Goldsmith: Gold-Loss-Guided Prompt Training for Agentic Annotation

Anonymous ACL submission

Abstract

Many annotation projects begin before experts have a stable manual or enough labels for a task-specific model. We present Goldsmith, an agentic annotation pipeline for this definition-building stage. Goldsmith treats the operational definition as a trainable textual object: a small gold set is used to test candidate definitions under structured loss, while the schema, format contract, retrieval, repair, judge routing, and human review remain in a model-external harness. The optimizer converts high-loss gold failures into textual-gradient edit directions, asks an LLM editor for abstract candidate definitions, and accepts a revision only when the same gold loss improves. We evaluate Goldsmith on typed multilingual NER and pair-level relation classification. Across these settings, gold-loss-guided prompt training optimizes prompts more efficiently than direct rewriting or history-only prompt search, yielding operational definitions that also produce more accurate downstream annotations. The downstream harness further improves annotation quality by combining optimized definitions with retrieval, judge-based routing, and human review. The results suggest that, before large stable datasets exist, carefully tested gold examples can serve as executable supervision for building reusable annotation definitions.

1 Introduction

Many annotation projects begin before the task is fully defined: experts may recognize the phenomenon to annotate, yet lack a stable boundary policy, a complete manual, or enough labels for a task-specific model (Settles, 2009; Tomanek and Olsson, 2009). PLM-first systems remain strong once this target is stable. BERT-style and XLM-R token classifiers are natural span-labeling baselines (Devlin et al., 2019; Conneau et al., 2020), while typed-marker relation classifiers can use stronger encoders such as RoBERTa or DeBERTa

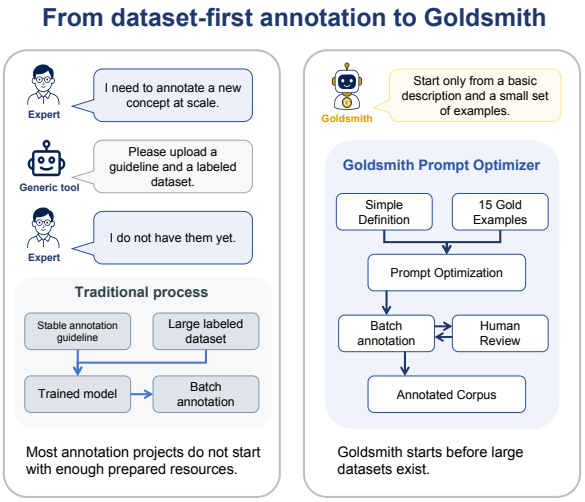

Figure 1: Goldsmith teaser for gold-loss-guided annotation.

(Liu et al., 2019; He et al., 2023). Even parameter-efficient updates such as LoRA reduce training cost, not the cost of deciding what the task means (Hu et al., 2022). The bottleneck is therefore earlier: converting a partially articulated concept into an operational annotation procedure, a problem also discussed in LLM annotation and guideline-following IE (Tan et al., 2024; Sainz et al., 2024).

Long-context in-context learning changes this early-stage design space. Few-shot prompting showed that examples can condition a model without weight updates (Brown et al., 2020); many-shot and long-context studies show that expert examples, failures, and revision history can now fit inside the working context (Abbas et al., 2024; Bertsch et al., 2025). However, unselected or unvalidated examples can dilute the expert signal (Zou et al., 2025; Zhang et al., 2025b). In this setting, carefully chosen gold examples can define the task boundary more concretely than an initial prose guideline.

We introduce Goldsmith, an annotation harness built around this premise. Gold examples

are the scarce material experts can provide early; the system repeatedly tests, reshapes, and polishes that material into a durable operational definition. Given a short concept description and a 15-example gold calibration set, Goldsmith treats each gold example as a test case for the current definition, using prediction errors to decide whether a revised definition has captured the expert's intended boundary. The definition then runs inside a model-external harness that connects model calls to retrieval tools, context construction, format repair, judges, review queues, and feedback loops; these tool-calling capabilities turn a revised definition into a stable workflow rather than a one-off LLM response (Liu et al., 2025; Qin et al., 2025).

The methodological core is prompt-as-parameter training: Goldsmith optimizes only the operational definition, while the schema, format reference, retrieved references, and harness controls remain outside the trainable definition. Candidate definitions come from history-only search, LLM-derived textual gradients, or Goldsmith's own gold-loss-guided editor (Yang et al., 2024; Pryzant et al., 2023), but acceptance is decided by the same gold loss rather than by the LLM's self-assessment (Yuksekgonul et al., 2025). The claim is bounded: Goldsmith targets the early, evolving setting before PLM-first pipelines have enough stable labels, and asks whether the same 15 gold examples can outperform zero-shot prompting, static ICL, retrieval-only prompting, history-only prompt search, and low-budget fine-tuning.

This paper makes three contributions. First, we introduce a gold-loss-guided prompt optimization method that treats the operational definition as the trainable textual parameter and accepts LLM-proposed revisions only when they improve structured loss on expert gold examples. Second, we build this optimizer into an annotation pipeline whose harness keeps schema, formatting, retrieval, repair, judge routing, and human review outside the trainable prompt while making the optimized definition usable for batch annotation. Third, we empirically validate the approach on two standard tasks—OpenNER for typed multilingual NER and CrossRE for pair-level relation classification—against LLM-only, prompt-optimization, and PLM baselines (Palen-Michel et al., 2025; Bassignana and Plank, 2022).

2 Related Work

Before LLM annotation became practical, span labeling and relation classification were usually supervised adaptation problems. BERT, XLM-R, RoBERTa, and DeBERTa remain strong encoder baselines once labels and boundaries are fixed (Devlin et al., 2019; Conneau et al., 2020; Liu et al., 2019; He et al., 2023). Parameter-efficient adaptation lowers update cost (Houlsby et al., 2019; Li and Liang, 2021; Lester et al., 2021; Hu et al., 2022), but still assumes a stable objective and reliable labels; this leaves an earlier definition-building phase in which experts may have only a partial concept description and a few authoritative cases.

Reliable LLM annotation also depends on retrieval, parser checks, repair calls, judge routing, and human review around the model call (Lewis et al., 2020; Es et al., 2024; Chen et al., 2024; Gu et al., 2026; Schroeder and Wood-Doughty, 2024; Shi et al., 2025; Raina et al., 2024; Liu et al., 2025; Qin et al., 2025). These controls turn a single response into a checked, recoverable workflow.

2.1 LLM Annotation and In-Context Task Definitions

Early work treated LLMs as possible substitutes or assistants for human labelers, asking whether GPT-style models could reduce labeling cost or approximate crowd annotations (Wang et al., 2021; Gilardi et al., 2023). Follow-up systems placed LLMs inside active or analyst-in-the-loop workflows, where uncertainty and human checks decide where expert effort should be spent (Zhang et al., 2023; Dai et al., 2023). Recent surveys broaden this view from one-off labeling to annotation and synthesis pipelines, while warning that generated labels require validation, calibration, and task-specific controls (Tan et al., 2024; Xu et al., 2024). In information extraction, guideline-following studies show that natural-language instructions can carry annotation policy, but structured outputs remain sensitive to parsing, schema constraints, and repair (Sainz et al., 2024; Huang et al., 2024).

In-context learning provides the mechanism that makes such task definitions practical at inference time. Few-shot prompting established that examples can condition behavior without parameter updates (Brown et al., 2020); long-context and many-shot work showed that modern models

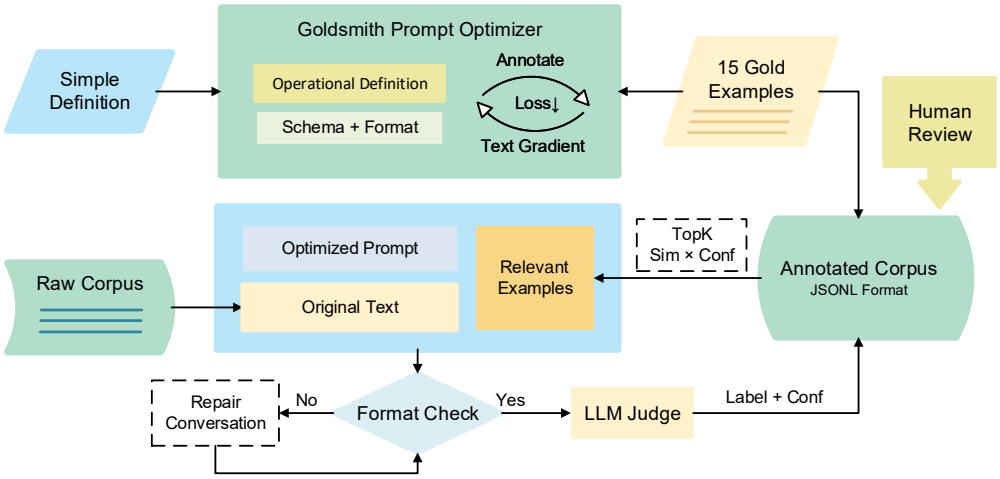

Figure 2: Goldsmith pipeline for training an operational definition and applying it through a model-external annotation harness.

can consume larger sets of demonstrations, failures, and revision context (Abbas et al., 2024; Bertsch et al., 2025; Baek et al., 2025). However, context length is not the same as useful supervision: many-shot evaluation, differentiated reweighting, and gradient-matching selection all suggest that demonstrations must be chosen and validated rather than appended indiscriminately (Zou et al., 2025; Zhang et al., 2025b,a). Thus, LLM annotation increasingly depends on treating examples as operational task-definition evidence, not merely as extra prompt tokens.

2.2 Prompt Optimization

Prompt optimization work moves beyond hand-written prompting; recent surveys describe a spectrum from discrete trigger search to LLM-driven instruction revision (Ramnath et al., 2025). Shin et al. (2020) showed that prompts can be searched as discrete triggers rather than only written by humans. APE and PromptBreeder extend this idea to natural-language instructions, using LLMs to propose, mutate, and score candidate prompts (Zhou et al., 2023; Fernando et al., 2024). Soft-prompt and prefix-tuning methods optimize continuous prompt-like parameters (Li and Liang, 2021; Lester et al., 2021), whereas instruction-level optimization keeps the artifact readable and directly deployable. OPRO frames an LLM as a black-box optimizer over prompt-score histories (Yang et al., 2024). ProTeGi uses a critic to convert failures into natural-language textual gradients and an editor to produce candidate prompts (Pryzant et al., 2023); TextGrad generalizes textual feedback as a differentiable-style signal for compound LLM systems (Yuksekgonul et al., 2025).

Recent systems make this loop more automatic. Ye et al. (2024), Zhang et al. (2024), Jin et al. (2025), and Kepel and Valogianni (2024) automate prompt engineering, feedback-reflect-refine, reranking, and autonomous search. The common pattern is that the LLM proposes textual updates, while an external objective, evaluator, or feedback loop selects among them; for annotation, a fluent rewritten guideline is not necessarily better unless it improves labeled-example behavior under the same evaluation contract.

3 Method

Goldsmith is a gold-loss-guided annotation pipeline for turning a small expert calibration set into a reusable annotation definition. The input is an initial task description c_0, a small gold set G, and an unlabeled corpus U. The output is an optimized operational definition $c^\star$, together with predictions and confidence scores on U. Here, *gold loss* is the structured error obtained by running a candidate definition on G, parsing the model outputs, and comparing them with the expert gold annotations. Figure 2 gives the full pipeline.

The key design is to separate the trainable task meaning from the model-external harness. Goldsmith optimizes only the operational definition: the textual description of what should be annotated and where its boundaries lie. The harness is the

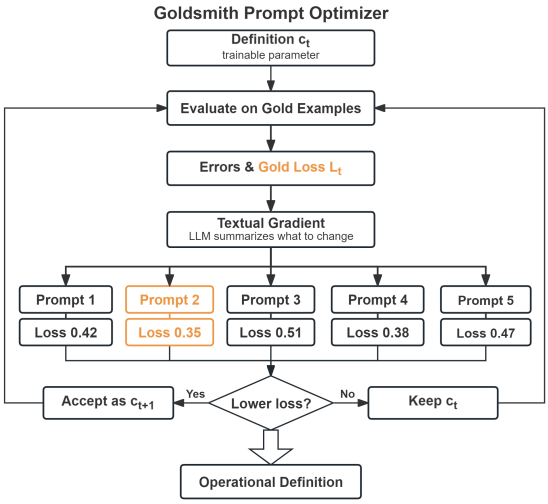

Figure 3: Goldsmith textual-gradient optimizer with gold-loss candidate selection.

runtime and control layer around model calls; it supplies the schema, format reference, retrieved examples, parser checks, repair calls, judge routing, review feedback, and export format. A concrete annotation prompt is assembled from the operational definition, schema, format reference, and optional retrieved references. This separation prevents the optimizer from improving loss by changing the output protocol rather than the task definition.

3.1 Gold-Loss-Guided Definition Training

Let c_t be the operational definition at round t. For an input x, retrieved references r, schema q, and format reference f, the full prompt is $p_t = h(c_t, x, r, q, f)$. Candidate definitions can modify c_t only; q, f, parsing, repair, retrieval, and review policies are fixed. Every candidate is therefore evaluated through the same annotation interface and the same gold loss.

Goldsmith is inspired by ProTeGi's textual gradients and TextGrad's view of textual feedback as an optimization signal (Pryzant et al., 2023; Yuksekgonul et al., 2025), but specializes this idea for structured annotation. At each round, the system evaluates c_t on all gold items, selects the k highest-loss failures, and summarizes them into textual-gradient edit directions. An LLM editor then proposes abstract candidate definitions. The editor does not decide whether a revision is better: candidates are accepted only after they are rerun on G and their measured gold loss decreases. Thus the LLM proposes definition edits, while gold examples supply the objective and acceptance rule.

The optimization is also stateful. Goldsmith keeps accepted and rejected candidates, their losses, the highest-loss failures before each edit, and the residual failures after accepted edits. This memory helps the editor avoid repeatedly producing plausible but ineffective rewrites: it receives evidence about which semantic changes reduced loss, which failed, and which hard cases remain. Training stops when all gold items pass or when the patience budget is exhausted.

3.2 Gold Loss for Annotation Interfaces

Goldsmith targets two common annotation interfaces: typed span annotation and given-pair relation annotation. For both, outputs are first parsed into the required schema; unparsable outputs receive a fixed parse-failure penalty.

For typed span annotation, each item contains a set of labeled spans. Let G_i^T and P_i^T be the gold and predicted typed spans (s, e, y), and let G_i^B, P_i^B be their boundary-only projections. Boundary loss combines missed and extra boundaries:

$$L_B(i) = \frac{1}{2.2}\left(1.2\,\frac{|G_i^B \setminus P_i^B|}{\max(1, |G_i^B|)} + \frac{|P_i^B \setminus G_i^B|}{\max(1, |P_i^B|)}\right).$$

Type loss is computed only on matched boundaries, $M_i = G_i^B \cap P_i^B$:

$$L_{\text{type}}(i) = \begin{cases} E_{\text{type}}(i)/|M_i|, & |M_i| > 0, \\ 0, & |M_i| = 0, \end{cases}$$

The span loss combines these terms, weighting boundary recovery more heavily while still penalizing label errors; small penalties are added for typed exact mismatch and unstable outputs.

For given-pair relation annotation, the input supplies a sentence and a marked head–tail entity pair. The output is one relation label, including `no_relation`; the per-item relation loss is label disagreement:

$$\tau_{\text{rel}}(i) = \mathbb{1}[\hat{y}_i \neq y_i],$$

with parse-failure and instability penalties added at the prompt level. No separate exact-match term is needed because the relation label is the complete output. For both interfaces, prompt-level loss is 100 times the mean per-example loss.

3.3 Batch Annotation and Review Harness

After training, Goldsmith applies $c^\star$ to U through the fixed harness. For each item, the context builder retrieves references from the gold set and from previously reviewed high-confidence annotations. In score-review runs, low judge-score items are corrected by human review, promoted to high-confidence references, and used when the system reruns annotation over the target corpus.

The harness also makes LLM calls recoverable. The parser checks schema validity and text preservation. If a model response violates the format contract, Goldsmith first performs conversational repair by asking the model to fix only the format while preserving the annotation content. Only unrepaired failures are retried as full annotation attempts.

Finally, an LLM judge can score parseable predictions for format validity, concept fit, boundary quality, relation quality, missed-span risk, extra-span risk, and overall quality. Judge scores are used only as routing signals. Before relying on them for review routing, Goldsmith checks that low-score bins concentrate true errors (Gu et al., 2026; Schroeder and Wood-Doughty, 2024; Doostmohammadi et al., 2024). Figure 4 shows this calibration check for the OpenNER dataset slice. Human review then prioritizes low-score items, so limited review effort is spent where errors are most likely.

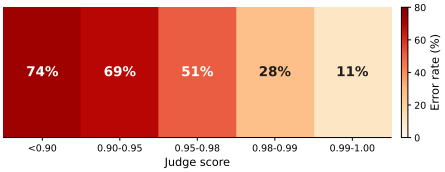

Figure 4: OpenNER dataset-slice error rate by LLM-judge score bin.

4 Experiments

4.1 Tasks and Datasets

We evaluate Goldsmith on two annotation tasks. OpenNER is a multilingual typed span-labeling task: the input is a text and the output is a set of entity spans labeled as PER, ORG, or LOC (Palen-Michel et al., 2025). It represents span-labeling tasks because the system must recover both entity boundaries and type labels. CrossRE is a given-pair relation-classification task: the input is a sentence with a marked head entity, tail entity, and entity types, and the output is one of 17 positive relation labels or `no_relation` (Bassignana and Plank, 2022). It represents pair-level relation classification.

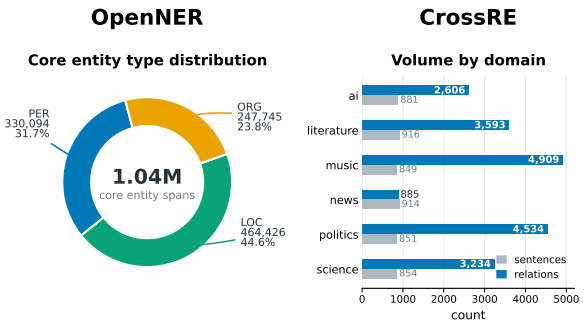

Figure 5: Dataset overview for the OpenNER span-labeling and CrossRE relation-classification slices.

For both tasks, we run experiments on fixed dataset slices and reserve a 15-example calibration set for prompt optimization. The OpenNER multilingual slice and gold selection are described in Appendix A.1 and Appendix A.2; the CrossRE six-domain slice and pair-level gold selection are described in Appendix A.3 and Appendix A.4. Unless otherwise specified, all LLM-based prompt optimization, annotation, judging, and review-routing experiments use GPT-5.5; model identifiers, access month, decoding parameters, retry bounds, and review settings are listed in Appendix B.1. For OpenNER task-stage evaluation, all rows use the same held-out evaluation subset. The 15 calibration examples and the items selected for human review are excluded from this subset for every setting. In review-based settings, selected low-score items are corrected as reviewed feedback, promoted to high-confidence references, and then used when the system reruns annotation over the target corpus.

4.2 Prompt-Optimization Baselines

The prompt-optimization experiment tests whether textual-gradient feedback improves the operational definition from 15 gold examples. All optimizers share the same initial description, gold set, candidate budget, gold loss, acceptance rule, and patience rule.

LLM rewrite only. This baseline receives only the current operational definition and a generic rewrite instruction, without gold examples, errors, history, or textual-gradient feedback.

OPRO history search. OPRO is implemented as a history-only black-box optimizer (Yang et al., 2024). It sees previous candidate definitions and loss-derived scores, but not source texts, gold answers, model outputs, or error types. Candidates are accepted only if measured gold loss improves.

Goldsmith. Goldsmith is the method described in Section 3. It summarizes the highest-loss gold errors into textual-gradient edit directions, maintains optimization memory across rounds, asks an LLM editor for abstract candidate definitions, and accepts a candidate only when the same gold loss decreases. We report loss, 15-gold pass count, best round, evaluated candidates, elapsed time, and token usage.

4.3 Task-Stage Baselines

Task-stage baselines compare the resulting annotation workflow with low-budget supervised PLM baselines under the same task interface where possible. For OpenNER, **Goldsmith score-review** uses the optimized definition followed by score-routed review, while **Goldsmith no-review** reports the first annotation pass before human correction. **Goldsmith zero-shot** uses the optimized definition but disables retrieval, judge scoring, and review, isolating the value of the learned definition alone. **Initial-prompt zero-shot** uses 0 gold examples and the initial task description before prompt optimization, also without retrieval, judge scoring, or review. The PLM reference condition follows the local gold4-per-language setup: **mBERT** and **XLM-R** token classifiers are fine-tuned on 208 deterministic multilingual training examples. For CrossRE, **Zero-shot LLM** receives the sentence, marked head–tail pair, entity types, and relation-label set, but no gold examples or optimized definition. The PLM comparison uses typed-marker relation classifiers: the official **BERT** CrossRE anchor and stronger **RoBERTa-large** or **DeBERTa-v3** encoder variants, all receiving the same sentence, gold entity pair, entity types, and relation-label set (Bassignana and Plank, 2022; Devlin et al., 2019; Liu et al., 2019; He et al., 2023). For CrossRE, all rows are evaluated on the same 29,171-pair evaluation slice. Goldsmith uses the optimized relation definition with reviewed feedback and retrieved references, while the PLM rows are low-budget supervised references trained from 200 sentences expanded into 10,593 training pairs.

4.4 Metrics

Prompt-optimization results report gold-set pass count, gold loss, best round, and evaluated candidate count. OpenNER task-stage results report supervision or feedback used by the system, typed precision, typed recall, micro-F1, macro-F1, and typed exact match on the shared held-out evaluation subset. CrossRE reports accuracy, positive-class macro-F1, positive-relation recall, and positive-relation F1; the positive-class macro-F1 excludes `no_relation`, since negative pairs dominate. OpenNER component ablations report precision, recall, F1, and F1 change relative to the main setting, while model ablations report F1 without review, F1 with review, and exact match.

5 Results and Analysis

5.1 Prompt Optimization Results

Table 1 isolates the definition-training problem before any held-out task evaluation. All optimizers start from the same short description, the same 15 gold examples, and the same candidate budget. Direct LLM rewriting improves the prompt but stalls at $7/15$ passing gold items; OPRO history search does better at $9/15$, showing that prompt–score history is a useful black-box signal. Goldsmith reaches $14/15$ with loss 1.8030, a much larger improvement under the same budget. The main difference is not that Goldsmith uses a stronger editor, but that it converts the highest-loss gold failures into textual-gradient edit directions and still accepts edits only after measured gold-loss reduction. This result establishes optimization efficiency on the shared gold-loss objective; the downstream ablation in Table 4 then checks whether the optimized definition remains useful inside the annotation workflow.

Optimizer	Pass	Loss ↓	Best round
LLM rewrite only	7/15	22.0303	7
OPRO history-only	9/15	14.5909	16
Goldsmith gold-loss	**14/15**	**1.8030**	18

Table 1: OpenNER prompt-optimizer comparison from the same initial task description and 15-example gold calibration set.

Figure 6 shows the same pattern over time. The baselines occasionally find better candidates, but their trajectories remain noisy because they receive either no diagnostic signal or only aggregate history. Goldsmith produces a steadier loss drop,

Setting	Supervision / feedback	P	R	F1	Macro	Exact
Goldsmith score-review	15 gold + 193 review	**0.8092**	**0.8257**	**0.8174**	**0.8919**	**0.8140**
Goldsmith no-review	15 gold	0.7293	0.7505	0.7398	0.8669	0.7749
Goldsmith zero-shot	15 gold	0.7133	0.7440	0.7283	0.8556	0.7613
Initial-prompt zero-shot	0 gold	0.7040	0.7441	0.7235	0.8546	0.7565
mBERT	208 ex.	0.4867	0.5595	0.5206	0.5413	0.5630
XLM-R	208 ex.	0.3281	0.3799	0.3521	0.3735	0.4210

Table 2: OpenNER task-stage results on the same held-out evaluation subset.

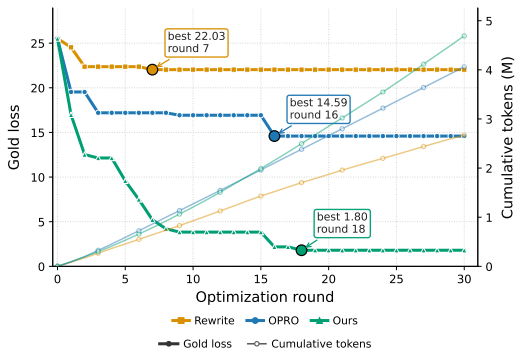

Figure 6: OpenNER prompt-optimizer efficiency from the same initial task description and gold calibration set.

Setting	Acc.	Pos. Macro	Pos. R	Pos. F1
Goldsmith	0.8650	**0.4273**	**0.7273**	**0.5439**
Zero-shot LLM	0.8720	0.1640	0.1980	0.2470
BERT	0.3402	0.0883	0.4845	0.1307
RoBERTa	0.3998	0.1108	0.5231	0.1511
DeBERTa	0.4174	0.1088	0.4834	0.1445

Table 3: CrossRE pair-level results on the same 29,171-pair evaluation slice.

which is the behavior needed in the definition-building stage: the optimizer must turn a few expert examples into a more precise operational definition rather than merely generate fluent prompt variants.

5.2 Task Results

OpenNER. Table 2 reports the multilingual typed span-labeling result. All rows use the same held-out evaluation subset, after excluding the 15 calibration examples and the reviewed feedback items. The no-review row evaluates the optimized definition after the first annotation pass. The score-review row first corrects low-judge-score items as reviewed feedback, promotes them to high-confidence references, and reruns annotation over the target corpus before evaluating on the same subset. Under this second-pass protocol, Goldsmith reaches 0.8174 micro-F1, 0.8919 macro-F1, and 0.8140 typed exact match. This supports the intended division of labor. Prompt training makes the definition usable, while judge routing and review concentrate expert effort on likely errors instead of treating all items equally.

The same table also shows why the comparison is not simply "LLM versus PLM." The optimized-definition zero-shot row improves over the initial-prompt zero-shot row, but both trail the normal no-review run with retrieved references. This separates two effects: prompt optimization improves the operational definition, while retrieval still provides useful in-context task grounding. The low-budget mBERT and XLM-R rows are weaker under this data budget, which is consistent with the paper's target setting: supervised encoders become attractive after the task is stable and more labels exist, but they are a poor fit for the earlier definition-building phase. These PLM numbers should therefore be read as budget-matched baselines, not as reproductions of full-data results reported in prior supervised NER work; their training data are deliberately limited to the same order of magnitude as the gold and review budget used by the annotation workflow.

CrossRE. Table 3 reports pair-level relation classification on the same 29,171-pair CrossRE evaluation slice. This removes the denominator ambiguity: Goldsmith and the PLM anchors are compared on the same final evaluation scope, although they use different learning mechanisms. Accuracy is not the decisive metric here because `no_relation` dominates the test pairs. Goldsmith reaches accuracy 0.8650, positive macro-F1 0.4273, positive recall 0.7273, and positive F1 0.5439. The result suggests that the pipeline is not limited to span boundaries. When the interface changes from span recovery to relation classification, the same principle still matters: a small set of reviewed examples and prompt-training feedback can steer the model toward rare positive labels that

accuracy-oriented baselines tend to miss.

5.3 Ablation Experiments

Setting	P	R	F1	ΔF1
Ours	0.8092	0.8257	0.8174	–
Initial def.	0.6510	0.6820	0.6661	-0.1513
Random-10 ref.	0.7832	0.8001	0.7915	-0.0259

Table 4: OpenNER component ablation over prompt optimization and reference selection.

Table 4 separates definition quality from reference selection. All ablation rows use the same OpenNER evaluation subset as Table 2. Removing prompt optimization drops F1 from 0.8174 to 0.6661, indicating that the learned operational definition is not replaceable by the initial prose description. Together with the calibration loss result in Table 1, this shows both that Goldsmith finds lower-loss definitions more efficiently and that using the optimized definition matters for the downstream task-stage run. Replacing structured retrieval with random references is less damaging but still reduces F1 to 0.7915. The two ablations therefore support different parts of the method: gold-loss prompt training supplies the main task-boundary improvement, while retrieval quality supplies additional stability during batch annotation.

Backend	Provider	w/o review	w/ review	Exact
GPT-5.5	OpenAI	0.7398	**0.8174**	**0.8140**
GPT-5.4	OpenAI	0.7222	0.8012	0.7989
Kimi K2.5	Moonshot	0.7160	0.7890	0.7820
DSv4p	DeepSeek	**0.7660**	0.7947	0.7863

Table 5: OpenNER model ablation with the same optimized operational definition where available.

Table 5 asks whether the result is tied to one backend. GPT-5.5 and GPT-5.4 are OpenAI models, Kimi K2.5 is from Moonshot AI, and DSv4p is DeepSeek V4 Pro from DeepSeek (OpenAI, 2026b,a; Moonshot AI, 2026; DeepSeek, 2026); the exact run-facing identifiers and access month are listed in Appendix B.1. The comparison keeps the optimized definition, review protocol, and held-out evaluation setting fixed across backends. The reviewed ranking broadly follows external model-capability leaderboards such as the Artificial Analysis Intelligence Index, which aggregates reasoning, coding, mathematics, and knowledge benchmarks (Artificial Analysis, 2026). Model quality still matters: GPT-5.5 gives the best reviewed F1 and exact match, while DSv4p has the strongest without-review F1 but loses that lead after review. At the same time, the reviewed scores are relatively close across capable backends. This is the desired behavior of the harness: optimized definitions, retrieval, repair, judge routing, and human review do not eliminate model differences, but they reduce the extent to which the final annotation quality is determined by raw backend strength alone.

6 Conclusion

Goldsmith addresses a narrow but common stage of annotation: experts know the phenomenon they want to annotate, but the operational definition is still being formed and only a few authoritative examples are available. The central result is that those examples can be used as executable supervision rather than only as static demonstrations. In prompt optimization, gold-loss-guided textual-gradient editing produces a substantially stronger operational definition than direct LLM rewriting or history-only search. In task-stage evaluation, the resulting definition, combined with retrieval, judge routing, and human review, improves OpenNER typed span labeling and CrossRE positive-relation classification under low-label budgets.

The broader implication is that long-context LLM annotation should not be reduced to adding more examples to a prompt. Its practical value depends on a harness that keeps the schema fixed, tests definition edits against gold loss, repairs and audits model outputs, and sends uncertain cases back to human review. This does not replace supervised PLM pipelines once large stable datasets exist. Instead, it provides a way to reach that stable task definition more efficiently, with explicit traces of which gold examples shaped the definition and where human review changed the corpus.

Limitations

Goldsmith is designed for tasks whose boundaries can be expressed through natural-language definitions and examples. It may be less suitable for tasks that require private expert knowledge not present in the gold examples, tasks with highly ambiguous labels, or settings where LLM API calls are prohibited.

Goldsmith is also not always the best execution strategy at very large scale. When the target corpus contains tens of thousands of items or more, direct LLM annotation may be constrained by token cost, provider concurrency limits, rate limits, and retry overhead. In such settings, a more practical workflow may be to use Goldsmith to build a smaller, high-quality labeled set and a stable operational definition, then train a supervised PLM or smaller task-specific model on that data for large-scale labeling. Goldsmith is therefore best viewed as a definition-building and low-budget annotation method, not as a replacement for efficient supervised models once enough reliable labels exist.

Ethical Considerations

Goldsmith is intended to reduce the cost of building annotation workflows, not to replace expert judgment in high-stakes domains. The system records model calls, prompt revisions, reviewer decisions, and export artifacts so that users can inspect and correct model behavior.

Our experiments use fixed research slices derived from existing OpenNER and CrossRE artifacts, and Appendix A documents how those slices and calibration sets are constructed. These derived evaluation slices are used for research comparison only; users should follow the licenses, terms, and redistribution conditions of the source datasets and model providers.

The workflow sends task text, prompts, retrieved references, model outputs, and judge inputs to hosted LLM APIs during annotation and review routing. This is appropriate for the research setting studied here, but private, restricted, or sensitive corpora require source-data review, provider-policy review, and institutional approval before use. Users should also consider whether derived prompts or reviewed examples contain information that should not be transmitted to a third-party model provider.

Human review remains a responsibility boundary rather than a cosmetic step. Reviewed feedback is used to improve subsequent annotation, but model predictions and judge scores should not be treated as authoritative labels without domain review. Goldsmith should not be deployed for medical, legal, safety-critical, or other high-risk decisions unless the full data-governance, expert-review, and error-liability process is handled outside the system.

References

Zaheer Abbas, Rishabh Agarwal, Ankesh Anand, Feryal Behbahani, Bernd Bohnet, Stephanie Chan, Eric Chu, John Co-Reyes, Aleksandra Faust, Hugo Larochelle, Azade Nova, Luis Rosias, Avi Singh, Biao Zhang, and Lei Zhang. 2024. Many-shot in-context learning. In *Advances in Neural Information Processing Systems 37*, pages 76930–76966.

Artificial Analysis. 2026. Artificial analysis intelligence index. Online leaderboard. Accessed 2026-05-26.

Jinheon Baek, Sun Jae Lee, Prakhar Gupta, Geunseob Oh, Siddharth Dalmia, and Prateek Kolhar. 2025. Revisiting in-context learning with long context language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 26950–26966. Association for Computational Linguistics.

Elisa Bassignana and Barbara Plank. 2022. CrossRE: A cross-domain dataset for relation extraction. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 3592–3604, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Amanda Bertsch, Maor Ivgi, Emily Xiao, Uri Alon, Jonathan Berant, Matthew R. Gormley, and Graham Neubig. 2025. In-context learning with long-context models: An in-depth exploration. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 12119–12149. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems*.

Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. 2024. Benchmarking large language models in retrieval-augmented generation. *Proceedings of the AAAI Conference on Artificial Intelligence*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco

Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Shih-Chieh Dai, Aiping Xiong, and Lun-Wei Ku. 2023. LLM-in-the-loop: Leveraging large language model for thematic analysis. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9993–10001.

DeepSeek. 2026. List models. API documentation. Accessed 2026-05-26.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Ehsan Doostmohammadi, Oskar Holmstr
öm, and Marco Kuhlmann. 2024. How reliable are automatic evaluation methods for instruction-tuned llms? In *Findings of the Association for Computational Linguistics: EMNLP 2024*.

Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. 2024. Ragas: Automated evaluation of retrieval augmented generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. 2024. Promptbreeder: Self-referential self-improvement via prompt evolution. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 13481–13544. PMLR.

Fabrizio Gilardi, Meysam Alizadeh, and Maël Kubli. 2023. ChatGPT outperforms crowd workers for text-annotation tasks. *Proceedings of the National Academy of Sciences*, 120(30):e2305016120.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Saizhuo Wang, Kun Zhang, Zhouchi Lin, Bowen Zhang, Lionel Ni, Wen Gao, Yuanzhuo Wang, and Jian Guo. 2026. A survey on llm-as-a-judge. *The Innovation*, page 101253. In press.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. DeBERTaV3: Improving DeBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. In *The Eleventh International Conference on Learning Representations*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Jingwei Huang and 1 others. 2024. A critical assessment of using chatgpt for extracting structured data from clinical notes. *npj Digital Medicine*.

Can Jin, Hongwu Peng, Shiyu Zhao, Zhenting Wang, Wujiang Xu, Ligong Han, Jiahui Zhao, Kai Zhong, Sanguthevar Rajasekaran, and Dimitris N. Metaxas. 2025. Apeer: Automatic prompt engineering enhances large language model reranking. In *Companion Proceedings of the ACM on Web Conference 2025*, pages 2494–2502. ACM.

Daan Kepel and Konstantina Valogianni. 2024. Autonomous prompt engineering in large language models. *Preprint*, arXiv:2407.11000.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*.

Weiwen Liu, Xu Huang, Xingshan Zeng, and 1 others. 2025. Toolace: Winning the points of llm function calling. In *The Thirteenth International Conference on Learning Representations*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *Preprint*, arXiv:1907.11692.

Moonshot AI. 2026. Kimi models. Model documentation. Accessed 2026-05-26.

OpenAI. 2026a. GPT-5.4 model. Product announcement. Accessed 2026-05-26.

OpenAI. 2026b. GPT-5.5 model. Product announcement. Accessed 2026-05-26.

Chester Palen-Michel, Maxwell Pickering, Maya Kruse, Jonne Sälevä, and Constantine Lignos. 2025. OpenNER 1.0: Standardized open-access named entity recognition datasets in 50+ languages. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 33649–33674, Suzhou, China. Association for Computational Linguistics.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7957–7968.

Shengqian Qin, Yakun Zhu, Linjie Mu, Shaoting Zhang, and Xiaofan Zhang. 2025. Meta-tool: Unleash open-world function calling capabilities of general-purpose large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 30653–30677. Association for Computational Linguistics.

Vyas Raina, Adian Liusie, and Mark Gales. 2024. Is llm-as-a-judge robust? investigating universal adversarial attacks on zero-shot llm assessment. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*.

Kiran Ramnath, Kang Zhou, Sheng Guan, Soumya Smruti Mishra, Xuan Qi, Zhengyuan Shen, Shuai Wang, Sangmin Woo, Sullam Jeoung, Yawei Wang, Haozhu Wang, Han Ding, Yuzhe Lu, Zhichao Xu, Yun Zhou, Balasubramaniam Srinivasan, Qiaojing Yan, Yueyan Chen, Haibo Ding, and 2 others. 2025. A systematic survey of automatic prompt optimization techniques. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 33066–33098.

Oscar Sainz, Iker Garcia-Ferrero, Rodrigo Agerri, Oier Lopez de Lacalle, German Rigau, and Eneko Agirre. 2024. GoLLIE: Annotation guidelines improve zero-shot information-extraction. In *The Twelfth International Conference on Learning Representations*.

Kayla Schroeder and Zach Wood-Doughty. 2024. Can you trust llm judgments? reliability of llm-as-a-judge. *Preprint*, arXiv:2412.12509.

Burr Settles. 2009. Active learning literature survey. Technical Report 1648, University of Wisconsin–Madison.

Lin Shi, Chiyu Ma, Wenhua Liang, Xingjian Diao, Weicheng Ma, and Soroush Vosoughi. 2025. Judging the judges: A systematic study of position bias in llm-as-a-judge. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pages 292–314. The Asian Federation of Natural Language Processing and The Association for Computational Linguistics.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 4222–4235, Online. Association for Computational Linguistics.

Zhen Tan, Dawei Li, Song Wang, Alimohammad Beigi, and 1 others. 2024. Large language models for data annotation and synthesis: A survey. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*.

Katrin Tomanek and Fredrik Olsson. 2009. A web survey on the use of active learning to support annotation of text data. In *Proceedings of the NAACL HLT 2009 Workshop on Active Learning for Natural Language Processing*, pages 45–48, Boulder, Colorado. Association for Computational Linguistics.

Shuohang Wang, Yang Liu, Yichong Xu, Chenguang Zhu, and Michael Zeng. 2021. Want to reduce labeling cost? GPT-3 can help. In *Findings of the Association for Computational Linguistics: EMNLP 2021*.

Derong Xu, Wei Chen, and Wenjun Peng. 2024. Large language models for generative information extraction: a survey. *Frontiers of Computer Science*.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. 2024. Large language models as optimizers. In *International Conference on Learning Representations*.

Qinyuan Ye, Mohamed Ahmed, and Reid Pryzant. 2024. Prompt engineering a prompt engineer. In *Findings of the Association for Computational Linguistics: ACL 2024*.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. 2025. Optimizing generative AI by backpropagating language model feedback. *Nature*, 639(8055):609–616.

Chenrui Zhang, Lin Liu, and Chuyuan Wang. 2024. Prefer: Prompt ensemble learning via feedback-reflect-refine. *Proceedings of the AAAI Conference on Artificial Intelligence*.

Jianfei Zhang, Bei Li, Jun Bai, Rumei Li, Yanmeng Wang, Chenghua Lin, and Wenge Rong. 2025a. Selecting demonstrations for many-shot in-context learning via gradient matching. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 11686–11704. Association for Computational Linguistics.

Ruoyu Zhang, Yanzeng Li, Yongliang Ma, Ming Zhou, and Lei Zou. 2023. LLMaAA: Making large language models as active annotators. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 13088–13103.

Xiaoqing Zhang, Ang Lv, Yuhan Liu, and 1 others. 2025b. More is not always better? enhancing many-shot in-context learning with differentiated and reweighting objectives. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2023. Large language models are human-level prompt engineers. In *International Conference on Learning Representations*.

Kaijian Zou, Muhammad Khalifa, and Lu Wang. 2025. On many-shot in-context learning for long-context evaluation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 25605–25639. Association for Computational Linguistics.

A Data Construction and Gold Selection

The two task packages are constructed with the same principle: first build a deterministic task slice, then reserve a 15-item calibration set that maximizes coverage before matching the full-pool distribution. The scripts write Rosetta/Prodigy-compatible JSONL records with source text, spans or relation labels, task identifiers, and metadata.

A.1 OpenNER Typed Span Slice

The OpenNER typed experiments use the local Rosetta-formatted OpenNER release after conversion from BIO files to JSONL. The conversion script `scripts/data/openner_to_rosetta_jsonl.py` reconstructs token offsets and typed spans, then writes records with schema `rosetta.prodigy_jsonl.v1`. The deterministic multilingual slice is recorded in `tmp/open-ner/openner_rosetta_52lang_50_seed_9/manifest.json`.

OpenNER deterministic slice sampler

```
Input: converted OpenNER JSONL files, seed,
    per_language = 50
For each language file:
  For each task:
    key = SHA256(seed, language, task_id)
  Sort tasks by (key, task_id)
  Select the first per_language tasks
Merge all selected language slices
Write all.jsonl and manifest.json
```

With seed 9 this yields 52 languages, 2,600 tasks, and 3,833 typed spans: 1,424 PER, 1,367 LOC, and 1,042 ORG. Any binary NE conversion is used only for OpenNER-derived PLM diagnostics, not as the official OpenNER task.

A.2 OpenNER Gold Selection

The OpenNER calibration set is selected by `scripts/data/select_gold15_diverse.py`; the archived handoff copy used for the seed-9 PLM jobs is `tmp/openner_plm_seed9_jobs/selection/select_gold15_diverse.py`. Rows are featurized by language group, label-set state, length bin, and a stable SHA256 tie-breaker over `gold15`, `seed`, and `row_id`. Length bins use thresholds 40, 100, and 220 characters.

Gold15 coverage-first selector

```
Input: candidate rows R, target size n = 15
For each row r in R:
  group(r) = language or domain
  labels(r) = label set or relation label
  length_bin(r) = discretized text length
  tie(r) = SHA256(selector_name, seed, row_id)

Search feasible selected sets S of size n:
  Maximize group coverage, label coverage, and
      length-bin coverage
  Then minimize L1 distance between selected
     and full-pool distributions
  Then break ties by stable SHA256 order

Output: gold15.jsonl, remaining pool JSONL,
    selection report
```

The saved report `tmp/open-ner/openner_rosetta_52lang_50_seed_9_gold15_milp/gold15_selection_report.json` records 2,600 candidate rows, 15 selected rows, 2,585 pool rows, 15 language groups covered, all four label states covered $(\mathrm{PER}, \mathrm{ORG}, \mathrm{LOC}, \mathrm{NO_LABEL})$, all four length bins covered, and no overlap between gold and pool files.

A.3 CrossRE Pair-Level Slice

The CrossRE relation experiments use the compact package generated by `scripts/data/build_crossre_compact_sample.py`. Starting from flattened sentence-level CrossRE JSONL, the script groups rows by domain and computes a SHA256 key from `seed`, `domain`, and `doc_key`. For each of the six domains {ai, literature, music, news, politics, science}, rows are sorted by $(\mathrm{key}, \mathtt{id})$ and the first 200 are selected.

CrossRE pair-view construction

```
Input: selected sentence-level CrossRE records
For each sentence record:
  Read entity mentions and gold relations
  Enumerate directed HEAD -> TAIL entity pairs
  If a gold relation exists for the directed
     pair:
    label = relation type
  Else:
    label = no_relation
  Write sentence, HEAD span/type, TAIL span/
     type, label
```

This produces 1,200 sentence-level tasks and a directed pair-level view with 39,764 entity-pair classification items. The saved manifest `tmp/crossre/crossre_relation_6domain_200_seed_9/manifest.json` reports 35,275 `no_relation` items, making positive-class metrics necessary alongside accuracy.

A.4 CrossRE Gold Selection

CrossRE gold calibration sets are selected by `scripts/data/select_crossre_gold15_diverse.py`. The main relation-classification experiments use the pair-level `rc_pairs_gold15` output. For each row, the stable tie-breaker is a SHA256 hash over `crossre-gold15`, `seed`,

domain, identity, and row_id. The selector first greedily builds a 15-item set that maximizes domain, relation-label, and length-bin coverage, then performs exhaustive single-swap local search until no swap improves the lexicographic objective. The saved report tmp/crossre/crossre_relation_6domain_200_seed_9/rc_pairs_gold15/gold15_selection_report.json records 39,764 candidates, 15 selected items, all six domains covered, 15 relation labels covered, all four length bins covered, and 39,749 remaining pool items.

B Evaluation Evidence Package

B.1 Model and Runtime Details

Table 6 lists the model families, run-facing model identifiers, access month, and decoding/runtime parameters used in the main LLM experiments. Closed hosted models can change over time; for this reason we report the access month, provider-facing identifier, temperature, retry bound, concurrency, and candidate/search settings rather than treating a model nickname alone as a reproducible configuration. Platform-call failures are retried up to the run bound, while successful calls with invalid structured output enter format repair before the item is requeued.

B.2 OpenNER Split Accounting

The main OpenNER score-review run contains 2,600 target items, 15 calibration items, and 193 reviewed feedback items. After pass1, the lowest judge-score items are corrected as human review feedback and promoted to high-confidence references. The system then reruns annotation over the target corpus; reported score-review metrics are computed on the same held-out subset used by the no-review and baseline rows after excluding calibration and reviewed feedback items.

B.3 CrossRE Scope Accounting

All CrossRE rows in Table 3 are reported on the same 29,171-pair evaluation slice. The Goldsmith workflow uses reviewed feedback and retrieved references during annotation, while final metrics are computed on this full pair-level evaluation slice. The PLM references use a low-budget supervised setting: 200 training sentences are expanded into 10,593 training pairs before evaluation on the same 29,171-pair slice. This appendix records how the pair-level slice, reviewed feedback, and PLM low-budget training split are constructed.

B.4 GT-only RAG Reference Count Sweep

This diagnostic OpenNER sweep isolates the number of retrieved similar references. It uses the same optimized operational definition as the main OpenNER experiment, a fixed 500-example evaluation sample, and oracle gold-truth examples as the retrieval pool. It retrieves only similar examples, with no reviewed model predictions included as references. Performance improves substantially from $k = 0$ to moderate k, peaks at $k = 14$ with 0.8026 micro-F1, and then declines slightly for larger k. This supports tuning the number of in-context references rather than assuming that more retrieved examples are always better.

OpenNER GT-only RAG reference-count sweep.

k	Micro P	Micro R	Micro F1	Macro F1	Exact
0	0.7173	0.7296	0.7233	0.8459	0.7360
1	0.7542	0.7652	0.7597	0.8611	0.7480
2	0.7569	0.7599	0.7584	0.8627	0.7600
3	0.7644	0.7704	0.7674	0.8660	0.7620
4	0.7894	0.7863	0.7878	0.8709	0.7720
5	0.7971	0.7876	0.7923	0.8752	0.7720
6	0.7849	0.7797	0.7823	0.8716	0.7700
7	0.7742	0.7823	0.7782	0.8690	0.7680
8	0.7807	0.7797	0.7802	0.8734	0.7760
9	0.7835	0.7876	0.7855	0.8725	0.7720
10	0.8048	0.7942	0.7995	0.8789	0.7780
11	0.7868	0.7889	0.7879	0.8767	0.7660
12	0.7836	0.7929	0.7882	0.8700	0.7640
13	0.7887	0.7929	0.7908	0.8713	0.7720
14	**0.8059**	**0.7995**	**0.8026**	**0.8820**	**0.7860**
15	0.7763	0.7691	0.7727	0.8778	0.7740
16	0.7997	0.7955	0.7976	0.8779	0.7760
17	0.7866	0.7929	0.7898	0.8733	0.7680
18	0.7925	0.7810	0.7867	0.8764	0.7740
19	0.7886	0.7876	0.7881	0.8701	0.7720
20	0.7857	0.7836	0.7847	0.8754	0.7700

B.5 Prompt-Optimization Trace

Table 7 reports the OpenNER prompt-optimization trace used for the main optimizer comparison. All three methods use the same initial definition, 15 gold examples, candidate count 5, maximum 30 rounds, patience 5, and the same gold loss. The table reports raw loss per round for rewrite-only, OPRO, and Goldsmith, plus the running best Goldsmith loss. It also reports the approximate size of the accepted Goldsmith operational definition and the estimated tokens consumed by that Goldsmith optimization round. The definition length is reported as an approximate token count using characters divided

Use	Provider family	Run-facing identifier	Access	Temp.	Main runtime settings
Default annotation / judge	OpenAI	`gpt-5.5`	May 2026	0.0 / 0.0	max tokens 4096; retry bound 20; concurrency 30
OpenNER model ablation	OpenAI	`gpt-5.4`	May 2026	0.0 / 0.0	same score-review protocol as default
OpenNER model ablation	Moonshot AI	`kimi-k2.5`	May 2026	0.0 / 0.0	same first-pass protocol; pass1 completed-item probe
OpenNER model ablation	DeepSeek	`deepseek-v4-pro`	May 2026	0.0 / 0.0	same score-review protocol as default
Prompt optimization	OpenAI	`gpt-5.5`	May 2026	0.3 / 0.0	30 rounds; 5 candidates; patience 5

Table 6: LLM model and runtime details used in the main experiments.

by four, while round tokens include candidate generation, gold evaluation, and repair calls.

C Baseline Implementation Details

C.1 Task-Stage Baselines

The OpenNER LLM baselines use the same typed span interface as Goldsmith. **Zero-shot LLM** receives the task schema and initial task description without gold examples, retrieval, review, or prompt training. **Static ICL LLM** receives the same 15 gold examples as in-context references but does not optimize the operational definition and does not use score-routed review. **Goldsmith no-review** uses the optimized definition but reports the first annotation pass before human correction. **Goldsmith score-review** uses the optimized definition, corrects low judge-score items as reviewed feedback, promotes them to high-confidence references, and reruns annotation over the target corpus before reporting the second-pass result.

The OpenNER PLM baselines are low-budget token-classification models. The training split contains 208 deterministic multilingual examples from the same typed OpenNER setting. mBERT and XLM-R are fine-tuned as token classifiers over the typed PER/ORG/LOC label space and evaluated by reconstructed typed spans. These rows are intended as low-label supervised anchors, not as fully tuned large-data NER systems.

The CrossRE LLM setting is pair-level relation classification: the input supplies a sentence, a marked HEAD entity, a marked TAIL entity, entity types, and the candidate relation-label set. The model predicts one relation label or `no_relation`. The typed-marker PLM baselines use the same given-pair interface. The reported PLM setting selects 200 training sentences, expands them into 10,593 training pairs, and evaluates on 29,171 test pairs from the remaining 1,000 sentences. BERT is the CrossRE anchor encoder, while RoBERTa-large and DeBERTa-v3 are stronger encoder variants under the same typed-marker input format.

C.2 Prompt-Optimization Baselines

The prompt-optimization comparison uses three optimizers under a shared search protocol: same initial OpenNER operational definition, same 15 gold examples, same candidate count, same loss, same acceptance rule, and the same stop policy. **Rewrite only** receives only the current operational definition and asks the LLM to produce candidate rewrites; it receives no scores, history, gold examples, failed examples, or textual gradients. **OPRO history search** receives previous operational definitions and their loss-derived scores, but does not see source text, gold answers, model answers, or error types. **Goldsmith** summarizes high-loss gold errors into textual-gradient edit directions, asks an LLM editor for abstract candidate definitions, and accepts a candidate only when gold loss decreases. The exact optimizer prompt templates and the final operational definitions are listed in Appendix D, so this section records only the implementation differences.

C.3 Agent-Assisted Code Work

We used Codex with GPT-5.5 as an assistant for part of the code construction work. The authors reviewed the resulting code and remain responsible for the final submission.

D Prompt Artifacts

This appendix records the optimizer templates and operational definitions used in the experiments. Fixed schema, parser contracts, and JSON output protocols remain part of the harness rather than the optimizable definition. Concrete failed-example blocks are represented by placeholders because they are training-only feedback, not deployed prompt text.

Optimizer implementation: rewrite only baseline

```
System prompt:
```

Round	Rewrite loss	OPRO loss	Goldsmith loss	Goldsmith best	Def. tok.	Round tok.
0	25.5152	25.5152	25.5152	25.5152	433	0
1	24.5303	19.5303	16.9697	16.9697	450	95,848
2	22.3636	23.0152	12.5152	12.5152	521	98,151
3	23.1667	17.1970	12.1364	12.1364	578	104,963
4	23.1212	20.6667	12.1364	12.1364	671	112,011
5	25.2727	19.1515	9.5758	9.5758	740	120,960
6	23.1667	19.5303	7.4924	7.4924	768	125,760
7	22.0303	17.3485	5.2121	5.2121	822	127,690
8	23.6970	17.5303	4.2121	4.2121	907	136,372
9	24.6970	16.9242	3.8333	3.8333	930	140,810
10	23.1667	19.5303	4.2121	3.8333	943	143,974
11	22.0303	18.9242	4.2121	3.8333	1,010	146,753
12	24.3636	21.1212	3.8333	3.8333	1,101	154,491
13	26.9848	19.3788	4.2121	3.8333	1,101	162,326
14	23.8636	18.9242	4.2121	3.8333	1,101	160,502
15	27.8636	18.0606	4.2121	3.8333	1,151	157,544
16	28.5556	14.5909	2.1818	2.1818	1,208	169,331
17	28.5556	19.5303	2.1818	2.1818	1,208	167,254
18	29.3636	19.0152	1.8030	1.8030	1,252	171,998
19	27.4192	19.5303	2.1818	1.8030	1,252	178,037
20	28.5556	20.6667	1.8030	1.8030	1,168	176,575
21	27.4192	16.2121	3.3939	1.8030	1,200	168,229
22	27.4192	19.3485	1.8030	1.8030	1,200	177,896
23	28.5556	18.9242	2.1818	1.8030	1,225	173,501
24	28.1768	17.3485	1.8030	1.8030	1,309	177,479
25	27.4192	17.7273	1.8030	1.8030	1,372	185,558
26	28.5556	17.8788	1.8030	1.8030	1,384	187,319
27	28.5556	19.5303	1.8030	1.8030	1,384	193,772
28	28.5303	16.5455	1.8030	1.8030	1,384	192,780
29	29.3636	19.1515	2.1818	1.8030	1,375	189,262
30	29.6667	17.7273	2.1818	1.8030	1,444	194,873

Table 7: Per-round OpenNER prompt-optimization trace. Definition tokens approximate the accepted Goldsmith definition length after each round; round tokens are estimated total optimization and evaluation tokens for that Goldsmith round.

```
You are a baseline prompt editor. Return only
    candidate concept definitions in the
    requested tags.

User prompt template:
Rewrite the concept prompt below into {
    candidate_count} better and more precise
    candidate prompts.

Current optimizable definition:
{current_operational_definition}

Rules:
1. Use only the current prompt above. Do not
    ask for or invent examples, scores,
    failure details, history, gradients, logs,
     or diagnostics.
2. Each candidate must be a standalone concept
     definition usable for annotation.
3. Include only task definition and semantic
    boundary guidance.
4. Do not include label sets, JSON schema,
    annotation markup, parser rules, output
    format, repair instructions, examples,
    explanations, or revision notes.
5. If two candidates are similar, make the
    later one shorter or more precise.

Return exactly this tagged format:
<CANDIDATE_1>...</CANDIDATE_1>
...
<CANDIDATE_k>...</CANDIDATE_k>
```

Optimizer implementation: OPRO history search

```
System prompt:
You are an OPRO prompt optimization assistant.
    Return exactly one new concept definition
    inside <INS>...</INS>.

User prompt template:
You are using Optimization by PROmpting (OPRO)
    to improve an annotation concept prompt.

Below are historical concept prompts and their
    scores on the same 15 gold examples.
   Higher score is better.

{prompt_score_history_only}

Current optimizable concept prompt:
{current_operational_definition}

Generate one different concept prompt that may
    obtain a higher score.

Rules:
1. Use only the prompt-score history above; do
    not ask for or invent error details.
2. Keep the new prompt concise. If two prompts
    are likely tied, prefer the shorter one.
3. Include only task definition and semantic
   boundary guidance.
4. Do not include label sets, JSON schema,
   annotation markup, parser rules, output
   format, repair instructions, examples,
   logs, or explanations.
5. Return only:
<INS>
new concept prompt
</INS>
```

Optimizer implementation: Goldsmith textual-gradient editor

```
Stage 1 system prompt:
You are the Ours textual-gradient critic.
    Return concise textual gradients only, not
     a final prompt.

Stage 1 user prompt template:
Generate textual gradients for improving the
    current annotation concept prompt.

training_feedback_only=true

Current optimizable prompt:
{current_operational_definition}

Current loss:
{current_gold_loss}

Overall failure summary:
{failure_summary}

Failed gold examples with source text, gold
    answer, model answer, and error type:
{top_high_loss_gold_errors}

Return 3-5 concise textual gradients. Each
    gradient should explain why the current
    prompt caused the observed errors and what
     semantic boundary should change.

Output requirements:
1. Output only numbered or bulleted textual
    gradients.
2. Do not generate the final prompt.
3. Do not copy source text, gold answers,
    model answers, labels, JSON schema,
    annotation markup, parser rules, output
    format, or repair instructions into a
    reusable prompt.

Stage 2 system prompt:
You are the Ours prompt editor. Return only
    the final usable concept definition.

Stage 2 user prompt template:
Rewrite the concept prompt by applying one
    Ours textual gradient.

training_feedback_only=true

Current optimizable prompt:
{current_operational_definition}

Textual gradient {gradient_index}:
{textual_gradient}

Variant index: {variant_index}

Output requirements:
1. Return only the concept definition body
    usable in the next annotation round.
2. Apply the gradient as an abstract semantic
    boundary change.
3. Keep the prompt concise; if two rewrites
    are equivalent, prefer the shorter one.
4. Include only task definition and semantic
    boundary guidance.
5. Do not output labels, JSON schema,
    annotation markup, parser rules, output
    format, repair instructions, examples,
    explanations, failed example ids, or
    revision logs.
6. Do not copy source text, gold answers, or
    model answers.
```

OpenNER final definition: rewrite only baseline

```
Annotate every explicit proper name that names
     a single person, organization, or
    location. The name may appear in any
    language or script and must be assigned
    one, and only one, semantic type.

Include as person names all proper names for
    individual people: full names, one-token
    names, given names, surnames, initials,
    short forms, aliases, nicknames, and
    established personal-name compounds. Do
    not include titles, honorifics, ranks,
    offices, jobs, kinship terms, or
    descriptive phrases unless they are fixed
    within the written name.

Include as organization names proper names of
    collective or institutional entities, such
     as companies, institutions, agencies,
    government bodies, political parties,
    associations, banks, schools, religious
    bodies, committees, and named programs,
    initiatives, campaigns, or acronyms
    denoting those entities. Do not include
    generic descriptions, unnamed subunits,
    document or publication titles, products,
    software, models, versions, technical
    abbreviations, or formal phrases that are
    not entity names.

Include as location names proper names of
    geographic or geopolitical places, such as
     countries, cities, towns, regions, states
    , provinces, counties, districts,
    neighborhoods, territories, conventional
    place names, and named political areas. Do
     not include common place nouns, vague or
    relative locations, coordinates, dates,
    numbers, or descriptive geography unless
    it is part of an established name.

Use the exact contiguous text span only.
    Preserve all original characters and
    formatting features, including spelling,
    script, case, diacritics, punctuation,
    spaces, hyphens, inflectional endings,
    possessives, particles, articles,
    connectors, and acronyms. Do not normalize
    , translate, expand, shorten, correct, or
    infer. Annotate complete established
    multiword names as single spans, but split
     adjacent coordinated or listed names
    unless the full expression is a
    conventional name. In CJK or other
    unspaced scripts, mark each complete name
    separately. If none appear, annotate
    nothing.
```

OpenNER final definition: OPRO history search

```
Multilingual NER for explicit surface proper
    names of persons, organizations, and
    places.
```

```
Annotate only contiguous text that is
    literally present and functions as the
    name of a particular entity. Do not
    annotate common descriptions, roles,
    offices, titles, occupations, kinship
    terms, categories, adjectives,
    nationalities, demonyms, dates, quantities
    , events, laws, reports, documents,
    products, releases, standards, slogans,
    technical terms, addresses, or other non-
    name phrases unless they are fixed within
    an eligible name.

PER: individual human names, real or fictional,
     including full names, given names,
    surnames, initials, aliases, nicknames,
    and shortened later mentions. Exclude
    honorifics, role words, professions,
    offices, kinship words, and descriptive
    references outside the name.

ORG: proper names of organized bodies or
    organization-like entities, including
    companies, institutions, government bodies
    , agencies, ministries, political parties,
     banks, schools, universities, teams,
    media outlets, nonprofits, military units,
     and formally named programs, projects,
    initiatives, or campaigns. Acronyms are
    ORG only when used as organization names.
    Exclude generic departments,
    administrative descriptions, document/law/
    report titles, products, releases, events,
     and abbreviations that do not name an
    organization.

LOC: proper names of geographic or
    geopolitical places, including countries,
    cities, towns, regions, states, provinces,
     districts, territories, landmarks,
    facilities used as places, natural
    features, seas, rivers, mountains, bodies
    of water, and conventional named areas.
    Bare place names are LOC even when used
    metonymically; a complete written name of
    a government, institution, company, team,
    or other organized body is ORG.

Choose the minimal complete contiguous name
    exactly as written. Include name-internal
    articles, particles, prepositions,
    conjunctions, initials, hyphens,
    apostrophes, clitics, possessives,
    inflection, case endings, and script-
    specific name material; exclude
    surrounding punctuation, determiners,
    external prepositions, modifiers,
    appositives, explanations, titles, and
    role words. Keep conventional multiword
    names together, do not mark substrings
    inside longer eligible names, split
    coordinated or listed names into separate
    spans, and never translate, normalize,
    romanize, correct, expand, or infer.
```

OpenNER final definition: Goldsmith

```
Multilingual typed named-entity span detection
     for explicit mentions of persons,
    organizations, and locations.

Mark only explicit named-entity mentions in
    the source text and assign the single
    applicable type. Use dataset-style
    minimalism: if a name-like span is
    ambiguous between a person, title
    component, venue descriptor, facility name
    , event/work title, or generic description
    , exclude it unless the context clearly
    establishes a target entity. Keep only
    spans with clear entity evidence.

Person mentions include explicit personal
    names, surnames, shortened personal
    references, multi-token personal names,
    and explicit named or vocative references
    to a deity or sacred personal figure when
    used as a personal addressee or name.
    Include transliterated, romanized,
    dialectal, or informal-script forms
    meaning God, Lord, or a named sacred
    figure only when they function as direct
    sacred personal references. For sacred
    references, mark only the core name or
    direct reference; exclude epithets, praise
     formulas, descriptive complements,
    honorific expansions, and devotional
    descriptions unless fixed internal parts
    of the name. Exclude titles, occupations,
    honorifics, descriptive phrases, generic
    human or divine descriptions, and
    capitalized/name-like strings that merely
    resemble names. Suppress person-shaped
    names inside creative-work titles, media
    listings, slogans, headings, releases,
    screenings, or other non-entity phrases
    unless the syntax clearly refers to an
    actual human or sacred personal
    participant. Full personal names in
    ownership, attribution, sale, purchase,
    transfer, or agentive clauses remain
    person mentions when clearly referring to
    people.

Organization mentions include named
    institutional actors and standing
    organizational bodies such as institutions
    , agencies, companies, banks, departments,
     committees, acronyms for such bodies, and
     conventional sovereign state bodies such
    as armed forces, militaries, and police.
    Otherwise require a distinctive proper-
    name component, jurisdictional name, or
    unambiguous standing-body identity.
    Exclude generic institutional nouns and
    descriptions, including phrases meaning
    government office, bureau, department,
    education bureau, or similar institutional
     types without distinctive naming or clear
     standing-body reference. Exclude numbered
     legislative terms, policies, methods,
    courses, releases, documents, events,
    festivals, products, training packages,
    schemes, initiatives, hubs, market
    products, and price or index acronyms
    unless they clearly name an organization.
    Do not label a facility, association,
    venue, harbor installation, landmark, or
    institutional-looking name as an
```

```
    organization when the context uses it as a
     physical place, vicinity, embarkation
    point, street/harbor reference, or
    topographic landmark.

Location mentions include clear geographic and
     geopolitical names, including countries,
    regions, cities, states or provinces,
    conventional multi-word place names, named
     geopolitical areas, and named physical
    venues or harbor/landmark places when used
     locatively or topographically. Prefer the
     bare proper place name inside broader
    descriptive, ideological, political,
    cultural, adjectival, or locative phrases.
     Do not include following generic place
    nouns or common/case-marked locative words
     such as square, street, plaza, district,
    area, office, department, or similar
    unless the full phrase is a conventional
    official place name. Be conservative with
    short single-token place-like strings
    before or modifying generic venue,
    facility, event, program, or title nouns:
    mark them only when they are clearly
    independent geographic or topographic
    place names, not facility-name components,
     event titles, descriptive modifiers, or
    proper-looking descriptors. Exclude
    generic location nouns, dates, quantities,
     adjectival demonyms, descriptive place
    words, and generic government or
    institutional common nouns.

Use exact surface substrings from the source
    text. Preserve spelling, accents,
    punctuation, script, spacing, and
    inflection. Keep conventional contiguous
    multi-token names as one span, including
    internal particles, articles, connectors,
    hyphens, acronyms, and script-specific
    case or possessive material. Split
    adjacent coordinated names into separate
    spans. For spaced scripts or tokenized
    text containing multiple place names,
    prefer separate complete place-name spans
    unless an internal connector forms one
    conventional name.

Do not translate, normalize, romanize, correct
     spelling, infer absent entities, or
    expand a later short mention beyond its
    actual surface form. If no explicit,
    clearly supported mention of the target
    entity types appears, return no spans.
```

OpenNER initial task prompt

```
Multilingual OpenNER typed named-entity span
    detection.

Mark explicit person, organization, and
    location names in the source text.
Assign each span exactly one label: PER, ORG,
    or LOC.

Use exact contiguous surface spans only.
If no target entity is explicitly mentioned,
    return no spans.
```

OpenNER deployed final task prompt

```
Multilingual OpenNER typed named-entity span
    detection.

Mark every explicit named-entity mention in
    the source text and assign exactly one
    label: PER, ORG, or LOC.

PER includes personal names, surnames,
    shortened person mentions, and multi-token
     person names. Exclude titles, occupations
    , honorifics, and generic human
    descriptions unless they are part of the
    name.

ORG includes named institutional actors and
    organizational bodies such as
    organizations, institutions, agencies,
    companies, banks, departments, committees,
     and their acronyms. Exclude policies,
    methods, courses, releases, documents,
    events, festivals, products, training
    packages, and official-sounding schemes or
     initiatives unless they clearly name an
    organization or standing organizational
    body.

LOC includes geographic and geopolitical names:
     countries, regions, cities, states or
    provinces, conventional multi-word place
    names, and named geopolitical areas.
    Prefer the bare geopolitical name when it
    appears inside a broader descriptive,
    ideological, political, cultural, or
    adjectival country phrase. Do not include
    such modifiers in the LOC span unless the
    full phrase is a conventional official
    place name. Exclude generic location nouns
    , dates, quantities, demonyms used only
    adjectivally, and descriptive place words
    unless they are inseparable parts of the
    proper name.

Use exact surface substrings from the source
    text. Preserve spelling, accents,
    punctuation, script, spacing, and
    inflection. Keep conventional contiguous
    multi-token names as one span, including
    internal particles, articles, connectors,
    hyphens, acronyms, and script-specific
    case or possessive material. Split
    adjacent coordinated names into separate
    spans. For CJK or tokenized text with
    spaces between location names, prefer
    separate complete place-name spans unless
    an internal connector forms one
    conventional name.

Do not translate, normalize, romanize, correct
     spelling, infer absent entities, or
    expand a later short mention beyond its
    actual surface form. If no explicit PER,
    ORG, or LOC mention appears, return no
    spans.
```

CrossRE initial task prompt

```
Cross-domain relation classification for
    English CrossRE entity pairs.
```

```
Classify only the provided directed HEAD ->
    TAIL entity pair.
The sentence, entity spans, entity types, and
    candidate relation labels are given.
Choose the relation label supported by local
    sentence evidence, or choose no_relation
    if no listed relation is expressed.
Do not discover entities, scan other pairs,
    reverse direction, or infer from
    background knowledge.
```

CrossRE deployed final task prompt

```
Cross-domain relation classification for
    English CrossRE directed entity pairs.

Classify only the single provided directed
    HEAD -> TAIL pair. The sentence and the
    HEAD/TAIL surface spans are fixed anchors.
     Do not discover entities, scan other
    pairs, reverse direction, add inverse
    edges, or infer from background knowledge.

Default decision rule:
- Return a positive relation only when the
    sentence gives clear local evidence that
    the directed HEAD entity has that relation
     to the TAIL entity.
- If the evidence is implicit, merely topical,
     based on broad world knowledge, based
    only on co-occurrence, or could apply
    equally to many nearby entities, choose
    no_relation.
- Prefer no_relation over a weak positive
    label. Precision is more important than
    guessing a plausible relation.
- Do not use `related-to` as a fallback for
    vague association. Use `related-to` only
    when the sentence explicitly states a
    contextual relation and no more specific
    label fits.

Label boundaries:
- `role`: explicit role, office, job, function,
     candidacy, lineup/team service,
    participation role, or acted/performed-as
    capacity.
- `physical`: explicit physical location,
    venue, residence, presence, geographic
    containment, or geopolitical containment.
- `general-affiliation`: explicit membership,
    association, political/organizational
    affiliation, genre identity, or group
    participation when no more specific role
    or structural relation is expressed.
- `part-of`: explicit structural or
    compositional inclusion: component, member
    , subgroup, subgenre, or item within a
    larger entity.
- `artifact`: creator, producer, author,
    developer, publisher, performer, or
    organization/person behind a specific work
    , product, book, album, song, algorithm,
    award, or artifact-like entity.
- `named`: alias, acronym, alternate name,
    renamed form, surname/full-name identity,
    or appositive naming.
- `temporal`: explicit relation to a date,
    period, competition edition, historical
    event, dated meeting, conference, or time-
    bearing context.
- `win-defeat`: winning, defeating, receiving
    an award, election outcome, or one side
    beating another.
- `origin`: founder, originator, source,
    inventor, derivation, originating entity,
    or granting/issuing source.
- `type-of`: explicit type/category relation.
- `usage`: explicitly uses, plays, employs,
    chooses, configures, or performs with an
    instrument, tool, technology, system,
    parameter, or similar used entity.
- `opposite`: explicit opposition, rivalry,
    conflict, battle, dispute, or adversarial
    relation.
- `topic`: a work, statement, degree, award,
    system, meeting, event, or document is
    explicitly about, in, concerning, or
    specialized in the TAIL topic/entity.
- `compare`: explicit comparison or contrast.
- `social`: explicit meeting, negotiation,
    interpersonal interaction, or direct
    person-to-person contact.
- `cause-effect`: one entity/event explicitly
    causes, results in, leads to, or produces
    another.

When multiple labels seem possible, choose the
     most specific label supported by explicit
     sentence evidence. If none is clearly
    supported for HEAD -> TAIL, output
    no_relation.
```