# Authors Response Letter

We thank the reviewers for the constructive comments! Along with this detailed response, we have also attached a full version, including the changes color-coded to address reviewers questions: ([R1]: in red, [R2]: in blue, [R3]: in orange; [R4]: in teal; and for common issues, in purple). Below please find our answers.

(1) For the lack of discussion of key surveys on background knowledge (R1), categorization of matching methods by information types (R1), and more "traditional" OM works (R2): we have cited 2 papers as suggested, and enriched our discussion for related work in the full version. Unlike these works, KROMA performs LLM-enhanced OM with enriched semantic contexts from high-quality, external knowledge resources.

(2) Regarding the insufficient discussion of design choices (e.g. confidence thresholds and caching strategies) (R1), we have an LLM confidence calibration test in full version included. Our experimental study has found that the LLMs assigned high confidence scores to correct answers in 90% of cases, with only minor deviations. Accordingly, we set our confidence threshold to 8.5. Node pairs that don't pass refinement are placed in a queue for later processing.

(3) Regarding the comments on lack of definitions for several notations (R1, R2), we have included a detailed notation and abbreviation table in the full version (see "Appendix").

(4) On missing entries in the evaluation table (R1, R2, R4): it is because that many baselines and their exact test-set pairings were not available to be called or not published, so direct reproduction or comparison were not possible. Retraining the LLMs would be prohibitively expensive. We will track the availability of these resources to update Table 1.

(5) Per the reviewer comment on including BIO-ML track (R4): we clarify that we use TogetherAI and OpenAI APIs as off-the-shelf inference services on hosted, pretrained models, and we are not performing any fine-tuning or weight updates.

(6) Regarding the "missing" of Algorithm 2 (R2, R4), we do have the details, pseudo-code and analysis ready but omitted in submission due to space limit. We have included the algorithm with full details about the pseudocode, correctness, and cost analysis.

(7) For the reviewers' suggestions on more pairs from datasets to be selected for evaluation (R3): Thanks! We tried our best to ensure a fair and thorough evaluation of KROMA in diverse domains despite the high-cost of LLM inference. We plan to sample larger scale tests, evaluate and report the impact of workload sizes in the revised version, and will report more benchmarking results in future.

(8) Addressing missing concrete metrics on LLM inference overhead as well as prompt frequency (R1), we have added to our full version an experiment on KROMA efficiency based on communication cost. Overall, KROMA saves approximately 25% of the number of API calls, with the number of token usage remaining modest, with less than 500 tokens per call.

(9) Following the request for runtime estimates (R3, R4), we reported in our full version that for an ontology of 40 nodes and 1000 test pairs, the total time on involving LLM calls varies depending on web communication environment, prompts settings, and third-party LLM provider, hence we plan to report the average prompt size (a more proper measure as justified by [64]) which is approximately 470 tokens. On the other hand, the refinement cost is comparatively efficient (on average 80 seconds).

(10) With respect to the motivation of the choice of datasets (R3), we followed the convention and used the standard Ontology Alignment Evaluation Initiative (OAEI) benchmarks to enable direct and fair comparison with prior LLM-based methods. The five tracks chosen span a range of domains, showcasing KROMA's performance under diverse conditions.

(11) As for why only the source is sampled and not also the targets (R4), we arbitrarily designate one concept as the "source" for sampling and its target counterpart. We remark that the source and target roles are interchangeable w.l.o.g given our theoretical analysis, algorithms and test results.

(12) Regarding whether knowledge bases are truly external and their usage (R2, R4): We first note that our approach operates at the ontology level rather than performing full KG alignment. We enrich each concept's representation with contextual information by issuing SPARQL queries to external KBs (e.g., DBpedia, WordNet, etc ) to retrieve relevant facts. Our ablation study in [1] confirms that this external retrieval consistently improves alignment accuracy.

(13) On our LLM selection logic (R1), we limited our experiments to LLMs whose Massive Multitask Language Understanding (MMLU) performance does not exceed those used by our baselines. This ensures a fair comparison, especially since several LLMs used by baseline models are no longer supported by third party LLM providers.

(14) In response to lack of discussion on how determinism in LLM is enforced (R1), we ensure reproducibility by issuing every LLM call with a fixed temperature of 0.3 (balancing exploration and exploitation) and by using a constant random seed for sampling.

(15) To clarify the term "star-shaped SPARQL" (R1) and the full SPARQL query structure and details (R4): This declares a query pattern in which the concept serves as the central node, with its neighbors around it in a star-like topology as "facet search" [21]. See details in [1].

(16) Reviewers requested a visual example showing algorithm 1 refinement steps on a small ontology pair (R1). We have addressed this by providing details of a running case study based on offline refinement in the appendix of the full version.

(17) For online refinement use case (R4), in dynamic alignments, the matching may soon be outdated (as observed in MILA), KROMA's online refinement ingests incoming triples to maintain alignment accuracy as the knowledge graph evolves. In [1], we included a correctness analysis of the refinement algorithm,

detailing the conditions under which new triples trigger an additional refinement (see Details of Algorithm 2).

(18) We have fixed all typo and reorganized our paper in the full version.

# KROMA: Ontology Matching with Knowledge Retrieval and Large Language Models

No Author Given

No Institute Given

**Abstract.** Ontology Matching (OM) is a cornerstone task of semantic interoperability, yet existing systems often rely on handcrafted rules or specialized models with limited adaptability. We present KROMA, a novel OM framework that harnesses Large Language Models (LLMs) within a Retrieval-Augmented Generation (RAG) pipeline, to dynamically enrich the semantic context of OM tasks with structural, lexical, and definitional knowledge. To optimize both performance and efficiency, KROMA integrates a bisimilarity-based concept matching and a lightweight ontology refinement step, which prune candidate concepts and substantially reduce the communication overhead from invoking LLMs. Through experiments on multiple benchmark datasets, we show that integrating knowledge retrieval with context-augmented LLMs significantly enhances ontology matching—outperforming both classic OM systems and cutting-edge LLM-based approaches—while keeping communication overhead comparable. Our study highlights the feasibility and benefit of the proposed optimization techniques (targeted knowledge retrieval, prompt enrichment, and ontology refinement) for ontology matching at scale. Code is available at: https://anonymous.4open.science/r/kroma/

**Keywords:** Ontology Matching · Large Language Models · Retrieval Augmented Generation

## 1 Introduction

Ontologies have been routinely developed to unify and standardize knowledge representation to support data-driven applications. They allow researchers to harmonize terminologies and enhance knowledge and sharing in their fields. Ontologies can be classified according to their level of specificity, ranging from more abstract, general-purpose ontologies such as Basic Formal Ontology (BFO)[41] or DOLCE[6] down to more domain-oriented 'mid-level" ones, such as CheBi [14] in chemistry, Industrial Ontology Foundy (IOF) [17] or Common Core Ontology(CCO) [29]. Domain ontologies are data-driven, task-specific "low-level" ontologies, containing concepts from domain-specific data, such as Materials Data Science Ontology (MDS-Onto) [51]. To achieve broad usability, ontologies need to be aligned for better interoperability via ontology matching.

Ontology matching has been studied to find correspondence between terms that are semantically equivalent [59]. It is a cornerstone task to ensure semantic interoperability among terms originated from different sources. Ontology matching methods can be categorized to rule-based or structural-based (graph pattern or path-based) matching [10], matching with semantic similarity, machine learning-based approaches and hybrid methods. Linguistic (terminological) methods are often used for ontology matching tasks, ranging from simple string matching or embedding learning to advance counterparts based on natural language processing (NLP). Nevertheless, conventional rule- or structural-based methods are often restricted to certain use cases and hard to be generalized for new or unseen concepts. Learning-based approaches may on the other hand require high re-training process, for which abundant annotated or training data remains a luxury especially for *e.g.,* data-driven scientific research.

In recent years, several overviews of matching methods have been proposed [19,50,52]. Some methods follow Euzenat and Shvaiko's context-based view [19,52], while others stick to Rahm and Bernstein's schema-only approach [50]. Rahm et al. treat all background knowledge the same, but Euzenat and coauthors divide it into formal resources (which can be used by automated reasoners) and informal ones (which cannot). They also rate these resources by how broad, formal, and reliable they are [19].

Ontology matching is hard because ontologies hide the context and expert knowledge that they don't spell out in their schemas [44]. Since 2004, many matching systems have shown that using outside information can greatly improve alignments. For example, in the OAEI 2021 Anatomy task, LogMapBio tapped into domain-specific data and got much higher recall than the original LogMap [31]. But making these systems faster often means they use more memory, bandwidth, or CPU time [53]. So it's important to track not just how long they take but also how much memory they need. Going forward, we'll need matching methods that run well on devices with limited resources, like smartphones and other handhelds.

Meanwhile, the emerging Large Language Models (LLMs) have demonstrated remarkable versatility for NL understanding. LLMs are trained on vast and diverse corpora, endowing them with an understanding of language nuances and contextual subtleties. Extensive training enables them to capture semantic relationships and abstract patterns that are critical for aligning concepts across different data sources. A missing yet desirable opportunity is to investigate whether and how LLMs can be engaged to automate and improve ontology matching.

This paper introduces KROMA, a novel framework that exploits LLMs to enhance ontology matching. Unlike existing LLM-based methods that typically "outsource" ontology matching to LLMs with direct prompting (which may have low confidence and risk of hallucination), KROMA maintain a set of conceptually similar groups that are co-determined by concept similarity and LLMs, both guided by their "context" obtained via a runtime knowledge retrieval process. Moreover, the groups are further refined by a global ontological equivalence relation that incorporate structural equivalence.

**Contributions**. Our main contributions are summarized below.

(1) We propose a formulation of semantic equivalence relation in terms of a

class of bisimilar equivalence relation, and formally define the ontology structure, called concept graph, to be maintained (**Sections 2 and 3**). We justify our formulation by showing the existence of an "optimal" concept graph with minimality and uniqueness guarantee, subject to the bisimilar equivalence.

(2) We introduce KROMA, an LLM-enhanced ontology matching framework (**Section 4**). KROMA fine-tunes LLMs with prompts over enriched semantic contexts. Such contexts are obtained from knowledge retrieval, referencing high-quality, external knowledge resources.

(3) KROMA supports both offline and online matching, to "cold-start" from scratch, and to digest terms arriving from a stream of data, respectively. We introduce efficient algorithms to correctly construct and maintain the optimal concept graphs (**Section 5**). (a) The offline refinement performs a fast grouping process guided by the bisimilar equivalence, blending concept equivalence tests co-determined by semantic closeness and LLMs. (b) The online algorithm effectively incrementalizes its offline counterpart with fast delay time for continuous concept streams. Both algorithms are in low polynomial time, with optimality guarantee on the computed concept graphs.

(4) Using benchmarking ontologies and knowledge graphs, we experimentally verify the effectiveness and efficiency of KROMA (**Section 6**). We found that KROMA outperforms existing LLM-based methods by 10.95% on average, and the optimization of knowledge retrieval and refinement improves its accuracy by 6.65% and 2.68%, respectively.

**Related Work**. We summarize related work below.

*Large Language Models.* Large language models (LLMs) have advanced NLP by enabling parallel processing and capturing complex dependencies [60,55], which have scaled from GPT-1's 117M [48], GPT-2's 1.5B [49] to GPT-3's 175B [38] and GPT-4's 1.8T parameters [39]. Open-source models like Llama have grown to 405B [35], with Mistral Large (123B) [36] and DeepSeek V3 (671B) [12] also emerging. Recent advances in specialized reasoning LLMs (RLLMs) such as OpenAI's O1 and DeepSeek R1 have further propelled Long Chain-of-Thought reasoning—shifting from brief, linear Short CoT to deeper, iterative exploration, and yielded substantial gains in multidisciplinary tasks [40,13,54,57,8,46,63,33].

*Ontology Matching with LLMs.* Several methods have been developed to exploit LLMs for ontology matching. Early work focused on direct prompting LLMs for ontology matching. For example, [42] frame product matching as a yes/no query, and [37] feed entire source and target ontologies into ChatGPT—both achieving high recall on small OAEI conference-track tasks but suffering from low precision. Beyond these "direct-prompt" approaches, state-of-the-art LLM-OM systems fall into two main categories: (1) retrieval-augmented pipelines, which first retrieve top-$k$ candidates via embedding-based methods and then refine them with LLM prompts (e.g. LLM4OM leverages TF–IDF and SBERT retrievers across concept, concept-parent, and concept-children representations [22], while MILA adds a prioritized depth-first search step to confirm high-confidence matches before any LLM invocation [56]); and (2) prompt-engineering systems, which generate candidates via a high-precision matcher or inverted index and then apply targeted prompt templates to LLMs in a single step (e.g. OLaLa embeds SBERT candidates into MELT's prompting framework with both independent and multi-choice formulations [27], and LLMap uses binary yes/no prompts over concept labels plus structural context with Flan-T5-XXL or GPT-3.5 [25]).

## 2 Ontologies and Ontology Matching

| Notation | Description |
|---|---|
| $\mathcal{O} = (C, E)$ | ontology $\mathcal{O}$, $C$: set of concepts, $E$: set of relations |
| $\lvert\mathcal{O}\rvert$ | size of ontology $\mathcal{O}$; $\lvert\mathcal{O}\rvert = \lvert C\rvert + \lvert E\rvert$ |
| $\mathrm{r}(c)$ | rank of concept node $c$ |
| $c.I$ | ground set of concept node $c$ |
| $\mathcal{O}_s = (C_s, E_s)$, $\mathcal{O}_t = (C_t, E_t)$ | source and target ontology, respectively |
| $R_{\simeq}$ | ontological equivalence relation |
| $\mathcal{C}$ | equivalence partition of concept set $C_s \cup C_t$ |
| $\mathcal{G}_O = (V_{\mathcal{O}}, E_{\mathcal{O}})$ | concept graph $\mathcal{G}_O$, $V_O$: set of nodes, $E_O$: set of edges |
| $\Delta\mathcal{G}$ | newly arrive edges; edge updates in $\mathcal{G}$ |
| $[c] \in V_O$ | an equivalence class in $\mathcal{C}$ |
| $M$ | a language model |
| $\mathcal{M}$ | set of Large Language Model(s) |
| $q$ | a prompt query |
| $q(M)$ | a natural language answer from language model $M$ |
| $F(c, c')$ | concept similarity between two concepts $c$ and $c'$ |
| $q(c, c', \mathcal{M})$ | a natural language answer to $\mathcal{M}$ asking if $c \simeq c'$ |
| $\alpha \in (0, 1]$ | threshold for asserting concept similarity |
| $W = (O_s, O_t, F, \mathcal{M}, \alpha)$ | configuration input for ontology matching |
| $z_c$ | embedding of concept $c$ |
| $\mathbb{S}$ | set of candidate concept pairs with similarity scores |
| $\mathbb{C}$ | top-k pairs with highest similarity scores |

Table 1: Summary of Notations.

**Ontologies**. An ontology $O$ is a pair $(C, E)$, where $C$ is a finite set of concept names, and $E \subset C \times C$ is a set of relations. An ontology has a graph representation with a set of concept nodes $C$, and a set of edges $E$. We consider ontologies as directed acyclic graphs (DAGs).

In addition, each concept node (or simply "node") $c$ in $O$ carries the following auxiliary structure. (1) The *rank* of a node $c \in C$ is defined as: (a) $r(c) = 0$ if $c$ has no child, otherwise, (b) $r(c) = \max(r(c') + 1)$ for any child $c'$ of $c$ in $O$. (2) A *ground set* $c.\mathcal{I}$, refers to a set of entities from *e.g.,* external ontologies or knowledge bases that can be validated to belong to the concept $c$.

**Ontology Matching**. Given a source ontology $O_s = (C_s, E_s)$ and a target ontology $O_t = (C_t, E_t)$, an *ontological equivalence relation* $R_{\simeq} \subseteq C_s \times C_t$ is an equivalence relation that contains pairs of nodes $(c, c')$ that are considered to be "semantically equivalent". The relation $R_{\simeq}$ induces an equivalence partition $\mathcal{C}$ of

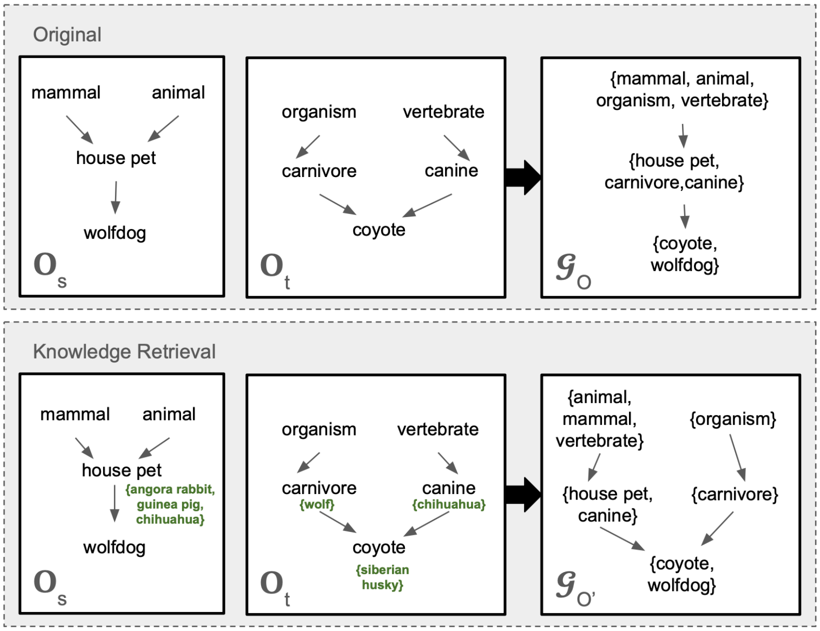


Fig. 1: Ontologies with ground sets, Ontology Matching and Concept Graphs

the concept set $C_s \cup C_t$, such that each partition is an equivalence class that contains a set of pairwise equivalent concepts in $C_s \cup C_t$.

Consistently, we define a *concept graph* $\mathcal{G}_O = (V_O, E_O)$ as a DAG with a set of nodes $V_O$, where each node $[c] \in V_O$ is an equivalence class in $\mathcal{C}$, and there exists an edge $([c], [c']) \in E_O$ if and only if there exists an edge $(c, c')$ in $E_s$ or in $E_t$. A concept graph $\mathcal{G}_O$ can be a multigraph: there may exist multiple edges of different relation names between two equivalent classes.

Given $O_s$ and $O_t$, the task of ontology matching is to properly formulate $R_{\simeq}$ and compute $\mathcal{C}$ induced by $R_{\simeq}$ over $O_s$ and $O_t$; or equivalently, compute the concept graph $\mathcal{G}_O$ induced by $R_{\simeq}$.

*Example 1.* Figure 1 depicts two ontologies $O_s$ and $O_t$ as DAGs, involving in total 9 concept nodes. An equivalence relation may suggest that "mammal", "animal", "organism" and "vertebrate" are pairwise similar; and similarly for the sets {"house pet", "carnivora" and "canine"}, and {"wolfdog, "coyote"}". This induces a concept graph $\mathcal{G}_o$ as shown on the top right as a result of ontology matching, with three equivalence classes.

**Language Models for Ontology Matching**. A language model $M$ takes as input a prompt query $q$, and generate an answer $q(M)$, typically an NL statement, for downstream processing. Large language models (LLMs) are foundation models that can effectively learn from a handful of in-context examples, included in a prompt query $q$, that demonstrate input–output distribution [61].

*Prompt query.* A prompt query $q$ is in a form of NL statements that specifies (1) a task definition $\mathcal{T}$ with input and output statement; (2) a set of in-context examples $\mathcal{D}$ with annotated data; (3) a statement of query context $Q$, which describe auxiliary query semantics; and optionally (4) specification on output format, and (5) a self-evaluation of answer quality such as confidence. An evaluation of a prompt query $q$ invokes an LLM $M$ to infer a query result $q(M)$.

We specify a prompt query $q$ and LLMs for ontology matching. A prompt query $q$ is in the form of $q(c, c')$, which asks "are $c$ and $c'$ semantically equivalent?" An LLM $M$ can be queried by $q(c, c')$ and acts as a Boolean "oracle" with "yes/no" answer. An LLM is *deterministic*, if it always generate a same answer for the same prompt query. We consider deterministic LLMs, as in practice, such LLMs are desired for consistent and robust performance.

# 3 Ontology Matching with LLMs

In this section, we provide a pragmatic characterization for the ontological equivalence relation $R_{\simeq}$. We then formulate the ontology matching problem.

## 3.1 Semantic Equivalence: A Characterization

**Concept similarity**. A variety of methods have been proposed to decide if two *concepts* are equivalent [7]. KROMA by default uses a Boolean function $F$ defined by a weighted combination of a semantic closeness metric sim[1] and the result from a set of LLMs $\mathcal{M}$ (see Section 4).

$$F(c, c') = \gamma \mathsf{sim}(c, c') + (1 - \gamma) q(c, c', \mathcal{M})$$

where $q$ is a prompt query that specifies the context of concept equivalence for LLMs, and $\gamma$ be a configurable parameter. KROMA supports a built-in library of semantic similarity functions sim, including (a) string similarity, feature and information measure [43], or normalized distances (NGDs) [30]; and (b) a variety of LLMs such as GPT-4o Mini (OpenAI) [39], LLaMA-3.3 (Meta AI) [34], and Qwen-2.5 (Qwen Team) [47].

**Ontological Bisimilarity**. We next provide a specification of the ontological equivalence relation, notably, *ontological bisimilarity*, denoted by the same symbol $R_{\simeq}$ for simplicity. Given a source ontology $O_s = (C_s, E_s)$, and a target ontology $O_t = (C_t, E_t)$, we say a pair of nodes $c_s \in C_s$ and $c_t \in C_t$ are *ontologically bisimilar*, denoted as $(c_s, c_t) \in R_{\simeq}$, if and only if there exists a non-empty binary relation $R_{\simeq}$, such that: (1) $c_s$ and $c_t$ are concept similar, *i.e.,* $F(c_s, c_t) \geq \alpha$, for a threshold $\alpha$; (2) for every edge $(c'_s, c_s) \in E_s$, there exists an edge $(c'_t, c_t) \in E_t$, such that $(c'_s, c'_t) \in R_{\simeq}$, and vice versa; and (3) for every edge $(c_s, c''_s) \in E_s$, there exists an edge $(c_t, c''_t) \in E_t$, such that $(c''_s, c''_t) \in R_{\simeq}$.

One can verify the following result.

**Lemma 1.** *The ontological bisimilar relation $R_{\simeq}$ is an equivalence relation.*

[1] We adopt similarity metrics that satisfy transitivity, *i.e.,* if concept c is similar to c', and c' is similar to c", then c is similar to c". This is to ensure transitivity of ontology equivalence, and is practical for representative concept similarity measures.

We can prove the above results by verifying that $R_{\simeq}$ is reflexive, symmetric and transitive over the concept set $C_s \cup C_t$, ensured by the transitivity of concept similarity and by definition. Observe that two nodes that are concept similar may *not* be ontologically bisimilar. On the other hand, two ontologically bisimilar entities must be concept similar, by definition.

### 3.2 Problem Statement

We now formulate our ontological matching problem. Given a *configuration* $W = (O_s, O_t, F, \mathcal{M}, \alpha)$ that specifies as input a source ontology $O_s$, a target ontology $O_t$, a Boolean function $F$ and threshold $\alpha \in (0, 1]$ that asserts concept similarity, and a set of LLMs $\mathcal{M}$, the problem is to compute a smallest concept graph $\mathcal{G}_O$ induced by the ontologically bisimilar equivalence $R_{\simeq}$.

We justify the above characterization by proving that there exists an "optimal", invariant solution encoded by a concept graph $\mathcal{G}_O$. To see this, we provide a *minimality* and *uniqueness* guarantee on $\mathcal{G}_O$.

**Lemma 2.** *Given a configuration $W$ and semantic equivalence specified by the ontological bisimilar relation $R_{\simeq}$, there is a unique smallest concept graph $\mathcal{G}_O$ that capture all semantically equivalent nodes in terms of $R_{\simeq}$.*

**Proof sketch:** We show that the above result holds by verifying the following. (1) There is a unique, maximum ontological bisimilar relation $R_{\simeq}$ for a given configuration $W = (O_s, O_t, F, \mathcal{M}, \alpha)$, where any LLM in $\mathcal{M}$ is a deterministic model for the same prompt query $q$ generated consistently from $W$. This readily follows from Lemma 1, which verifies that $R_{\simeq}$ is an equivalence relation. (2) Let the union of $O_s$ and $O_t$ be a graph $O_{st} = \{C_s \cup C_t, E_s \cup E_t\}$. By setting $\mathcal{G}_O$ as the quotient graph induced by the largest ontological bisimilar relation $R_{\simeq}$ over $O_{st}$, $\mathcal{G}_O$ contains the smallest number of equivalent classes (nodes) and edges. This can be verified by proof by contradiction. (3) The uniqueness of the solution can be verified by showing that $R_{\simeq}$ induces only one unique partition $\mathcal{C}$ and results in a concept graph $\mathcal{G}_O$ up to graph isomorphism. In other words, for any two possible concept graphs induced by $R_{\simeq}$, they are isomorphic to each other. □

The above analysis suggests that for a configuration $W$, it is desirable to compute such an optimal concept graph as an invariant, stable result with guarantees on sizes and uniqueness on topological structures. We next introduce KROMA and efficient algorithms to compute such optimal concept graphs.

## 4 KROMA Framework

### 4.1 Framework Overview

Given a *configuration* $W = (O_s, O_t, F, \mathcal{M})$ where $\mathcal{M}$ is a set of pre-trained LLMs $\mathcal{M}$, KROMA has the following major functional modules that enables a multi-session ontology matching process.

*Concept Graph Initialization* Upon receiving two ontologies $O_s = (C_s, E_s)$ and $O_t = (C_t, E_t)$, KROMA initializes the concept graph $\mathcal{G}_O = (V, E)$ with $V = C_S \cup C_T$ and $E = E_S \cup E_T$. For each concept $c \in V$, KROMA computes the *rank* $r(c)$ as described in Section 2.

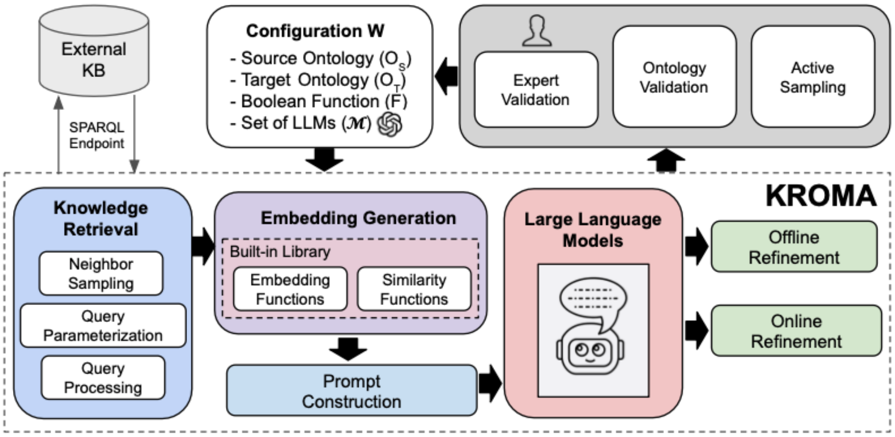


Fig. 2: KROMA Framework Overview: Major Components and Dataflow

*Knowledge Retrieval.* To assemble a rich, yet compact, context for each concept $c \in V$, we perform: (1) *Neighborhood Sampling*: traverse up to its two hops in $\mathcal{G}_O$ to collect parents, children, and "sibling" concepts of $c$, creating a node induced subgraph centered at $c$; (2) *subgraph parameterization*: Sample and substitute constant values from the subgraph with variables to create SPARQL queries $S_q$; and (3) *ground set curation*: process $S_q$ onto external knowledge bases to augment its ground set $c.\mathcal{I}$ with auxiliary information (*e.g.,* relevant entities, definition, labels, etc.).

*Embedding Generation.* In this phase, KROMA consults a built-in library of semantic similarity functions $\mathsf{sim}(\cdot,\cdot)$ and embedding functions $\mathsf{embd}(\cdot)$. For each concept $c \in V$, KROMA computes:

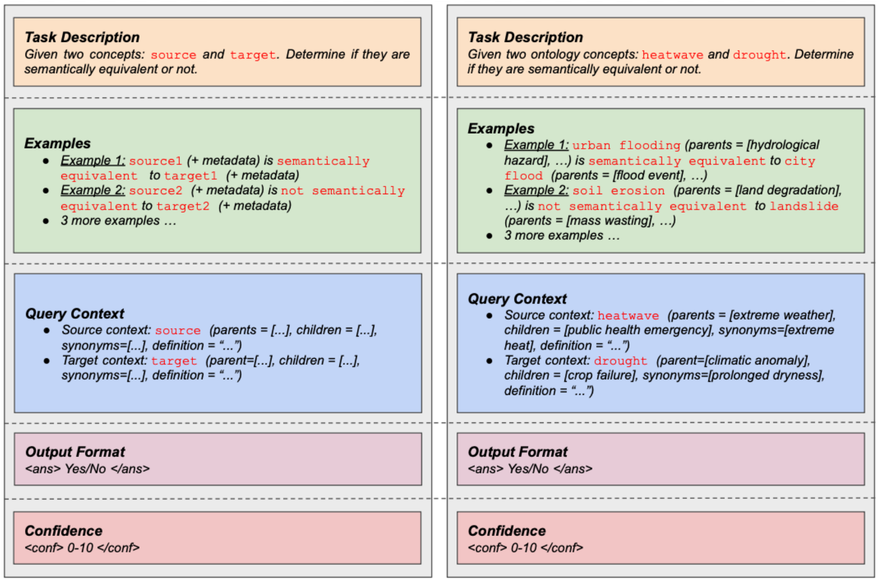
$$z_c = \alpha \ \mathsf{embd}_{graph}(c) + (1-\alpha) \ \mathsf{embd}_{text},$$

where $\mathsf{embd}_{graph}$ (e.g. node2vec [24]) captures the $c$'s topology information and $\mathsf{embd}_{text}$ (e.g. SciBERT [5]) encodes $c$'s textual context. After obtaining the necessary embeddings ($z_c \ \forall c \in V$), KROMA computes pairwise concept similarity between source and target ontologies using $\mathsf{sim}$ functions:

$$\mathbb{S} = \{(c_s, c_t, score_{s,t}) \mid c_s \in C_S, c_t \in C_T, score_{s,t} = \mathsf{sim}(\mathsf{z}_{\mathsf{c}_\mathsf{s}}, \mathsf{z}_{\mathsf{c}_\mathsf{t}})\}$$

From $\mathbb{S}$, we select the top-$k$ pairs with the highest similarity scores (i.e. in descending order of $score_{s,t}$), yielding the candidate list $\mathbb{C}$.

*Example 2.* We revisit Example 1. (1) A knowledge retrieval for node "house pet" samples its neighbors and issues a set of "star-shaped" SPARQL queries centered at "house pet" to query an underlying knowledge graph $KG$. This enriches its ground set with a majority of herbivore or omnivorous pets that are not "carnivore". Similarly, the ground set of "carnivore" is enriched by "wolf", unlikely a house pet. Despite "coyote" and "wolfdog" (house pet) alone are less similar, the ground set of "coyote" turns out to be a set of canine pets *e.g.,* "husky" that are



**Task Description**
*Given two concepts:* source *and* target*. Determine if they are semantically equivalent or not.*

**Examples**
- *Example 1:* source1 *(+ metadata) is* semantically equivalent *to* target1 *(+ metadata)*
- *Example 2:* source2 *(+ metadata) is* not semantically equivalent *to* target2 *(+ metadata)*
- *3 more examples …*

**Query Context**
- *Source context:* source *(parents = [...], children = [...], synonyms=[...], definition = "...")*
- *Target context:* target *(parent=[...], children = [...], synonyms=[...], definition = "...")*

**Output Format**
<ans> Yes/No </ans>

**Confidence**
<conf> 0-10 </conf>

**Task Description**
*Given two ontology concepts:* heatwave *and* drought*. Determine if they are semantically equivalent or not.*

**Examples**
- *Example 1:* urban flooding *(parents = [hydrological hazard], …) is* semantically equivalent *to* city flood *(parents = [flood event], …)*
- *Example 2:* soil erosion *(parents = [land degradation], …) is* not semantically equivalent *to* landslide *(parents = [mass wasting], …)*
- *3 more examples …*

**Query Context**
- *Source context:* heatwave *(parents = [extreme weather], children = [public health emergency], synonyms=[extreme heat], definition = "...")*
- *Target context:* drought *(parent=[climatic anomaly], children = [crop failure], synonyms=[prolonged dryness], definition = "...")*

**Output Format**
<ans> Yes/No </ans>

**Confidence**
<conf> 0-10 </conf>

Fig. 3: KROMA Prompt Query Template and Query Example

"coyote-like", hence similar with "wolfdog". (2) The embedding generation phase incorporates enriched ground sets and generate embeddings accordingly, which scores that distinguishes "house pet" from "carnivore" due to embedding difference, and assert "coyote" and "wolfdog" to be concept similar, hence a candidate pair to be "double checked" by LLMs in the next phase.

*Prompt Querying LLMs.* For each candidate pair $(c_s, c_t) \in \mathbb{C}$, KROMA automatically generates an NL prompt that includes: (1) Task description $\mathcal{T}$ (e.g. "*Given two ontology concepts and their contextual metadata, decide if they are related or not.*"), (2) In-context examples $\mathcal{D}$ containing both positive and negative matches, (3) Query context for $c_s$ and $c_t$ including their ground sets, (4) Output format and confidence (e.g., "Answer Yes or No, and provide a confidence score between 0 and 10."). It then calls (a set of) LLMs $\mathcal{M}$ to obtain a match decision with confidence. Low-confidence or conflicting outputs are routed to validation module. A query template and a generated example is illustrated in Figure 3.

*Ontology Refinement & Expert Validation.* KROMA next invokes a refinement process, OfflineRefine (Algorithm 1), to "cold-start" the construction of $\mathcal{G}_O$, or OnlineRefine (Algorithm 2), to incrementally refine $\mathcal{G}_O$ for any unseen, newly arrived concept nodes or edges. Any pair of nodes whose structural ranks remain in "conflict" is routed into a set of queries for expert validation (see Algorithm 2, Section 4); once approved, are integrated back into $\mathcal{G}_O$. An active sampling strategy is applied, to select pairs of nodes that have low confidence from LLMs for expert validation, to reduce the manual effort.

*Example 3.* Continuing with Example 2, as "golden retriever" and "coyote" are asserted by the function $F$ that combines the descision of semantic similarity

**Algorithm 1:** Offline Refinement

```
   Input: Source ontology $O_S = (C_S, E_S)$, target ontology $O_T = (C_T, E_T)$
   Output: Concept graph $\mathcal{G}_O$.
1  set $V \leftarrow C_S \cup C_T$; set $E \leftarrow E_S \cup E_T$;
2  Initialize concept graph $\mathcal{G}_O = (V, E)$;
3  foreach $c \in V$ do
4      compute rank $r(c)$ as in Section 2;
5  $\rho \leftarrow \max_{c \in V} r(c)$;
6  for $i \leftarrow 0$ to $\rho$ do
7      $B_i \leftarrow \{c : r(c) = i\}$;
8  $P \leftarrow \{B_0, \dots, B_\rho\}$;
9  for $i \leftarrow 0$ to $\rho$ do
10     $D_i \leftarrow \{X \in P : X \subseteq B_i\}$;
11     foreach $X \in D_i$ do
12         $G \leftarrow \mathsf{collapse}(G, X)$;
13     foreach $c \in B_i$ do
14         foreach $C \in P$ with $C \subseteq \bigcup_{j>i} B_j$ do
15             Split $C$ into $C_1, C_2$ by adjacency to $c$;
16             $P \leftarrow (P \setminus \{C\}) \cup \{C_1, C_2\}$;
17 return $\mathcal{G}_O$;
```

function sim and LLMs, an equivalence class is created in $\mathcal{G}'_O$. As "house pet" and "carnivore" has quite different embedding considering the features from themselves and their ground sets, "carnivore" is separated from the group of "house pet". This suggests further that "organism" now has a different context that distinguish it from the group {mammal, animal, vertebrate}, by the definition of bisimilarity equivalence. This leads to a finer-grained concept graph $\mathcal{G}'_O$.

## 5 Ontology Refinement

We next describe our ontology refinement algorithms. KROMA supports ontology refinement in two modes. The offline mode assumes that the source ontology $O_s$ and the target ontology $O_t$ are known, and performs a batch processing to compute the concept graph $\mathcal{G}_O$ from scratch. The online refinement incrementally maintains $\mathcal{G}_O$ upon a sequence of (unseen) triples (edges) from external resources.

The offline refinement algorithm is outlined as Algorithm 1. (1) It starts by initializing $\mathcal{G}_O$ (lines 1-6) as the union of $O_s$ and $O_t$, followed by computing the node ranks. At each rank, it initializes a "bucket" $B_i$ (as a single node set) that simply include all the concept nodes at rank $i$ (lines 7-8), and initialize a partition $\mathcal{P}$ with all the buckets (line 9). It then follows a "bottom-up" process to refine the buckets iteratively, by checking if two concepts $c$ and $c'$ in a same bucket $B_i$ are concept similar (as asserted by LLM and embedding similarity), and have all the neighbors that satisfy the requirement of ontological bisimilar relation by definition (lines 10-13). If not, a procedure collapse is invoked, to (1) split the bucket $B_i$ into three fragments: $B_i^1 = B_i \setminus \{c, c'\}$, $B_i^2 = \{c\}$, and $B_i^3 = \{c'\}$, followed by a "merge" check to test if $c$ and $c'$ can be merged to $B_i^1$; and (2) propagate this change to further "split-merge" operators to affected buckets at higher ranks (lines 14-17). This process continues until no change can be made.

*Correctness*. Algorithm 1 correctly terminates at obtaining a maximum bisimilar ontological equivalence relation $R_{\simeq}$, with two variants. (1) As ontologies are DAGs, it suffices to perform a one-pass, bottom-up splitting of equivalence classes following the topological ranks; (2) the collapse operator ensures the "mergable" cases to reduce unnecessary buckets whenever a new bucket is separated. This process simulates the correct computation of maximum bisimulation relation in Kripke structures (a DAG) [15], optimized for deriving ontology matching determined by LLM-based concept similarity and bisimilar equivalence.

*Time Cost.* The initialization of concept graph $\mathcal{G}_O$ is in $O(|O_s| + |O_t|)$. Here $|O_s|{=}|V_s|{+}|E_s|$; and $|O_t|$ is defined similarly. The iterative collapse (lines 10-17) takes in $O(|O_s| + |O_t|)$ time as the number of buckets (resp. edges) is at most $|C_s| + |C_T|$ (resp. $|E_s|{+}|E_t|$). The overall cost is in $O(|O_s| + |O_t|)$.

*Overall Cost*. We consider the cost of the entire workflow of KROMA. (1) The knowledge checking takes $O((|C_s| + |C_T|)|KG|)$ time, where $|KG|$ refers to the size of the external ontology or knowledge graph that is referred to by the knowledge retrieval via *e.g.,* SPARQL access. Note here we consider SPARQL queries with "star" patterns, hence the cost of query processing (to curate ground sets) is in quadratic time. (2) The total cost of LLM inference is in $O(|C_s||C_t|T_I)$, for a worst case that any pair of nodes in $C_s \times C_t$ are concept similar in terms of $\sim$. Here $T_I$ is the unit cost of processing a prompt query. Putting these together, the total cost is in $O(|C_s||C_t|T_I + (|C_s| + |C_t|)|KG| + (|O_s| + |O_t|))$ time.

**Online Refinement**. We next outline the online matching process. In this setting, KROMA receives new ontology components as an (infinite) sequence of triples (edges), and incrementally maintain a concept graph $\mathcal{G}_O$ by processing the sequence input in small batched updates $\Delta\mathcal{G}$. For each newly arrived concept (node) $c$ in $\Delta\mathcal{G}$, KROMA conducts knowledge retrieval to curate $c.\mathcal{I}$; and consult LLMs to decide if $c$ is concept similar to any node in $\mathcal{G}_O$. It then invokes online refinement algorithm to enforce the ontological bisimilar equivalence.

The algorithm (with pseudoscope reported in [1]) first updates the buckets in $\mathcal{G}_O$ by incorporating the nodes from $\Delta\mathcal{G}$ that are verified to be concept similar, as well as their ranks. It then incrementally update the buckets to maintain the bisimilar equivalence consistently via a "bottom-up" split-merge process as in Algorithm 1 (lines 6-15). Due to the unpredictability of the "unseen" concepts, the online refinement defers the processing of two "inconsistent" cases for experts' validation: (1) When a concept $c$ is determined to be concept similar by function $\sim$ but not LLMs with high confidence; or (2) whenever for a new edge $(c, c') \in \Delta\mathcal{G}$, $(c, [c_1]) \in R_{\simeq}$, $(c', [c_2]) \in R_{\simeq}$, yet $r(c_1) < r(c_2)$ in $\mathcal{G}_O$. Both require domain experts' feedback to resolve. These cases are cached into a query set $\mathcal{Q}$ to be further resolved in the validation phase (see Section 4). *We cache these cases into a query set by using an auxiliary data structure (*e.g.,* priority queue ranked by* LLM *confidence score) to manage their processing.*

*Analysis.* The correctness of online refinement carries over from its offline counterpart, and that it correctly incrementalize the split-merge operator for each newly arrived edges. For time cost, for each batch, it takes a delay time to update $\mathcal{G}_O$ in $O(|\mathcal{G}_O| + |\Delta G| \log |\Delta G| + |\Delta G| \log |V_O|)$ time. This result verifies that online refinement is able to response faster than offline maintenance that recomputes the concept graph from scratch. We present the detailed analysis in [1].

## 6 Experimental Study

We investigated the following research questions. **[RQ1]**: *How well* KROMA *improves state-of-the-art baselines with different* LLMs*?* **[RQ2]**: *How can knowledge retrieval and ontology refinement enhance matching performance?* and **[RQ3]**: What are the impact of alternative LLM reasoning strategies?

### 6.1 Experimental Setting

**Benchmark Datasets**. We selected five tracks from the OAEI campaign [45], covering various domain tracks. For each track, we selected two representative ontologies as a source ontology $O_s$ and a target ontology $O_t$. The selected tracks include Anatomy [16] (Mouse-Human), Bio-LLM [26] (NCIT-DOID), CommonKG (CKG) [20] (Nell-DBpedia, YAGO-Wikidata), BioDiv [32] (ENVO-SWEET), and MSE [28] (MI-MatOnto). To ensure a fair and comprehensive evaluation of KROMA, we adopted the standard benchmarks from the Ontology Alignment Evaluation Initiative (OAEI), enabling direct comparison with prior LLM-based methods. Despite the high cost of LLM inference, we tested KROMA across five diverse tracks to demonstrate its robustness across domains. As future work, we plan to scale up our experiments and examine the impact of workload size on performance.

**LLMs selection.** To underscore KROMA 's ability to achieve strong matching performance even with smaller or lower-performance LLMs, we selected models with relatively modest Chatbot Arena MMLU scores [9]: Gemma-2B (51.3%) and Llama-3.2-3B (63.4%), compared to the baseline systems Flan-T5-XXL (55.1%) and MPT- 7B (60.1%). We have selected a diverse set of LLMs, ranging from ultra-lightweight to large-scale—to demonstrate KROMA's compatibility with models that can be deployed on modest hardware without sacrificing matching quality. Our core evaluations use DeepSeek-R1-Distill-Qwen-1.5B [58] (1.5B), GPT-4o-mini [39] (8B), and Llama-3.3 [34] (70B), each chosen in a variant smaller than those employed by prior OM-LLM baselines. To further benchmark our framework, we include Gemma-2B [11] (2B), Llama-3.2-3B [3] (3B), Mistral-7B [4] (7B), and Llama-2-7B [2] (7B) in our ablation studies. To run inference on the aforementioned models, we make use of TogetherAI and OpenAI APIs as off-the-shelf inference services on hosted, pretrained models, and we are not performing any fine-tuning or weight updates.

LLMs *Determinism & Reproducibility.* To enforce determinism and ensure reproducibility, all LLM calls are issued with a fixed temperature of 0.3, striking a balance between exploration and exploitation; and a constant random seed is used for sampling.

*Confidence Calibration.* Our selection of LLMs is justified by a calibration test with their confidence over ground truth answers. The self-evaluated confidence by LLMs align well with performance: over 80% of correct matches fall in the top confidence bins (9–10), while fewer than 5% of errors are reported. Gemma-2B shows almost no errors above confidence 8, and both Llama-3.2-3B and Llama-3.3-70B maintain $\geq$ 95% precision at confidence thresholds of 9 or greater. We thus choose a confidence threshold to be 8.5 for all LLMs to accept their output.

**Test sets generation**. Following [25,23], for each dataset, we arbitrarily designate one concept as the "source" for sampling and its target counterpart. We remark that the source and target roles are interchangeable w.l.o.g given our theoretical analysis, algorithms and test results. We randomly sample 20 matched concept pairs from the ground truth mappings. For each source ontology concept, we select an additional 24 unmatched target ontology concepts based on their cosine similarity scores, thereby creating a total of 25 candidate mappings (including the ground truth mapping). Finally, we randomly choose 20 source concepts that lack a corresponding target concept in the ground truth and generate 25 candidate mappings for each. Each subset consists of 20 source ontology concepts with a match and 20 without matches, with each concept paired with 25 candidate mappings, totaling 1000 concept pairs per configuration.

Concepts and entities not selected as test sets are treated as external knowledge base for **knowledge retrieval**. All models operate with SciBERT [5] for **Embedding Generation**, leveraging its strength in scientific data embedding.

**Baselines.** Our evaluation considers **four** state-of-the-art LLM-based ontology matching methods: LLM4OM [22], MILA [56], OLaLa [27], and LLMap [25]. For RQ2, we also developed KROMA-NR, which skip knowledge retrieval, and KROMA-NB, which disables the bisimilarity-based clustering (hence clusters are determined by concept similarity alone).

*Naming Convention.* Each configuration uses a pattern `[Method][Optional Suffix][LLM Version]`, where the initials (K, M, O, L) specify the method (KROMA, MILA, OLaLa, LLM4OM). Suffixes "NKR" and "NR" denote "no knowledge retrieval" and "no ontology refinement", respectively, and the trailing version number (e.g., 3.3, 2.0, 4mini) specifies the underlying LLM release.

### 6.2 Experimental Results

**Exp-1: Effectiveness (RQ1)**. In this set of tests, we evaluate the performance of KROMA compared with baselines, and the impact of factors such as test sizes.

KROMA *vs. Baselines: Overall Performance.* Across all six datasets in Table 3 (abbreviated by their first capitalized letters), the full KROMA configuration (KL3.3) achieves the highest $F_1$ on every task, substantially outperforming competing baselines. For Mouse–Human, KL3.3 delivers 94.94 $F_1$, eclipsing MILA's best (ML3.1) at 92.20, OLaLa (OL2.0) at 90.20, and LLM4OM (L4G3.5) at 89.11. On NCIT–DOID, KL3.3 reaches 98.63 versus 94.80 for MILA, 83.01 for

| **Model Code** | **Full Description** |
|---|---|
| KL3.3 | Full KROMA using LLaMA-3.3 |
| KNR3.3 | KROMA without Knowledge Retrieval using LLaMA-3.3 |
| KNB3.3 | KROMA without Ontology Refinement using LLaMA-3.3 |
| KL3.1 | Full KROMA using LLaMA-3.1 |
| KL2.0 | Full KROMA using LLaMA-2-7B |
| KG2 | Full KROMA using Gemma-2B |
| KM7 | Full KROMA using Mistral-7B |
| K4mini | Full KROMA using GPT-4o-mini |
| KNR4mini | KROMA without Knowledge Retrieval using GPT-4o-mini |
| KNB4mini | KROMA without Ontology Refinement using GPT-4o-mini |
| KL3.2 | Full KROMA using LLaMA-3.2-3B |
| ML3.1 | MILA using LLaMA-3.1 |
| OL2.0 | OLaLa using LLaMA-2-7B |
| LLFT | LLM4OM using Flan-T5-XXL |
| L4G3.5 | LLM4OM using GPT-3.5 |
| L4L2 | LLM4OM using LLaMA-2-7B |
| L4M7 | LLM4OM using Mistral-7B |
| L4MPT | LLM4OM using MPT-7B |

Table 2: Model Codes and their full descriptions.

Table 3: KROMA Performance vs. Baselines.

| Model | MH | ND | NDB | YW | ES | MM |
|---|---|---|---|---|---|---|
| KL3.3 | 94.94 | 98.63 | 97.08 | 95.54 | – | 61.25 |
| KNR3.3 | 91.25 | 95.01 | 94.26 | 91.98 | – | 59.95 |
| KNB3.3 | 86.59 | 90.91 | 93.58 | 89.74 | – | 55.00 |
| KL3.1 | 94.50 | 98.24 | – | – | 91.43 | – |
| KL2.0 | 93.24 | – | 96.02 | – | 85.06 | – |
| KG2 | – | 85.53 | – | – | – | – |
| KM7 | – | – | – | – | 92.98 | – |
| ML3.1 | 92.20 | 94.80 | – | – | 83.70 | 32.97 |
| OL2.0 | 90.20 | – | 96.00 | – | 51.10 | – |
| L4G3.5 | 89.11 | 83.01 | 94.26 | – | – | – |
| K4mini | – | – | – | – | 93.18 | – |
| LLFT | – | 72.10 | – | – | – | – |
| L4L2 | – | – | – | 92.19 | – | – |
| L4M7 | – | – | – | – | 55.09 | – |
| L4MPT | – | – | – | – | – | 32.97 |

LLM4OM, and 72.10 for Flan-T5-XXL. Similar gaps appear on Nell–DBpedia (97.08 vs. 96.00/94.26), YAGO–Wikidata (95.54 vs. 92.19/93.33), ENVO–SWEET (93.18 vs. 92.98/85.06), and MI–MatOnto (61.25 vs. 59.05/32.97). These verify the effectiveness of KROMA over representative benchmark datasets. More details (*e.g.,* Precision and Recall) are reported in [1].

*Impact of different* LLMs. Across six ontology-matching tracks, KROMA equipped with Qwen2.5-1.5B outperforms the best existing baseline on five out of the six datasets (see Table 4). In the Anatomy's Mouse–Human track, Qwen2.5-1.5B achieves $F_1 = 83.58$ (versus 92.20 for the OM-LLM baseline), while GPT-

4o-mini reaches $F_1$ = 91.96. On NCIT–DOID both models surpass the baseline with $F_1$ of 97.44 and 97.56 (baseline: 94.80), and similar gains appear on Nell–DBpedia (95.02, 95.67 vs. 96.00), YAGO–Wikidata (95.45, 95.44 vs. 92.19), and MI–MatOnto (61.25, 60.88 vs. 32.97). Only on ENVO–SWEET does the smallest model dip below the baseline (79.80 vs. 83.70), whereas GPT-4o-mini (93.18) and Llama-3.3-70B (91.95) still lead. These results confirm that KROMA with knowledge retrieval and ontolgoy refinement can well exploit even relatively "smaller" LLMs with desirable performance, and further benefit from larger LLMs.

Table 4: KROMA Performance with different LLMs.

| Dataset | Qwen2.5 | GPT-4o-mini | Llama-3.3 |
|---|---|---|---|
| Mouse–Human | 83.6 | 92.0 | **94.9** |
| NCIT–DOID | 97.4 | 97.6 | **98.6** |
| Nell–DBpedia | 95.0 | 95.7 | **97.1** |
| YAGO–Wikidata | 95.5 | 95.4 | **95.5** |
| ENVO–SWEET | 79.8 | **93.2** | 92.0 |
| MI–MatOnto | 61.3 | 60.9 | **62.2** |

*Impact of Test sizes.* We next report the impact of ontology sizes to the performance of KROMA in performance (detailed results reported in [1]). For each dataset, we varied test sizes from 200 (xsmall) to 1,000 (full) pairs. KROMA's performance is in general insensitive to the change of test sizes. For example, its $F_1$ stays within a 1.90 variance on NCIT–DOID and a 1.10-point range on YAGO–Wikidata. This verifies the robustness of KROMA in maintaining desirable performance for large-scale ontology matching tasks.

**Exp-2: Ablation Analysis (RQ2).** In this test, we perform ablation analysis, comparing KROMA with its two variants, KROMA-NKR and KROMA-NR, removing knowledge retrieval and ontology refinement, respectively.

Table 5: KROMA Performance with/without Ontology Refinement (Left); and with/without Knowledge Retrieval (Right).

| Dataset | Model | **P** | **R** | $\mathbf{F_1}$ |
|---|---|---|---|---|
| Mouse–Human | KL3.3 | 90.78 | 99.50 | **94.94** |
| | KNR3.3 | 92.34 | 90.16 | 91.25 |
| NCIT-DOID | KL3.3 | 97.59 | 99.69 | **98.63** |
| | KNR3.3 | 93.20 | 96.90 | 95.01 |
| Nell-DBpedia | KL3.3 | 94.32 | 100.00 | **97.08** |
| | KNR3.3 | 97.46 | 91.27 | 94.26 |
| YAGO-Wikidata | KL3.3 | 91.80 | 99.60 | 95.54 |
| | KNR3.3 | 93.50 | 90.50 | 91.98 |
| ENVO-SWEET | K4mini | 87.23 | 100.00 | **93.18** |
| | KNR4mini | 86.90 | 98.00 | 92.12 |
| MI-MatOnto | KL3.3 | 45.74 | 92.69 | **61.25** |
| | KNR3.3 | 44.80 | 91.40 | 59.95 |

| Dataset | Model | **P** | **R** | $\mathbf{F_1}$ |
|---|---|---|---|---|
| Mouse–Human | KL3.3 | 90.78 | 99.50 | **94.94** |
| | KNKR3.3 | 100.00 | 76.36 | 86.59 |
| NCIT-DOID | KL3.3 | 97.59 | 99.69 | **98.63** |
| | KNKR3.3 | 83.33 | 100.00 | 90.91 |
| Nell-DBpedia | KL3.3 | 94.32 | 100.00 | **97.08** |
| | KNKR3.3 | 91.17 | 96.12 | 93.58 |
| YAGO-Wikidata | KL3.3 | 91.80 | 99.60 | 95.54 |
| | KNKR3.3 | 90.50 | 89.00 | 89.74 |
| ENVO-SWEET | K4mini | 87.23 | 100.00 | **93.18** |
| | KNKR4mini | 82.00 | 88.00 | 84.89 |
| MI-MatOnto | KL3.3 | 45.74 | 92.69 | **61.25** |
| | KNKR3.3 | 42.00 | 85.00 | 55.00 |

*Impact of Ontology Refinement.* Table 5 (Left) shows that by incorporating Ontology Refinement (KROMA vs. KROMA-NR), KROMA improves $F_1$ score across 6 datasets: Mouse–Human (+3.69), NCIT–DOID (+3.62), Nell–DBpedia (+2.82), YAGO–Wikidata (+3.56), ENVO–SWEET (+1.06), MI–MatOnto (+1.30). By pruning false candidate pairs via offline and online refinement strategies on the concept graph, ontology refinement retains true semantic connections, by filtering out noise while preserving true alignments.

*Impact of Knowledge Retrieval.* Table 5 (Right) shows that ablating Knowledge Retrieval (KROMA-KNR vs. KROMA) results in significant $F_1$ drop across all six benchmark datasets: Mouse–Human (–8.35), NCIT–DOID (–7.72), Nell–DBpedia (–3.50), YAGO–Wikidata (–5.80), ENVO–SWEET (–8.29), MI–MatOnto (–6.25). By enriching concepts with external, useful semantic context that are overlooked by other baselines, knowledge retrieval helps improving the performance.

We have also evaluated the efficiency of KROMA in terms of communication cost and performance gain from online processing. We found that KROMA is feasible for large-scale ontology matching, and the online refinement response fast to continuous matching tasks.

**Exp-3: Alternative LLM Reasoning Strategies**. We evaluate KROMA's performance with two alternative LLM reasoning strategies: "Debating" and "Deep reasoning" to understand their impact to ontology matching.

Table 6: KROMA Performance with/without "Debating" (Left); and with/without "Deep reasoning" (Right).

| Dataset | Model | P | R | $F_1$ |
|---|---|---|---|---|
| Mouse–Human | D2A3R | 100 | 70 | 82.35 |
| | D2A5R | 100 | 74 | **85.06** |
| | D4A3R | 100 | 66 | 79.52 |
| | D4A5R | 100 | 68 | 80.95 |
| Nell–DBpedia | D2A3R | 97.83 | 90 | 93.75 |
| | D2A5R | 95.91 | 94 | **94.95** |
| | D4A3R | 95.55 | 86 | 90.53 |
| | D4A5R | 93.88 | 92 | 92.93 |
| YAGO–Wikidata | D2A3R | 100 | 92 | 95.83 |
| | D2A5R | 100 | 96.9 | **98.41** |
| | D4A3R | 100 | 86 | 92.47 |
| | D4A5R | 100 | 90 | 94.73 |

| Dataset | Model | P | R | $F_1$ |
|---|---|---|---|---|
| NCIT–DOID | L3.3Short | 97.59 | 99.69 | **98.63** |
| | L3.3Long | 97.66 | 96.31 | 96.98 |
| Nell–DBpedia | L3.3Short | 94.32 | 100.00 | **97.08** |
| | L3.3Long | 94.67 | 95.48 | 95.07 |
| YAGO–Wikidata | L3.3Short | 91.80 | 99.60 | **95.54** |
| | L3.3Long | 92.31 | 94.80 | 93.53 |

*Can "Debating" help?* We implemented an LLM Debate ensemble [18], where multiple agents propose alignments with their chain-of-thought and then iteratively critique one another. To bound context size, we drop 50% of historic turns and keep only each agent's latest reply. Due to its expense, we tested this on Mouse-Human, Nell-DBpedia, and YAGO-Wikidata using four permutation of configurations: 2 or 4 agents over 3 or 5 debate rounds, denoted as `D[Number Of Agent]A[Number Of Round]R`. From Table 6 (Left), on Mouse–Human dataset, extending rounds in the 2-agent setup raised F1 from 82.35 to 85.06, whereas adding agents degraded performance (4 agents: 79.52 at 3 rounds, 80.95 at 5). Similar trends follow for Nell-DBpedia and YAGO-Wikidata. This interestingly indicates that "longer debates" help small ensembles converge to accurate matches, but larger groups introduce too much conflicting reasoning. In contrast, a single-agent, single-round KROMA pass attains F1 = 90.78, underscoring that a well-tuned solo model remains the most efficient and reliable choice.

*Can "Deep reasoning" help?* Building on DeepSeek-R1's "Aha Moment" [13], we extended KROMA to include a forced long chain-of-thought for self-revision. Noting that vanilla DeepSeek-R1 often emits empty "`<think>\n`" tags (i.e. "`<think>\n\n</think>`"), we altered our prompt so every response must start with "`<think>\n`", ensuring the model spells out its reasoning. We then compared standard Llama-3.3 (70B) ("short" reasoning, noted as L3.3Short) to DeepSeek-R1-Distill-Llama-3.3 (70B) ("long" reasoning, noted as L3.3Long), finding that the added `"<think>\n"` tag inflated inputs by 20% and outputs by 600% but delivered only tiny precision gains (e.g., NCIT–DOID: 97.59% → 97.66%) while slashing recall (99.69% → 96.31%), dropping F1 by 1.65–2.01 points, based on Table 6 (Right). This suggests that verbose self-reflection may improve transparency but consumes crucial context and can overly filter valid matches—especially in binary tasks, where lengthy chains of thought have been shown to harm recall, which has been consistently observed in [62].

## 7 Conclusion

We have presented KROMA, an ontology matching framework that exploits the semantic capabilities of LLMs within a bisimilar-based ontology refinement process. We show that KROMA computes a provably unique minimized structure that captures semantic equivalence relations, with efficient algorithms that can significantly reduces LLMs communication overhead, while achieving state-of-the-art performance across multiple OAEI benchmarks. Our evaluation has verified that LLMs, when guided by context and optimized prompting, can rival or surpass much larger models in performance. A future topic is to extend KROMA with more LLM reasoning strategies and ontology engineering tasks.

**Supplemental Material Statement**. Our code and experimental data has been made available at `https://anonymous.4open.science/r/kroma/`, including an extended version of the paper [1].

## References

1. Full version (2025), https://anonymous.4open.science/r/kroma/full.pdf
2. AI, M.: Llama 2 7b chat model card. https://huggingface.co/meta-llama/Llama-2-7b-chat (2023)
3. AI, M.: Llama 3.2 3b instruct model card. https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct (2024)
4. AI, M.: Announcing mistral 7b. https://mistral.ai/news/announcing-mistral-7b (2023)
5. Beltagy, I., Lo, K., Cohan, A.: Scibert: A pretrained language model for scientific text. In: EMNLP-IJCNLP (2019)
6. Borgo, S., Ferrario, R., Gangemi, A., Guarino, N., Masolo, C., Porello, D., Sanfilippo, E.M., Vieu, L.: Dolce: A descriptive ontology for linguistic and cognitive engineering. Applied Ontology **17**(1), 45–69 (Mar 2022)
7. Chandrasekaran, D., Mago, V.: Evolution of semantic similarity—a survey. Acm Computing Surveys (Csur) **54**(2), 1–37 (2021)
8. Chen, A., Song, Y., Zhu, W., Chen, K., Yang, M., Zhao, T., zhang, M.: Evaluating o1-like llms: Unlocking reasoning for translation through comprehensive analysis (2025), https://arxiv.org/abs/2502.11544
9. Chiang, W.L., Zheng, L., Sheng, Y., Angelopoulos, A.N., Li, T., Li, D., Zhang, H., Zhu, B., Jordan, M., Gonzalez, J.E., Stoica, I.: Chatbot arena: An open platform for evaluating llms by human preference (2024)
10. Cruz, I.F., Sunna, W.: Structural Alignment Methods with Applications to Geospatial Ontologies. Transactions in GIS **12**(6), 683–711 (2008)
11. DeepMind, G.: Gemma 2b model card. https://huggingface.co/google/gemma-2-2b (2024)
12. DeepSeek-AI: Deepseek-v3 technical report (2023)
13. DeepSeek-AI: Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning (2025)
14. Degtyarenko, K., Hastings, J., Matos, P., Ennis, M.: Chebi: An open bioinformatics and cheminformatics resource. Current Protocols in Bioinformatics **26**(1) (Jun 2009)
15. Dovier, A., Piazza, C., Policriti, A.: A fast bisimulation algorithm. In: Berry, G., Comon, H., Finkel, A. (eds.) Computer Aided Verification (CAV 2001). Lecture Notes in Computer Science, vol. 2102 (2001)
16. Dragisic, Z., Ivanova, V., Li, H., Lambrix, P.: Experiences from the anatomy track in the ontology alignment evaluation initiative. In: Journal of Biomedical Semantics (2017)
17. Drobnjakovic, M., Ameri, F., Will, C., Smith, B., Jones, A.: The Industrial Ontologies Foundry (IOF) Core Ontology
18. Du, Y., Li, S., Torralba, A., Tenenbaum, J.B., Mordatch, I.: Improving factuality and reasoning in language models through multiagent debate (2023), https://arxiv.org/abs/2305.14325
19. Euzenat, J., Shvaiko, P.: Ontology matching, 2nd edn (2013)
20. Fallatah, O., Zhang, Z., Hopfgartner, F.: A gold standard dataset for large knowledge graphs matching. In: ISWC 2020 (2020)
21. Ferré, S.: Expressive and scalable query-based faceted search over SPARQL endpoints. In: Proceedings of the 10th International Conference on Semantic Systems (SEMANTiCS) (2014)

22. Giglou, H.B., D'Souza, J., Engel, F., Auer, S.: Llms4om: Matching ontologies with large language models (2024), https://arxiv.org/abs/2404.10317
23. Giglou, H.B., D'Souza, J., Engel, F., Auer, S.: Llms4om: Matching ontologies with large language models. In: ESWC (2024)
24. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks (2016)
25. He, Y., Chen, J., Dong, H., Horrocks, I.: Exploring large language models for ontology alignment (2023), https://arxiv.org/abs/2309.07172
26. He, Y., Chen, J., Dong, H., Horrocks, I.: Exploring large language models for ontology alignment. In: ISWC 2023 Posters and Demos: 22nd International Semantic Web Conference (2023)
27. Hertling, S., Paulheim, H.: Olala: Ontology matching with large language models. In: Proceedings of the 12th Knowledge Capture Conference 2023. p. 131–139 (Dec 2023). https://doi.org/10.1145/3587259.3627571, http://dx.doi.org/10.1145/3587259.3627571
28. Huschka, M., Nasr, E.: Mse benchmark (2023), https://tinyurl.com/MSE-Benchmark
29. Jensen, M., Colle, G.D., Kindya, S., More, C., Cox, A.P., Beverley, J.: The common core ontologies (arXiv:2404.17758) (Aug 2024)
30. Jiang, Y., Wang, X., Zheng, H.T.: A semantic similarity measure based on information distance for ontology alignment. Information Sciences **278**, 76–87 (2014)
31. Jiménez-Ruiz, E., Cuenca Grau, B., Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E.: Logmap: Logic-based and scalable ontology matching. In: ISWC (2011)
32. Karam, N., Khiat, A., Algergawy, A., Sattler, M., Weiland, C., Schmidt, M.: Matching biodiversity and ecology ontologies: challenges and evaluation results. The Knowledge Engineering Review **35**, e9 (2020)
33. Li, X.: A survey on llm test-time compute via search: Tasks, llm profiling, search algorithms, and relevant frameworks (2025), https://arxiv.org/abs/2501.10069
34. Llama Team, A..M.: The llama 3 herd of models (2024)
35. Llama Team, A.: Llama 2: Open foundation and fine-tuned chat models (2023)
36. Mistral-AI: Mistral large 2 (2023)
37. Norouzi, S.S., Mahdavinejad, M.S., Hitzler, P.: Conversational ontology alignment with chatgpt (2023), https://arxiv.org/abs/2308.09217
38. OpenAI: Language models are few-shot learners. In: NeurIPS 2020 (2020)
39. OpenAI: Gpt-4 technical report (2024)
40. OpenAI: Openai o1 system card (2024), https://arxiv.org/abs/2412.16720
41. Otte, J.N., Beverley, J., Ruttenberg, A.: Bfo: Basic formal ontology. Applied Ontology **17**(1), 17–43 (Jan 2022)
42. Peeters, R., Bizer, C.: Using chatgpt for entity matching (2023), https://arxiv.org/abs/2305.03423
43. Pirró, G., Euzenat, J.: A feature and information theoretic framework for semantic similarity and relatedness. In: ISWC (2010)
44. Portisch, J., Hladik, M., Paulheim, H.: Background knowledge in ontology matching: A survey (2024)
45. Qiang, Z., Taylor, K., Wang, W., Jiang, J.: Oaei-llm: A benchmark dataset for understanding large language model hallucinations in ontology matching. arXiv preprint arXiv:2409.14038 (2024)

46. Qin, L., Chen, Q., Zhou, Y., Chen, Z., Li, Y., Liao, L., Li, M., Che, W., Yu, P.S.: Multilingual large language model: A survey of resources, taxonomy and frontiers (2024), https://arxiv.org/abs/2404.04925
47. Qwen, Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Dang, K., Lu, K., Bao, K., Yang, K., Yu, L., Li, M., Xue, M., Zhang, P., Zhu, Q., Men, R., Lin, R., Li, T., Tang, T., Xia, T., Ren, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Wan, Y., Liu, Y., Cui, Z., Zhang, Z., Qiu, Z.: Qwen2.5 technical report (2025), https://arxiv.org/abs/2412.15115
48. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding by generative pre-training (2018)
49. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners (2019)
50. Rahm, E., Bernstein, P.: A survey of approaches to automatic schema matching (2001)
51. Rajamohan, B.P., Bradley, A.C.H., Tran, V.D., Gordon, J.E., Caldwell, H.W., Mehdi, R., Ponon, G., Tran, Q.D., Dernek, O., Kaltenbaugh, J., Pierce, B.G., Wieser, R., Yue, W., Lin, K., Kambo, J., Lopez, C., Nihar, A., Savage, D.J., Brown, D.W., Sharma, H., Giera, B., Tripathi, P.K., Wu, Y., Li, M., Davis, K.O., Bruckman, L.S., Barcelos, E.I., French, R.H.: Materials data science ontology(mds-onto): Unifying domain knowledge in materials and applied data science. Scientific Data **12**(1), 628 (Apr 2025)
52. Shvaiko, P., Euzenat, J.: A survey of schema-based matching approaches (2005)
53. Shvaiko, P., Euzenat, J.: Ontology matching: State of the art and future challenges. IEEE Transactions on Knowledge and Data Engineering **25**(1), 158–176 (2013)
54. Sun, J., Zheng, C., Xie, E., Liu, Z., Chu, R., Qiu, J., Xu, J., Ding, M., Li, H., Geng, M., Wu, Y., Wang, W., Chen, J., Yin, Z., Ren, X., Fu, J., He, J., Yuan, W., Liu, Q., Liu, X., Li, Y., Dong, H., Cheng, Y., Zhang, M., Heng, P.A., Dai, J., Luo, P., Wang, J., Wen, J.R., Qiu, X., Guo, Y., Xiong, H., Liu, Q., Li, Z.: A survey of reasoning with foundation models (2024), https://arxiv.org/abs/2312.11562
55. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: NeurIPS 2014 (2014)
56. Taboada, M., Martinez, D., Arideh, M., Mosquera, R.: Ontology matching with large language models and prioritized depth-first search (2025), https://arxiv.org/abs/2501.11441
57. Team, K.: Kimi k1.5: Scaling reinforcement learning with llms (2025), https://arxiv.org/abs/2501.12599
58. Team, Q.: Introducing qwen1.5 (2024), https://qwenlm.github.io/blog/qwen1.5/
59. Trojahn, C., Vieira, R., Schmidt, D., Pease, A., Guizzardi, G.: Foundational ontologies meet ontology matching: A survey. Semantic Web **13**(4), 685–704 (2022)
60. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: (NeurIPS (2017)
61. Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E.H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., Fedus, W.: Emergent abilities of large language models (2022), https://arxiv.org/abs/2206.07682
62. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., Zhou, D.: Chain-of-thought prompting elicits reasoning in large language models (2023), https://arxiv.org/abs/2201.11903

63. Xu, Y., Hu, L., Zhao, J., Qiu, Z., Xu, K., Ye, Y., Gu, H.: A survey on multilingual large language models: corpora, alignment, and bias. Frontiers of Computer Science **19**(11) (2025)
64. Zhou, Z., Ning, X., Hong, K., Fu, T., Xu, J., Li, S., Lou, Y., Wang, L., Yuan, Z., Li, X., Yan, S., Dai, G., Zhang, X.P., Dong, Y., Wang, Y.: A survey on efficient inference for large language models (2024)

# Appendix

## A Proofs of Theorems and Lemmas

**Proof of Lemma 1**. The relation $R_{\simeq}$ is an equivalence relation.

*Proof.* Let the set $C = C_s \cup C_t$. It suffices to show that the relation $R_{\simeq}$ is reflexive, symmetric, and transitive for a concept equivalence function $F$ that is transitive, *i.e.,* for any three concepts $c_1, c_2, c_3$ in $C$, if $F(c_1, c_2) =$ true, $F(c_2, c_3)$ $=$ true, then $F(c_1, c_3)$ is true.

(1) For any node $c \in C$, $(c, c)$ is a trivial case as a concept is the same (hence bisimilar) to itself.

(2) We next show that $R_{\simeq}$ is symmetric, *i.e.,* for any node pair $(c_1, c_2) \in R_{\simeq}$, $(c_2, c_1) \in R_{\simeq}$. To see this, assume that $(c_2, c_1) \notin R_{\simeq}$. Then there exists a neighbor $c_2'$ of $c_2$ such that no neighbor of $c_1$ can be found to match any neighbor of $c_2'$. This violates the condition that for every edge $(c_2', c_2) \in E_s$, one can find a corresponding edge with the "matched" end nodes in $R_{\simeq}$, indicating that $R_{\simeq}$ is not an ontological bisimilarity function. This leads to the contradiction. Hence $(c_1, c_2) \in R_{\simeq}$ indicates $(c_2, c_1) \in R_{\simeq}$, *i.e.,* $R_{\simeq}$ is symmetric.

(3) We next show that if $(c, c') \in R_{\simeq}$ and $(c', c'') \in R_{\simeq}$, then $(c, c'') \in R_{\simeq}$. Assume $(c, c'') \notin R_{\simeq}$. Then either there exists a neighbor $c_u$ of $c$, such that no equivalent "match" can be found in the neighbor of $c''$, or vice versa. We consider w.l.o.g the first case. As $(c', c'') \in R_{\simeq}$, then $(c'', c') \in R_{\simeq}$ given the symmetric property. Then for any neighbor $c_u''$ of $c''$, one can find an equivalent match $c_u'$ of $c'$, such that $(c_u'', c_u') \in R_{\simeq}$. As $(c, c') \in R_{\simeq}$, then $(c', c) \in R_{\simeq}$. Hence for the neighbor $c_u'$, there must exist a neighbor $c_u$ of $c$, such that $(c_u', c_u) \in R_{\simeq}$. Given the transitivity of the concept similarity determined by $F$ function, $c_u''$ and $c_u'$ are concept similar, and $c_u'$ and $c_u$ are concept similar; hence $c_u''$ and $c_u$ are concept similar. This suggests $(c_u'', c_u) \in R_{\simeq}$. Hence $(c_u, c_u'') \in R_{\simeq}$ given the symmetric property. This contradicts to the assumption that $c_u$ has no equivalent counterpart in $R_{\simeq}$. Hence the transitivity of $R_{\simeq}$ holds.

Putting these together, $R_{\simeq}$ is an equivalence relation.

**Proof of Lemma 2**. Given a configuration $W$ and semantic equivalence specified by the ontological bisimilar relation $R_{\simeq}$, there is a unique smallest concept graph $\mathcal{G}_O$ that capture all semantically equivalent nodes in terms of $R_{\simeq}$.

*Proof.* To see this, we let $R^*$ to be the largest ontological bisimilar relation that holds on $C = C_s \cup C_t$. That is, there is no other relation $R$ such that $R* \subset R$ and $R$ is an ontological bisimilar relation. Let a graph $\mathcal{G}^*$ be the quotient graph of the ontology graphs $O_s \cup O_t$ induced by the concept set $C$. It suffices to show that $\mathcal{G}^*$ is the smallest concept graph over $C$, and is unique up to all smallest concept graphs one may construct in terms of ontological equivalence relations.

(1) We first show that $\mathcal{G}^*$ is the smallest concept graph. To see this, assume there exists another concept graph $\mathcal{G}'$ that contains fewer nodes or edges than $\mathcal{G}^*$, induced by another ontological equivalence relation $R'$. Assume w.l.o.g that

it contains fewer nodes. Then there must exist two concept nodes $c$ and $c'$ in $C$, such that $c$ and $c'$ are in the same equivalence class (node) in $\mathcal{G}'$ but not in $\mathcal{G}^*$. In other words, $(c, c') \in R'$ but not in $R^*$. This suggests that $|R^*| \leq |R'|$, violating the fact that $\mathcal{G}^*$ is induced by a largest ontological equivalence relation $R^*$. The above analysis also holds for the case that $G^*$ contains more edges than $G'$. Hence, $\mathcal{G}^*$ contains the least number of nodes and edges, and is the smallest concept graph.

(2) We next show the uniqueness of $\mathcal{G}^*$. Let there be another smallest concept graph $\mathcal{G}'$ induced by another ontological equivalence relation $R'$. Moreover, $\mathcal{G}^*$ and $G'$ are different in terms of graph isomorphism. Then there are three cases. (i) $\mathcal{G}^*$ and $\mathcal{G}'$ have different number of nodes or edges. For either case, an analysis similar to (1) leads to a contradiction that $R^*$ is the maximum ontological equivalence relation. (ii) $|\mathcal{G}'| = |\mathcal{G}^*|$, and there exists a node $c \in C$ such that its equivalence classes $[c]'$ and $[c]^*$ are different in $\mathcal{G}'$ and $\mathcal{G}^*$. Then there must exist a pair of nodes $(c, c')$, such that $(c, c') \in [c]'$ and $(c, c') \notin [c]^*$. This case reduces to the analysis in (1), leading to a contradiction that $R^*$ is the largest ontological equivalence relation. Hence $\mathcal{G}^*$ is structurally unique up to graph isomorphism, *i.e.,* if there exists a second smallest concept graph $\mathcal{G}'$, then $\mathcal{G}^*$ and $\mathcal{G}'$ are identical in terms of graph isomorphism.

# B Algorithm Analysis

**Algorithm 1: Running example**. We describe Algorithm 1 with a running example to clarify the dynamic clustering process.

*Example 4.* Revisiting Example 3, an initial concept graph $\mathcal{G}$ is created with three blocks having ranks from 0 to 2, following an ascending order of their ranks: $B_0 = \{$ wolfdog, coyote $\}$, which contains two semanticaly similar concepts; similarly, $B_1 = \{$ house pet, canine, carnivore $\}$, and $B_2 = \{$ mammal, animal, vertebrate, organism$\}$. This provides a node partition $P$ of the concept set $V_s \cup V_t$ as $\{B_0, B_1, B_2\}$, involving all the 9 concepts. The offline refinement algorithm starts with $B_0$. As $\mathcal{G}$ is a DAG, $B_0$ contains two semantically similar concepts, neither has a child. Hence $B_0 = D_0$, and collapse does not produce new blocks within $B_0$. Indeed, the knowledge retrieval phase of "coyote" issues a star query that retrieves a match of "coyote" as "siberian husky", which indicates that the concept "coyote" does not necessarily refer to the class of wild coyotes, but actually refers to a context of "coyote-like" canine that have similar look and can be tamed as pets.

It then processes block $B_1$. Following example 3, as the knowledge retrieval phase enriched the context of concepts between house pet and carnivore, with example matches such as "angora rabbit" that are strictly not "carnivore", $B_1$ "collapses" to two blocks, $B_1^1$, that contains $\{$ house pet, canine $\}$, and a new block $B_1^2$ $\{$carnivore$\}$. Note that "house pet" and "canine" are still grouped in the same block, due to their matches "chihuahua" in common, ensuring instance-level similarity to some extend (as reflected in the concatenated features, and confirmation of positive semantic relevance from LLMs). The algorithm then

**Algorithm 2:** Online Ontology Refinement

```
   Input: a concept graph $\mathcal{G}_O$=$(V_O, E_O)$; a sequence of triples $\Delta\mathcal{G}$;
   Output: Updated concept graph $\mathcal{G}_O \oplus \Delta\mathcal{G}$
1  set $\mathcal{Q} \leftarrow \emptyset$;
2  update $r(c)$ for all $c \in V_O$ w.r.t. new edges in $\Delta\mathcal{G}$;
   // Process new insertion in ascending order of rank
3  Sort $\Delta\mathcal{G}$ with ascending ranks ;
4  foreach $e = (u \rightarrow u') \in \Delta\mathcal{G}$ do
       // Maintain bisimilar equivalence classes $[\cdot]_{R_e}$
5      Split$(u,\ u',\ [u]_{R_e},\ [u']_{R_e})$;
6      if $r\big([u]_{R_e}\big) > r\big([u']_{R_e}\big)$ then
7          foreach $v \in \mathcal{B}\big([u]_{R_e}\big)$ do
8              Merge$(\{u\}, v)$;
9          foreach $v' \in \mathcal{B}\big([u']_{R_e}\big)$ do
10             Merge$(\{u'\}, v')$;
11     else if $r\big([u]_{R_e}\big) = r\big([u']_{R_e}\big)$ then
12         foreach $v \in \mathcal{P}\big([u']_{R_e}\big)$ do
13             Merge$(\{u\}, v)$;
14         foreach $v' \in \mathcal{C}\big([u]_{R_e}\big)$ do
15             Merge$(\{u'\}, v')$;
16     else
           // for inconsistent ranks, defer to further
              validation
17         Q $\leftarrow$ Q $\cup \{(u, u')\}$;
18 return $\mathcal{G}_O$ with $\mathcal{Q}$ to be resolved;
```

split the rank 2 blocks to two. For each block, it further verifies if such split propagates to partition of $B_1$. As house pet links to "wolfdog", "canine" links to "coyote", both are still semantically similar, such collapse does not affect $B_0$.

The process continues to rank 2 blocks, for which the block $B_2$ is split into two blocks, $\{animal, mammal, vertebrate\}$ and $\{organism\}$, respectively, due to the semantic difference between $B_1^1$ and $B_2^1$. As no new changes can be made at rank 3, the refinement outputs $\mathcal{G}'_O$ as the result.

*Time Cost Analysis*. The initialization of concept graph $\mathcal{G}_O$ is in $O(|O_s| + |O_t|)$. Here $|O_s|$=$|V_s| + |E_s|$; and $|O_t|$ is defined similarly. The iterative collapse (lines 10-17) takes in $O(|O_s| + |O_t|)$ time as the number of buckets (resp. edges) is at most $|C_s| + |C_T|$ (resp. $|E_s|$+$|E_t|$). The overall cost is in $O(|O_s| + |O_t|)$.

### Algorithm 2: Analysis.

*Correctness*. It suffices to show that Algorithm 2 correctly maintains the ontological equivalence in terms of bisimilar relation upon the arrival of a single triple. For this, we perform a case analysis on the triple $e = (u, u') \in \Delta\mathcal{G}$. First, the procedure Split correctly decides if the concepts $u$ and $u'$ already included in any of the block of the current concept graph $\mathcal{G}_O$, and verifies if the knowledge retrieval, co-determined by LLMs and semantic similarity indicators, Introduce changes to their semantic similarity. If so, it performs the necessary split operators to separate $u$ or $u'$ out of their original block to be a new, singleton equivalence class (new blocks). We provide a case-by-case analysis below.

(1) Case 1: $u \in B_i$, $u' \in B_j$, and $i > j$. In this case, the "Merge" operators (Procedure Merge) correctly maintain the equivalence classes at rank $i$ and $j$ by merging $[u]$ (resp. $[u']$) with their equivalence classes at rank $i$ (resp. $j$), by checking if any entity is semantically similar to $u$ (resp. $u'$). Note that the merge operators have a strong data locality: this does not incur new split operations at higher-level blocks. Indeed, the "Upper-level" collapse of blocks can only be triggered by lower-level partition of blocks, given the definition of ontological bisimilar equivalence. Moreover, such a merge does not propagate to new merges at higher ranks.

(2) Case 2: $u \in B_i$, $u' \in B_j$, and $i = j$. Recall that KROMA performs a semantic similarity test before the refinement process. As the edge $e = (u, u')$ arrives, $u$ and $u'$ should have already be asserted to be not similar as co-determined by LLMs and knowledge retrieval. Hence the equivalence class $[u]$ can only be merged to the "parent" level blocks of $[u']$ (denoted as $\mathcal{P}([u'])$), if any bisimilar equivalent blocks exists; similarly, the block $[u']$ can only be merged to the "child" level blocks of $[u]$, if any. These two cases are correctly coped with by consistently invoking procedure Merge.

(3) Case 3: $u \in B_i$, $u' \in B_j$, and $i < j$; or $u$ and $u'$ are not semantically similar in any concept in the current $\mathcal{G}_O$ (hence $i$ and $j$ are unknown). For the first case, adding edges $e$ has a potential risk of forming a cycle in the DAG-structured concept graph. For the second case, the uncertainty remains and requires further validation from human experts, as current LLMs and knowledge retrieval-based verification remains insufficient to decide even the semantic similarity. Hence, the edge is properly added to a set of queries $Q$, for inconsistency checking. The correctness of $\mathcal{G}_O$ remains, as the further investigation (a) "reduces" $i < j$ case to be either Case 1 or 2, (b) rejects the pair due to inconsistency or semantic irrelevance, or (c) initializes a new connected concept graph that contains $[u]$ and $[u']$ as a single edge.

Putting the above analysis together, Algorithm 2 correctly processes each of these three cases to maintain the concept graph.

*Time Cost Analysis.* The Online Ontology Refinement procedure runs in near-linear time in the size of the compressed graph and the update batch. Recomputing all node ranks on $\mathcal{G}_r$ costs $O(N + M)$ (with $N = |V_r|$, $M = |E_r|$), and sorting the $\Delta$ new edges by rank takes $O(\Delta \log \Delta)$. Each insertion is then processed via union-find splits and merges in $O\big((M + \Delta)\,\alpha(N)\big)$ overall—since every edge participates in at most one split or merge and each such operation is $O(\alpha(N))$, where $\alpha$ is the inverse Ackermann function. Finally, dispatching $Q$ deferred pairs to the human verifier incurs an extra $O(Q)$. Altogether, the time complexity is

$$O\big(N + M + \Delta \log \Delta + (M + \Delta)\,\alpha(N) + Q\big)\,,$$

with space overhead $O(N + M + \varDelta + Q)$.

## C Additional Experiments and Discussion

**KROMA vs Baselines.** Table 7 shows that the full KROMA configuration (KL3.3) consistently achieves the highest $F_1$-score on all six benchmarks. For Mouse–Human, KL3.3 reaches 94.94%, outperforming the next best model (KL3.1) by 0.44 points. On NCIT–DOID, KL3.3 posts 98.63%, edging out KL3.1 (98.24%) and far surpassing other baselines. In Nell–DBpedia, YAGO–Wikidata, and ENVO–SWEET, KL3.3 attains 97.08%, 95.54%, and 93.18% $F_1$, respectively—each more than 1.5 points above any alternative. Even on the challenging MI–MatOnto track, KL3.3's 61.25% $F_1$ clearly beats KL3.2 (59.05%) and other methods. These results confirm that, across diverse domains, KROMA 's retrieval-augmented, refinement-driven pipeline with Llama-3.3 yields the best precision–recall balance compared to existing LLM-based and traditional baselines. On missing entries in the evaluation table, it is because that many baselines and their exact test-set pairings were not available to be called or not published, so direct reproduction or comparison were not possible. Retraining the LLMs would be prohibitively expensive. We will track the availability of these resources to update Table 7.

**Confidence Calibration.** Figure 4 illustrates how well each model's self-reported confidence aligns with its actual performance across six diverse ontology-matching datasets (Mouse–Human, NCI Thesaurus–DOID, NELL–DBpedia, YAGO–Wikidata, ENVO–SWEET, and MI–MAtONTO). In each subplot, the horizontal axis is divided into eleven "confidence bins" (e.g., 1–2, 2–3, . . . , 10–11), and the vertical axis separates correct versus incorrect predictions. The heatmap's color intensity corresponds to the number of predictions falling into each cell (correct/wrong × confidence bin).

A clear pattern emerges across all six datasets: the vast majority of correct matches are concentrated in the highest confidence bins—particularly 9–10 and 10–11—while almost no wrong answers appear at those high confidence levels. Conversely, whenever an error does occur, it is overwhelmingly reported with lower self-evaluated confidence (bins $< 8$). For example, in the Mouse–Human subplot, more than 80 percent of correct alignments lie in the 9–10 and 10–11 bins (bright yellow and light green shading), whereas the "Wrong" row is nearly empty in those same bins. Similarly, the NCI Thesaurus–DOID and NELL–DBpedia plots show that over 80 percent of all correct alignments land in the top two confidence buckets, with fewer than 5 percent of errors ever reported above a confidence of 8.

Taken together, these calibration curves give us strong evidence that a confidence cutoff around 8.5 will safely distinguish nearly all true positives from false positives. By accepting only those matches where the model's own confidence exceeds 8.5, we preserve the vast majority of correct alignments (since 80–90 percent of them sit in bins 9–10 and 10–11) and reject almost all of the few remaining errors. Consequently, for our downstream ontology-matching pipeline,

Table 7: KROMA vs Baselines.

| **Dataset** | **Model** | $P$ | $R$ | $F_1$ |
|---|---|---|---|---|
| Mouse-Human | KL3.3 | 90.78 | 99.50 | **94.94** |
| | KNR3.3 | 92.34 | 90.16 | 91.25 |
| | KNB3.3 | 100.00 | 76.36 | 86.59 |
| | KL3.1 | 90.42 | 99.00 | 94.50 |
| | KL2.0 | 100.00 | 87.34 | 93.24 |
| | ML3.1 | 100.00 | 87.50 | 92.20 |
| | OL2.0 | 91.40 | 89.10 | 90.20 |
| | L4G3.5 | 90.82 | 87.46 | 89.11 |
| NCIT-DOID | KL3.3 | 97.59 | 99.69 | **98.63** |
| | KNR3.3 | 93.20 | 96.90 | 95.01 |
| | KNB3.3 | 83.33 | 100.00 | 90.91 |
| | KL3.1 | 97.43 | 99.08 | 98.24 |
| | KG2 | 84.76 | 86.32 | 85.53 |
| | ML3.1 | 96.40 | 93.20 | 94.80 |
| | LLFT | 86.10 | 62.00 | 72.10 |
| | L4G3.5 | 86.19 | 80.06 | 83.01 |
| Nell-DBpedia | KL3.3 | 94.32 | 100.00 | **97.08** |
| | KNR3.3 | 97.46 | 91.27 | 94.26 |
| | KNB3.3 | 91.17 | 96.12 | 93.58 |
| | KL2.0 | 95.50 | 96.55 | 96.02 |
| | OL2.0 | 100.00 | 92.20 | 96.00 |
| | L4G3.5 | 100.00 | 89.14 | 94.26 |
| YAGO-Wikidata | KL3.3 | 91.80 | 99.60 | 95.54 |
| | KNR3.3 | 93.50 | 90.50 | 91.98 |
| | KNB3.3 | 90.50 | 89.00 | 89.74 |
| | KL3.2 | 89.09 | 98.00 | 93.33 |
| | L4L2 | 100.00 | 85.52 | 92.19 |
| ENVO-SWEET | K4mini | 87.23 | 100.00 | **93.18** |
| | KNR4mini | 86.90 | 98.00 | 92.12 |
| | KNB4mini | 82.00 | 88.00 | 84.89 |
| | KL3.1 | 87.11 | 96.19 | 91.43 |
| | KL2.0 | 92.50 | 78.72 | 85.06 |
| | KM7 | 87.19 | 99.59 | 92.98 |
| | ML3.1 | 80.30 | 59.50 | 83.70 |
| | OL2.0 | 43.10 | 61.30 | 51.10 |
| | L4M7 | 59.00 | 51.67 | 55.09 |
| MI-MatOnto | KL3.3 | 45.74 | 92.69 | **61.25** |
| | KNR3.3 | 44.80 | 91.40 | 59.95 |
| | KNB3.3 | 42.00 | 85.00 | 55.00 |
| | KL3.2 | 56.36 | 62.00 | 59.05 |
| | L4MPT | 89.70 | 20.19 | 32.97 |

we adopt a uniform confidence threshold of 8.5 for every LLM, striking a balance between recall (capturing most correct alignments) and precision (avoiding high-confidence mistakes).

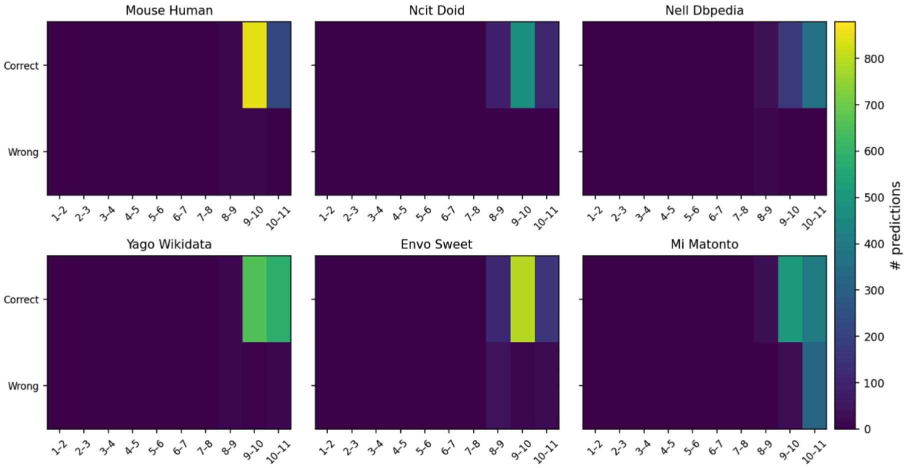


Fig. 4: LLMs Confidence Calibration

KROMA **Performance on different ontology sizes.** Table 8 demonstrates that KROMA 's precision, recall, and $F_1$-scores remain remarkably stable as we move from very small (xsmall) to full-sized ontologies. For NCIT–DOID, $F_1$ fluctuates only between 97.50% (xsmall) and 98.63% (full), with recall slightly increasing on larger splits and precision peaking at 98.00% on the large subset. Similarly, on NELL–DBpedia, $F_1$ rises steadily from 95.83% (xsmall) to 97.58% (large) before a minor dip to 97.08% at full scale, reflecting high recall (>99%) even as more candidate classes are introduced. YAGO–Wikidata shows the same trend: recall climbs from 98.00% to 99.60% as the ontology grows, while precision decreases only marginally (92.00% to 91.80%), resulting in consistent $F_1$ values in the mid-90s. These results indicate that KROMA 's retrieval-augmented pipeline can accommodate ontologies of increasing size without sacrificing matching quality, suggesting strong scalability for real-world, large-scale entity-matching tasks.

KROMA 's robustness to growing ontology size stems from its two-stage retrieval-augmented design, which keeps the LLM 's decision space both focused and consistent as more classes are added. First, instead of asking the model to compare every source–target pair globally, KROMA uses a semantic-search index to pull in only the top-k most relevant candidates for each source concept; as the ontology grows, irrelevant classes are filtered out early, so the LLM is still evaluating a bounded, high-precision shortlist. Second, once a small set of likely matches is identified, KROMA applies a fixed-length prompting and refinement step—complete with a confidence-based acceptance threshold—so that the LLM sees a similar amount of contextual information regardless of the overall ontology's full size. In other words, expanding the classifier's universe does not force the model to process vastly more text at inference time; it simply broadens the retrieval pool, but the same narrow, top-k candidates are forwarded to the matching prompt. Finally, because the confidence filter rejects low-certainty pairs consistently across all scales, noisy or spurious candidates introduced by a larger ontology rarely make it through to final alignment.

Table 8: KROMA Performance on different ontology sizes.

| Dataset | Metric | xsmall | small | medium | large | full |
|---|---|---|---|---|---|---|
| NCIT-DOID | $P$ | 97.50 | 97.88 | 97.92 | 98.00 | 97.59 |
| | $R$ | 97.50 | 98.12 | 97.50 | 98.13 | 99.69 |
| | $F_1$ | 97.50 | 98.00 | 97.71 | 98.06 | 98.63 |
| Nell-DBpedia | $P$ | 95.00 | 95.19 | 95.24 | 95.63 | 94.32 |
| | $R$ | 96.67 | 98.33 | 99.17 | 99.58 | 100.00 |
| | $F_1$ | 95.83 | 96.74 | 97.18 | 97.58 | 97.08 |
| YAGO-Wikidata | $P$ | 92.00 | 92.50 | 92.67 | 92.75 | 91.80 |
| | $R$ | 98.00 | 98.75 | 99.17 | 99.38 | 99.60 |
| | $F_1$ | 94.90 | 95.58 | 95.89 | 96.04 | 95.54 |

KROMA **Efficiency on Communication Cost and Offine/Online Refinement Delay speedup.** Across all six benchmark datasets, KROMA cuts the average communication cost to roughly a quarter of what a naïve full-pairwise matching approach would require: Qwen2.5, GPT-4o-mini, and Llama-3.3 each achieve overall token-exchange rates of about 25.9%, 25.6%, and 26.4%, respectively. On smaller ontologies like Mouse–Human and ENVO–SWEET, the cost dips below 7%, while even the largest schemas (NCIT–DOID and MI–MatOnto) stay in the 40–54% range. This reduction arises because KROMA 's retrieval stage pre-filters each source concept to a compact shortlist of candidates, so the LLM only receives a fraction of the total class descriptions instead of the entire ontology. As a result, fewer tokens are sent in both the query prompts and the LLM 's responses, directly lowering per-dataset communication without sacrificing matching quality.

When comparing end-to-end latency, performing KROMA 's refinement offline yields a consistent speedup of over 9× on average (with individual dataset factors ranging from 2.33× on NCIT–DOID to 18.76× on YAGO–Wikidata). In the offline scenario, all heavy LLM calls—such as deep refinement of candidate pairs—are run ahead of time, leaving only lightweight lookups and confidence checks to occur at inference. Online, that means the service can respond almost instantly with precomputed results instead of re-invoking the model for every request. Because each source term has already been matched and filtered offline, the system avoids redundant LLM invocations, dramatically cutting delay and delivering real-time throughput in applications.

KROMA **Token Usage.** Token footprint of KROMA remains modest even after refinement: Deep reasoning DR1D-Llama-3 (70B) issues roughly 620K total tokens (130K outputs) across all calls, versus 480K (15K outputs) for Llama-3.3 (70B), translating to an average of 605 tokens per API call compared to 470 tokens (Table 10). This ≈ 29% increase incurs only a mild cost for the substantial gains in alignment quality.

Table 9: KROMA Efficiency on Communication Cost (Left); and Offline vs. Online Refinement Delay speedup (Right).

| Dataset | Qwen2.5 | GPT-4o-mini | Llama-3.3 |
|---|---|---|---|
| Mouse–Human | 3.06% | 2.88% | 6.76% |
| NCIT-DOID | 54.48% | 54.63% | 54.48% |
| Nell-DBpedia | 26.09% | 25.76% | 26.09% |
| YAGO-Wikidata | 21.86% | 21.93% | 22.01% |
| ENVO-SWEET | 6.99% | 6.82% | 6.99% |
| MI-MatOnto | 43.04% | 41.66% | 41.81% |
| **Overall** | **25.92%** | **25.61%** | **26.36%** |

| Dataset | Online ÷ Offline |
|---|---|
| Mouse–Human | 7.17× |
| NCIT-DOID | 2.33× |
| Nell-DBpedia | 5.63× |
| YAGO-Wikidata | 18.76× |
| ENVO-SWEET | 7.00× |
| MI-MatOnto | 13.31× |
| **Overall** | **9.03×** |

| Model | Total tokens | Average tokens per call |
|---|---|---|
| Llama-3.3-70B | 480 | 470 |
| DR1D-Llama-3-70B | 620 | 605 |

Table 10: KROMA Token Usage.

KROMA limits the number of tokens sent to API-based LLMs, ensuring it remains highly scalable. Data-level optimization, including input compression and output organization techniques, would become increasingly necessary to enhance efficiency in the foreseeable future [64].

KROMA **Runtime.** We evaluated KROMA on the ENVO–SWEET ontology, which comprises 40 nodes and 1,000 edges. For each source concept, one target candidate pair is selected (*e.g.,* a single "pair" evaluation), yielding 1,000 total candidate pairs. Because our matching pipeline's workload—both for the LLM and the refinement stages—depends directly on the ontology's size (nodes + edges) and the number of candidate pairs, these statistics serve as the basis for extrapolating runtimes to other datasets.

*LLM Invocation Time.* The end-to-end inference step with a single-agent LLM required 10,916 seconds on ENVO–SWEET. Following the survey on efficient inference for LLMs [64], inference cost scales approximately linearly with the total token count of the input graph encoding [(nodes + edges) → tokens]. Thus, we report the average token size, which is approximately 470 tokens per API call.

*Refinement Time.* Offline refinement took 68 seconds and online refinement 14 seconds on the same 1,000 candidate pairs. This breaks down to:

$$\frac{68\text{ s}}{1\,000} = 0.068\text{ seconds/pair (offline)}, \qquad \frac{14\text{ s}}{1\,000} = 0.014\text{ seconds/pair (online)}.$$

For a dataset with $P$ candidate pairs, you would therefore expect roughly $0.068P$ seconds of offline processing and $0.014P$ seconds for online refinement, scaling directly wih the number of source-target candidates.

**Can "Active Sampling" help?** In semi-supervised experiments on the Bio-LLM NCIT-DOID benchmark, KROMA paired with Llama-3.3-70B matches or exceeds the performance of MILA + Llama-3.1-70B. Under an active-learning regime with few-shot demonstrations, our method scores 1.90 F1 points higher than MILA and surpasses the previous state of the art by over 6 points. As Table 11 shows, combining bisimilarity-guided refinement with selective querying delivers a substantial boost in alignment accuracy.

Table 11: Active Learning on NCIT-DOID.

| **Dataset** | **Model** | $P$ | $R$ | $F_1$ |
|---|---|---|---|---|
| NCIT-DOID | KROMA + Llama-3.1 (70B) | 97.85 | 98.15 | **98.00** |
| | MILA + Llama-3.1 (70B) | 96.70 | 92.80 | 94.67 |

**SPARQL Queries.** KROMA issues parameterized SPARQL templates to fetch auxiliary data. The“star-shaped SPARQL" declares a query pattern in which the concept serves as the central node, with its neighbors around it in a star-like topology as “facet search” [21]. For example, when querying the DBpedia endpoint, the following templates are used to extract a concept’s parents, children, synonyms, and labels, each limited to at most 10 results:

*Query Parents Template.*

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?object WHERE {
 dbo:<code> rdf:type ?object
}
LIMIT 10
```

*Query Children Template.*

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?object WHERE{
   ?object rdf:type dbo:<code> .
}
LIMIT 10
```

*Query Synonyms Template.*

```
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
```

```
SELECT DISTINCT ?object ?synonym
WHERE {
  ?object rdf:type dbo:<code> .
  ?object skos:altLabel ?synonym .
  FILTER (LANG(?synonym) = "en")
}
LIMIT 10
```

*Query Labels Template.*

```
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>

SELECT DISTINCT ?altLabel
WHERE {
  dbo:<code> skos:altLabel ?altLabel .
  FILTER (LANG(?altLabel) = "en")
}
LIMIT 10
```