3595–3600, Marseille, France. European Language Resources Association.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: Memory optimizations toward training trillion parameter models. *Preprint*, arXiv:1910.02054.

Oscar Sainz, Iker García-Ferrero, Rodrigo Agerri, Oier Lacalle, German Rigau, and Eneko Agirre. 2024. Gollie: Annotation guidelines improve zero-shot information-extraction. In *International Conference on Representation Learning*, volume 2024, pages 47083–47107.

Marija Sakota and Robert West. 2025. Combining constrained and unconstrained decoding via boosting: BoostCD and its application to information extraction. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 18356–18371, Suzhou, China. Association for Computational Linguistics.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2025. Scaling llm test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*.

Hugo Touvron, . . . , and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Somin Wadhwa, Silvio Amir, and Byron Wallace. 2023. Revisiting relation extraction in the era of large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15566–15589, Toronto, Canada. Association for Computational Linguistics.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. Ace 2005 multilingual training corpus. *Linguistic Data Consortium, Philadelphia*, 57:45.

Zhen Wan, Fei Cheng, Zhuoyuan Mao, Qianying Liu, Haiyue Song, Jiwei Li, and Sadao Kurohashi. 2023. GPT-RE: In-context learning for relation extraction using large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3534–3547, Singapore. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. 2025. Inference scaling laws: An empirical analysis of compute-optimal inference for llm problem-solving. In *The Thirteenth International Conference on Learning Representations*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Meishan Zhang, Bin Wang, Hao Fei, and Min Zhang. 2024. In-context learning for few-shot nested named entity recognition. *Preprint*, arXiv:2402.01182.

Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Wenyue Hua, Haolun Wu, Zhihan Guo, Yufei Wang, Niklas Muennighoff, Irwin King, Xue Liu, and Chen Ma. 2025a. A survey on test-time scaling in large language models: What, how, where, and how well? *Preprint*, arXiv:2503.24235.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025b. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *Preprint*, arXiv:2506.05176.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, pages 35–45.

Zikang Zhang, Wangjie You, Tianci Wu, Xinrui Wang, Juntao Li, and Min Zhang. 2025c. A survey of generative information extraction. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 4840–4870, Abu Dhabi, UAE. Association for Computational Linguistics.

A Prompt

In order to define the Information Extraction task to the LLMs, we have created the prompt detailed below. For **Llama 3.3 70B** and **Qwen 3 32B**, the prompt is structured as follows:

System Prompt: You are an expert in information extraction, you need to extract the information of the document that is provided in *{language}* as a template in JSON format. The guidelines for the dataset you need to extract are the followings: *{guidelines}*

User Prompt: Create the template for the next document: *{document}*

In the case of **Llama R1 70B**, as system prompts are not recommended for it, the instructions are combined into a single user prompt.

B JSON Schemas

The inference process utilizes two distinct JSON schemas: one tailored for MUC-4 (muc, 1992) and MultiMUC (Gantt et al., 2024), and another specifically for the BETTER (Mckinnon and Rubino, 2022) dataset. While this overview outlines their primary structures, the complete schemas are documented in the supplementary materials.

B.1 MUC-4 And MultiMUC

This schema defines a list of templates for each document, where each template corresponds to a single incident. For every incident, the schema requires the explicit specification of an **incident_type**, chosen from a fixed set of categories (kidnapping, attack, bombing, robbery, arson, or forced work stoppage).

In addition to the incident type, each template captures the main participants and elements involved in the event. These include the **PerpInd** (individual perpetrators) and **PerpOrg** (organizational perpetrators), represented as lists of names, as well as **Target**, which refers to inanimate objects or entities attacked. The schema also records the **Victim** slot, defined as people who were targeted or harmed, and the **Weapons** or devices used to carry out the incident. All participant-related fields are modeled as lists of strings to allow for multiple entities per incident.

B.2 BETTER

The BETTER dataset adopts a multi-template schema in which each document is annotated with a set of event templates. Each template corresponds to a specific event type, such as protests, terrorism, epidemics, disasters, displacement, or corruption, and is explicitly identified by a **template_type** field. Templates comprise a structured collection of optional slots designed to capture salient aspects of an event, including participants, locations, temporal information, causes, and outcomes.

To illustrate this structure, consider the **Terrorplate** template, which represents terrorist incidents. Terrorplate captures the basic properties of an attack, including the type of attack, its completion status, and whether it was part of a coordinated or ongoing campaign. It records information about perpetrators, such as named individuals, organizations, or organizers, as well as attribution, blame, and claims of responsibility. The template also includes slots for targets, weapons, and casualties, covering both victims and perpetrators. Outcome fields allow the annotation of effects that occurred, were prevented, or were only potential. Finally, temporal and spatial fields specify when and where the incident took place.

Other templates in the BETTER schema follow the same design principles. Each focuses on a distinct event domain while employing a consistent slot based structure. However, each template has its own slots that are needed in order to extract the concrete information of that event type.

C Guidelines

This section provides a concise overview of the guidelines for the MUC-4 and MultiMUC datasets, as well as those for the BETTER dataset. Full versions of all guidelines are available in Markdown format within the supplementary material.

C.1 MUC-4 and MultiMUC

The MUC-4 and MultiMUC guidelines begin with a concise overview of the MUC dataset. They then establish the criteria for relevant incidents through a set of specific inclusion and exclusion rules. Subsequently, the guidelines define the dataset slots and the required information for each. Finally, they specify the mandatory format for the output templates.

C.2 BETTER

The BETTER dataset guidelines begin by defining the dataset and its six constituent template types: Protesplate, Corruplate, Terrorplate, Epidemiplate,

Hyperparameter	Reasoning Llama	Reasoning Qwen	Non-Reasoning Llama	Non-Reasoning Qwen
Temperature	0,7	0,6	0,6	0,7
Top p	1	0,95	1	0,8
Top k	-1	20	-1	20
Min p	0	0	0	0

Table 4: Inference hyperparameters.

Hyperparameter	Reasoning	Reward
Batch Size	128	512
Learning Rate	$5e^{-5}$	$2e^{-5}$
Epoch	5	4
Weight Decay	$5e^{-5}$	$5e^{-5}$

Table 5: Fine-tuning hyperparameters.

Disasterplate, and Displacemtplate. Subsequently, the documentation provides a detailed description of each specific template type.

D Hyperparameters

The THINKTWICE method requires several hyperparameters, which we categorize into three types: inference hyperparameters, fine-tuning hyperparameters, and rejection sampling hyperparameters.

Inference Due to the different behavior of Reasoning and Non-reasoning models, they usually need to be sampled using different hyperparameters. Table 4 shows the selected inference hyperparameters; the hyperparameters were selected based on the recommended hyperparameters of each model. When sampling is required, **each model is sampled 64 times per prompt**. In those cases, to ensure reproducibility, we fix the random seed to 42.

Fine-tuning The fine-tuning hyperparameters are shown in 5, in which we distinguish two types of fine-tuning: Reasoning models fine-tuning and Reward model fine-tuning. For the two types, the checkpoints are saved on each epoch, and the best performing one on the development split is selected. As in the inference hyperparameters, a random seed of 42 is used in order to ensure reproducibility.

Rejection Sampling We applied iterative rejected-sampling fine-tuning. Llama R1 converged after 8 iterations, and Qwen 3 needed 14 iterations. In both cases, the top 8 templates are used at each iteration for both models. After completing these iterations, we performed a final iteration. In this final step, we used the top 4 templates across the full dataset for Qwen 3 and the top 1 (i.e., the best-performing) template for Llama R1.

E Computational Infrastructure

All experiments were conducted using NVIDIA A100 and H100 GPUs. The specific hardware allocation varied depending on the model scale and the nature of the computational task (fine-tuning and inference).

Fine-tuning Configuration For the **Llama R1 70B model**, we employed DeepSpeed ZeRO-3 (Rajbhandari et al., 2020) to manage memory sharding across 64 A100 GPUs. In contrast, the **Qwen3 32B** model was fine-tuned using 16 H100 GPUs utilizing the same ZeRO-3 configuration.

Inference Configuration Standard model inference was executed using Tensor Parallelism across 4 A100 GPUs to speed up generation. The only exception was the Reward Model, which, due to its smaller size, was deployed on a single A100 GPU.

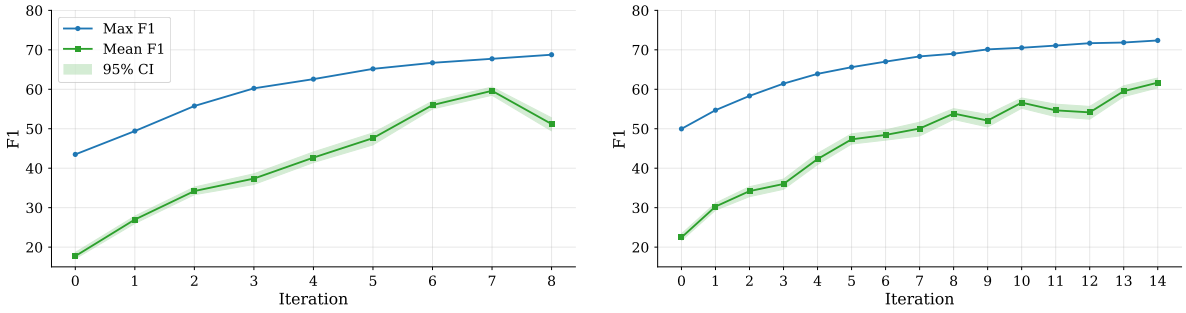
F Output Diversity Analysis

Model	F1-Similarity
GPT 5.5	$71.76_{\pm 12.65}$
Llama R1	$23.20_{\pm 10.03}$
Qwen3	$26.72_{\pm 11.93}$

Table 6: Mean pairwise F1-similarity between N=64 sampled templates on the BETTER dataset for each model. Higher values indicate lower output diversity across samples.

In order to apply THINKTWICE, it is required that the model produces sufficiently diverse predictions. While this is easily obtainable—even for non-reasoning models—with standard inference hyperparameters, a low temperature, top-p or top-k value can make the model yield very similar outcomes.

When attempting to apply THINKTWICE to $GPT_{5.5}$, we found that the API does not provide access to inference hyperparameters, preventing any direct control over sampling behavior. In addition, qualitative inspection of the generated outputs revealed very limited variability, with many predictions being nearly identical. Table 6 quantifies this observation by reporting the mean pairwise F1 similarity between N=64 sampled templates on



(a) Llama R1 70B

(b) Qwen3 32B (Think)

Figure 4: Maximum, mean, and 95% confidence interval results in train split across different numbers of rejection sampling fine-tuning iterations.

the BETTER dataset. $GPT_{5.5}$ shows a substantially higher similarity (71.76) compared to Llama R1 (23.2) and Qwen3 (26.72), indicating a markedly lower level of output diversity.

Given this limited variability and the lack of controllable sampling parameters, we restrict our experiments to open-source models, where inference conditions can be fully controlled and the diversity necessary for THINKTWICE can be reliably ensured.

G Analysis of Rejection Sampling fine-tuning

We analyze rejection sampling fine-tuning by computing the maximum, mean, and 95% confidence interval of the results at each iteration. To estimate the maximum performance, we select, for each document, the best-performing template generated by the model and use these to compute the final score. In contrast, calculating the exact mean and confidence intervals requires evaluating an intractable combinatorial space (64^{1300}). We therefore approximate them using a bootstrapping approach: for each document, we randomly sample one template to compute an overall score across the dataset, repeating this procedure 100 times. The mean and 95% confidence intervals are then derived from these bootstrap samples.

The resulting values are presented in Figure 4, which illustrates the evolution of the rejection-sampling fine-tuned models across iterations. The maximum-performance curves show a clear convergence toward a horizontal asymptote in later iterations, stabilizing around 70 F1 points. In contrast, the mean score exhibits a consistent upward trend across most iterations. The confidence interval also changes over time, becoming wider in later iterations compared to earlier ones.

H Full Results

We present our experimental results below. The unsupervised results are divided into two tables: Table 7 contains the results for the BETTER dataset, while Table 9 details the results for MUC-4 and MultiMUC. Finally, Table 8 summarizes all supervised results.

Method	Selector	BETTER
GPT $_{5.5}$	✗	27.95
Greedy $_{\text{Llama 3.3}}$	✗	3.20
	Majority	$1.72_{\pm 0.42}$
THINKTWICE $_{\text{Llama 3.3}}$	F1	$1.60_{\pm 0.22}$
	(oracle)	5.71
Greedy $_{\text{Llama R1}}$	✗	14.78
	Majority	$10.51_{\pm 2.30}$
THINKTWICE $_{\text{Llama R1}}$	F1	$15.29_{\pm 0.88}$
	(oracle)	34.08
Greedy $_{\text{Qwen no-think}}$	✗	2.46
	Majority	$1.73_{\pm 0.05}$
THINKTWICE $_{\text{Qwen no-think}}$	F1	$1.33_{\pm 0.03}$
	(oracle)	3.27
Greedy $_{\text{Qwen think}}$	✗	16.12
	Majority	$16.26_{\pm 0.82}$
THINKTWICE $_{\text{Qwen think}}$	F1	$19.45_{\pm 0.16}$
	(oracle)	36.74

Table 7: Zero-shot scenario results in BETTER Granular dataset. We compare our approach using unsupervised selectors.

Method	Selector	English	Arabic	Farsi	Korean	Russian	Chinese	Average
Reference	✗	35.2	21.46	20.66	**23.91**	23.77	21.93	24.49
Greedy $_{\text{Llama R1}}$	✗	28.52	3.57	1.30	3.07	0.51	4.21	6.86
THINKTWICE $_{\text{Llama R1}}$	Majority	28.41$_{\pm 1.49}$	2.69$_{\pm 0.50}$	0.53$_{\pm 0.57}$	2.29$_{\pm 0.15}$	0.06$_{\pm 0.11}$	4.72$_{\pm 0.84}$	6.45$_{\pm 0.77}$
	F1 Voting	36.56$_{\pm 0.22}$	3.79$_{\pm 0.30}$	1.43$_{\pm 0.39}$	3.88$_{\pm 0.31}$	0.24$_{\pm 0.10}$	5.34$_{\pm 0.46}$	8.54$_{\pm 0.32}$
	Reward	41.11$_{\pm 0.48}$	5.65$_{\pm 0.15}$	2.43$_{\pm 0.34}$	5.08$_{\pm 0.21}$	1.36$_{\pm 0.21}$	10.98$_{\pm 0.27}$	11.10$_{\pm 0.30}$
	(oracle)	56.60	12.79	4.66	11.82	2.96	19.44	18.04
Greedy $_{\text{Qwen think}}$	✗	35.97	17.60	16.17	7.79	15.33	16.14	18.17
THINKTWICE $_{\text{Qwen think}}$	Majority	35.67$_{\pm 0.59}$	18.20$_{\pm 0.25}$	17.13$_{\pm 0.87}$	5.78$_{\pm 0.70}$	16.11$_{\pm 1.60}$	15.26$_{\pm 0.28}$	18.02$_{\pm 0.85}$
	F1 Voting	40.77$_{\pm 0.53}$	22.66$_{\pm 0.44}$	23.61$_{\pm 0.77}$	8.73$_{\pm 0.75}$	20.53$_{\pm 1.18}$	22.15$_{\pm 0.92}$	23.07$_{\pm 0.80}$
	Reward	**42.51**$_{\pm 0.72}$	**30.27**$_{\pm 0.49}$	**30.16**$_{\pm 0.66}$	9.62$_{\pm 0.14}$	**29.95**$_{\pm 1.23}$	**29.75**$_{\pm 0.75}$	**28.71**$_{\pm 0.74}$
	(oracle)	58.32	45.52	46.74	22.63	45.14	46.73	44.18

Table 8: Supervised and cross-lingual results on MUC-4 (English) and MultiMUC comparing greedy decoding against different selection strategies in reasoning fine-tuned models on English data. We also report the results of the state-of-the-art method, IterX $_{\text{T5-enc large}}$ (Chen et al., 2023) fine-tuned using both English and target-language data for training (Reference) reported in the work of (Gantt et al., 2024).

Method	Selector	English	Arabic	Farsi	Korean	Russian	Chinese	Average
GPT $_{5.5}$	✗	29.50	24.60	22.03	7.59	17.70	17.59	19.84
Greedy $_{\text{Llama 3.3}}$	✗	18.27	14.52	14.57	5.55	10.84	11.02	12.46
THINKTWICE $_{\text{Llama 3.3}}$	Majority	18.69$_{\pm 0.19}$	14.62$_{\pm 0.63}$	15.09$_{\pm 0.18}$	5.69$_{\pm 0.12}$	11.27$_{\pm 0.43}$	11.68$_{\pm 0.27}$	12.84$_{\pm 0.35}$
	F1 Voting	18.60$_{\pm 0.14}$	14.80$_{\pm 0.11}$	15.02$_{\pm 0.06}$	5.79$_{\pm 0.17}$	10.99$_{\pm 0.18}$	11.62$_{\pm 0.18}$	12.80$_{\pm 0.15}$
	Reward	22.84$_{\pm 0.36}$	18.83$_{\pm 0.13}$	18.60$_{\pm 0.62}$	7.26$_{\pm 0.20}$	15.04$_{\pm 0.28}$	17.01$_{\pm 0.03}$	16.60$_{\pm 0.33}$
	(oracle)	26.76	25.71	26.07	10.87	19.37	20.51	21.55
Greedy $_{\text{Llama R1}}$	✗	18.68	15.11	12.55	5.53	11.85	12.27	12.67
THINKTWICE $_{\text{Llama R1}}$	Majority	21.52$_{\pm 0.76}$	16.23$_{\pm 0.96}$	14.42$_{\pm 0.62}$	5.74$_{\pm 0.28}$	12.31$_{\pm 0.75}$	12.78$_{\pm 0.29}$	13.83$_{\pm 0.66}$
	F1 Voting	21.69$_{\pm 0.15}$	16.81$_{\pm 0.50}$	16.26$_{\pm 0.38}$	6.18$_{\pm 0.26}$	12.63$_{\pm 0.25}$	12.87$_{\pm 0.29}$	14.41$_{\pm 0.33}$
	Reward	31.43$_{\pm 0.80}$	24.58$_{\pm 0.41}$	21.15$_{\pm 0.38}$	8.02$_{\pm 0.60}$	18.32$_{\pm 0.25}$	19.68$_{\pm 0.77}$	20.53$_{\pm 0.57}$
	(oracle)	42.32	36.06	36.78	16.87	29.89	28.72	31.77
Greedy $_{\text{Qwen no-think}}$	✗	17.71	14.30	13.74	5.26	10.54	12.43	12.33
THINKTWICE $_{\text{Qwen no-think}}$	Majority	17.99$_{\pm 0.17}$	14.83$_{\pm 0.26}$	14.22$_{\pm 0.28}$	5.78$_{\pm 0.43}$	10.85$_{\pm 0.09}$	12.99$_{\pm 0.46}$	12.78$_{\pm 0.31}$
	F1 Voting	18.22$_{\pm 0.10}$	15.30$_{\pm 0.05}$	14.49$_{\pm 0.08}$	5.71$_{\pm 0.26}$	11.01$_{\pm 0.08}$	13.79$_{\pm 0.46}$	13.09$_{\pm 0.22}$
	Reward	24.06$_{\pm 0.25}$	17.27$_{\pm 0.07}$	16.61$_{\pm 0.31}$	7.14$_{\pm 0.27}$	13.94$_{\pm 0.15}$	17.74$_{\pm 0.26}$	16.13$_{\pm 0.23}$
	(oracle)	26.92	22.05	21.79	9.62	18.11	21.46	19.99
Greedy $_{\text{Qwen think}}$	✗	22.99	14.28	16.13	6.88	11.68	15.92	14.65
THINKTWICE $_{\text{Qwen think}}$	Majority	25.80$_{\pm 0.87}$	15.81$_{\pm 0.38}$	17.47$_{\pm 1.24}$	6.93$_{\pm 0.88}$	14.66$_{\pm 0.75}$	16.43$_{\pm 0.98}$	16.18$_{\pm 0.89}$
	F1 Voting	24.30$_{\pm 0.57}$	17.66$_{\pm 0.32}$	19.62$_{\pm 0.44}$	7.22$_{\pm 0.37}$	14.34$_{\pm 0.16}$	16.20$_{\pm 0.16}$	16.56$_{\pm 0.37}$
	Reward	**33.46**$_{\pm 0.70}$	**26.05**$_{\pm 0.24}$	**26.80**$_{\pm 0.21}$	**8.71**$_{\pm 0.53}$	**20.66**$_{\pm 0.25}$	**25.47**$_{\pm 0.74}$	**23.53**$_{\pm 0.50}$
	(oracle)	46.48	39.15	39.00	17.94	31.21	38.13	35.32

Table 9: Zero-shot scenario results. We compare our approach using unsupervised selectors (F1 and Majority Voting) and supervised selectors (Reward trained in English data) in MUC-4 (English) and MultiMUC.