Wikontic: Constructing Wikidata-Aligned, Ontology-Aware Knowledge Graphs with Large Language Models

Alla Chepurova[1,2]   **Aydar Bulatov**[1,2]   **Mikhail Burtsev**[3]   **Yuri Kuratov**[1,2]

[1]Cognitive AI Systems Lab, Moscow, Russia
[2]Moscow Independent Research Institute of Artificial Intelligence, Moscow, Russia
[3]London Institute for Mathematical Sciences, London, UK
{chepurova,bulatov,kuratov}@cogailab.com, mb@lims.ac.uk

Abstract

Knowledge graphs (KGs) provide structured, verifiable grounding for large language models (LLMs), but current LLM-based systems commonly use KGs as auxiliary structures for text retrieval, leaving their intrinsic quality underexplored. In this work, we propose *Wikontic*, a multi-stage pipeline that constructs KGs from open-domain texts by extracting candidate triplets with qualifiers, enforcing Wikidata-based type and relation constraints, and normalizing entities to reduce duplication. The resulting KGs are compact, ontology-consistent, and well-connected; on MuSiQue, the correct answer entity appears in 96% of generated triplets. On HotpotQA, our triplets-only setup achieves 76.0 F1, and on MuSiQue 59.8 F1, matching or surpassing several retrieval-augmented generation baselines that still require textual context. In addition, Wikontic attains state-of-the-art information-retention performance on the MINE-1 benchmark (86%), outperforming prior KG construction methods. Wikontic is also efficient at build time: KG construction uses less than 1,000 output tokens, about 3× fewer than AriGraph and <1/20 of GraphRAG. The proposed pipeline improves the quality of the generated KG and offers a scalable solution for leveraging structured knowledge in LLMs. Wikontic is available at https://github.com/screemix/Wikontic.

1 Introduction

A substantial amount of the world's knowledge exists solely in unstructured textual form, such as news, scientific articles, blogs, and posts on social networks. While large language models (LLMs) are capable of extracting insights from these data, their internal representations are latent and often unverifiable, making them susceptible to hallucinations. In contrast, knowledge graphs (KGs) record information as explicit subject-relation-object triplets, supporting verifiable queries, incremental updates, and multi-step reasoning (e.g., in answering compositional questions), making KGs a reliable complement to both LLMs and retrieval-augmented generation (RAG) systems. Therefore, creating high-quality KGs directly from raw text provides reliable, transparent knowledge that complements LLMs and RAG systems.

Extracting structured knowledge from text is a long-standing challenge in Information Retrieval. A common formulation is closed information extraction (cIE), which assumes predefined fixed sets of entities and relation predicates drawn from an existing KG and seeks to recover all triplets that conform to that schema. Classical cIE pipelines decompose the task into stages such as named entity recognition and relation classification (Zeng et al., 2014; Zhang et al., 2020). However, separating these stages leads to error accumulation and prevents the sharing of information between tasks. More recent end-to-end approaches approach extraction as a sequence-to-sequence problem, training models to directly generate triplets from text (Distiawan et al., 2019; Cabot and Navigli, 2021; Josifoski et al., 2022) or complete missing links in KG (Yao et al., 2019). While this reduces error propagation, such models remain difficult to adapt to new domains, requiring costly retraining on high-quality annotated corpora that remain scarce. LLMs offer a promising alternative: their broad knowledge and strong prompting abilities enable open-domain extraction without expensive task-specific training (Wang et al., 2020; Josifoski et al., 2023; Chepurova et al., 2024).

In contrast to cIE, open information extraction (oIE) does not impose predefined entity and relation names, ontology constraints, and constructs KGs from scratch. This flexibility makes oIE an attractive tool for augmenting LLMs and RAG systems, with recent work demonstrating reduced inference costs and more reliable retrieval (Chen et al., 2024; Gutierrez et al., 2024; Guo et al., 2024; Li et al., 2024; Han et al., 2024; Gutiérrez et al.,

2025). For example, AriGraph (Anokhin et al., 2025) learns KGs to create long-term semantic memory, while Distill-SynthKG (Choubey et al., 2024) integrates mixed text–triplet structures to improve question answering. However, most current oIE pipelines rely on KGs mainly as auxiliary scaffolds to structure text retrieval, rather than treating the KG itself as a high-quality knowledge resource. This perspective overlooks the potential of compact and non-redundant KGs to represent information directly, and leaves their quality and reasoning capabilities underexploited. As a result, the practical use of oIE KGs remains limited. Extracted triplets often contain heterogeneous surface forms—for example, "NYC located in USA" versus "New York City in-country United States"—fragmenting the KG into redundant or inconsistent representations. Synonymy, coreference, and predicate variation accumulate with scale, eroding the very strengths that motivate KG construction in the first place: precision, interpretability, and logical consistency.

To address this, we combine the flexibility of oIE with the structural rigor of cIE by leveraging external ontologies. Wikidata (Vrandečić, 2012), one of the largest community-maintained knowledge bases, offers rich entity classes, relation schemas, and domain-range constraints across more than 100M entities. Its breadth allows coverage from common sense to specialized domains, while its formal constraints provide principled supervision for validating LLM outputs. Yet, integrating such ontology guidance into a fully automated pipeline poses key challenges: (i) extracting candidate triplets without predefined labels, (ii) typing and disambiguating entities under ontology classes despite lexical ambiguity, and (iii) refining nodes and edges iteratively while preserving alignment.

In this paper, we address these challenges with **Wikontic**, a multi-stage framework that constructs Wikidata-aligned, ontology-aware KGs directly from texts using LLMs. Unlike prior works that apply Wikidata ontology only for evaluation or entity linking (Polat et al., 2025), we *integrate a large-scale ontology from Wikidata directly into the information extraction pipeline*. Wikontic includes six components: (i) a curated ontology database derived from Wikidata, (ii) candidate triplet extraction with qualifiers, (iii) ontology-aware triplet refinement enforcing schema constraints, (iv) subject/object name refinement for entity deduplication, (v) KG storage, and (vi) retrieval for multi-hop question answering. Starting from unstructured text, Wikontic extracts triplets with LLMs, types, and validates them against Wikidata, and deduplicates entities to yield compact, consistent KGs.

The resulting KGs are both interpretable and effective. When used as the sole knowledge source for multi-hop QA, Wikontic achieves competitive performance with RAG and KG-based methods that rely on raw text as context (Lee et al., 2024; Li et al., 2024; Anokhin et al., 2025; Panda et al., 2024; Gutiérrez et al., 2025). Moreover, our graphs exhibit superior coverage of salient information and strong internal connectivity between relevant nodes. In summary, our main **contributions** are:

1. We introduce *Wikontic*, which (1) extracts candidate triplets, (2) enforces schema and ontology constraints on entity types and domain–range, and (3) performs alias-aware entity normalization and deduplication to reduce redundancy, yielding ontology-consistent KGs.

2. We show that Wikontic's KGs are ontology-consistent, have low redundancy, strong coverage, and connectivity; on MuSiQue, the correct answer entity is present in 96% of generated triplets.

3. Using a KG as the sole knowledge source (no access to the original text) on multi-hop question answering, Wikontic attains 76.0 F1 on HotpotQA and 59.8 F1 on MuSiQue, matching or surpassing several RAG/KG baselines that still rely on text.

4. Wikontic achieves state-of-the-art results on the MINE-1 benchmark, reaching 86% information-retention score.

5. Wikontic's KG construction uses less than 1,000 output tokens, which is $\sim 3\times$ fewer than AriGraph and $< 1/20$ of GraphRAG.

2 Methods: Wikontic

Wikontic is a multi-stage pipeline for constructing high-quality, ontology-aware KGs directly from unstructured text (Figure 1). Unlike prior approaches that directly map text to graph form and often yield noisy, redundant, or inconsistent outputs, our pipeline explicitly integrates LLMs with Wikidata-derived ontological constraints, entity normalization, and iterative refinement.

To enable triplet validation and alignment, the pipeline stores ontology rules and the current KG (Section 2.1). The triplet extraction pipeline (Section 2.2) consists of three main stages: (i) triplet candidate extraction with contextual metadata, (ii) ontology-aware triplet refinement, and (iii) entity normalization and deduplication. These stages aim
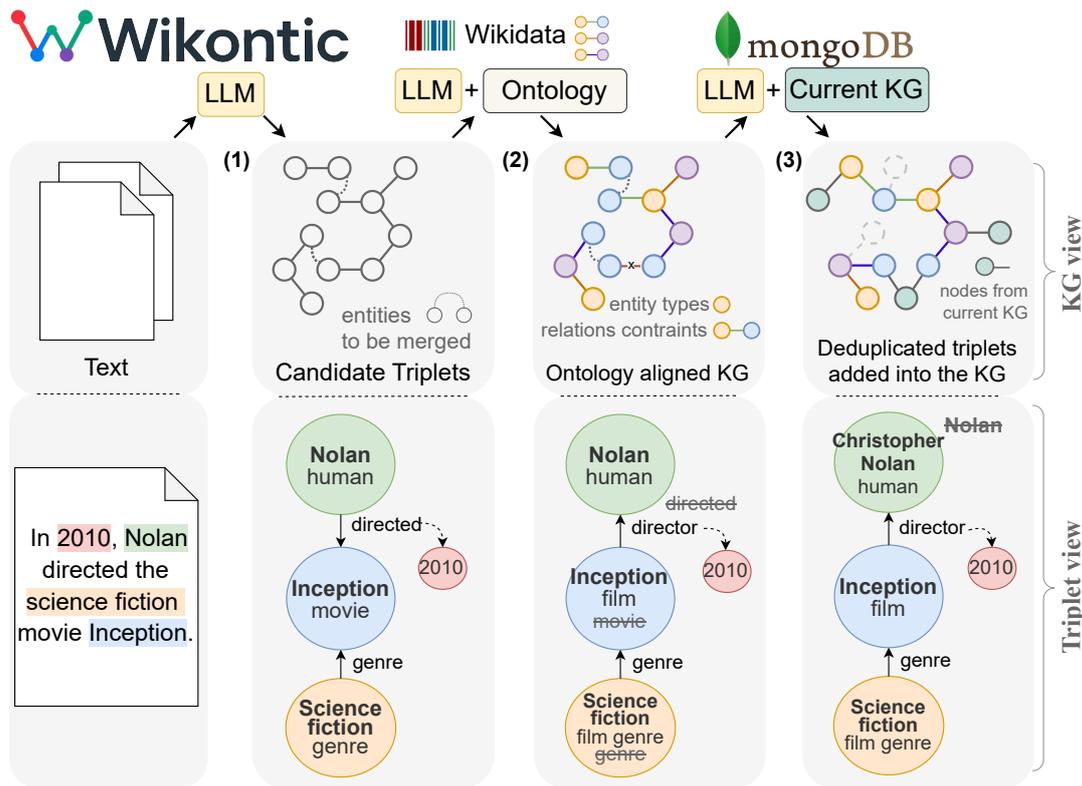
Figure 1: Overview of *Wikontic*: an ontology-guided pipeline that constructs a Wikidata-aligned KG from text. (1) An LLM extracts candidate (subject, relation, object) triplets (gray). (2) The extracted triplets are then refined using Wikidata's ontology: entity types are assigned (colored nodes), and relations that violate ontology constraints are corrected or removed. (3) Finally, entity names are normalized, and duplicated surface forms are merged. The resulting graph is de-duplicated, ontology-consistent, and ready for downstream tasks.

to enforce structural validity and reduce redundancy, to produce a cleaner and more semantically coherent KG that can replace raw text in RAG for multi-hop QA tasks (Section 2.3).

2.1 Ontology and KG Databases

We built a custom **ontology schema database** derived from Wikidata. The schema database includes properties (i.e., relations) and their compatible entity types. Properties required solely for linking external data (e.g., multimedia or external identifiers) were excluded, leaving 2,464 factual properties with suitable datatypes (e.g., WikibaseItem, Quantity, Point in time).

For each property, we retrieved subject and object type constraints from Wikidata (e.g., Q21503250 for subject, Q21510865 for object). These constraints define type compatibility rules, specifying which entity classes a relation can logically connect and thereby guiding ontology-consistent triplet construction.

To support constraint generalization, we recursively expanded entity types using 'instance of' (P31) and 'subclass of' (P279) relations, building full taxonomies from each type up to the root. Such a hierarchy is essential because relation constraints are defined at different levels of abstraction. For example, a relation may allow connections between instances of the broader class 'audiovisual work', even if the entity is typed more specifically as 'film'. By propagating the allowed properties of each parent type downwards to its children entity type, we ensure that entities can still be matched to valid relations whenever the constraint applies to its parent class.

We collected labels and aliases for all entity types and relations. Dense retrieval indexes for relation and entity type names, as well as their respective aliases, support semantic search. These indexes allow us to semantically align relation and entity type names from extracted triplets with Wikidata definitions, even when surface forms differ.

The KG database stores triplets, canonical entity names, and aliases. A dense retrieval index over aliases supports efficient linking and deduplication. As extraction proceeds, new entities are

inserted with canonical labels and aliases, keeping the KG compact yet incrementally extensible.

Dense retrieval indexes used in both databases were built with Contriever embeddings (Izacard et al., 2022) and Atlas MongoDB vector search[1]. MongoDB's hybrid support for structured queries and dense retrieval enables both efficient graph and semantic search.

2.2 Ontology-aware Triplet Extraction

Stage 1: Candidate Triplet Extraction. We extract factual triplet candidates from unstructured text with an LLM, capturing subject-relation-object triplets, along with contextual qualifiers that enhance the semantic meaning of the triplet. LLM is prompted with instruction and in-context examples to extract triplets that include entity types for both the subject and object, as well as additional metadata that mirrors the structure of Wikidata qualifiers. These qualifiers are essential because they capture contextual information such as time, location, or conditions. While such details usually cannot be expressed as standalone facts, they are critical for preserving factual precision and avoiding loss of accurate knowledge during knowledge extraction.

For instance, given the text *"In 2010, Christopher Nolan directed the science fiction movie Inception"*, the extracted triplet would be: (Nolan, directed, Inception) with entities types (human, film) and the qualifier: {point in time: 2010}. Further details and examples are provided in Appendix A.4.

However, LLM outputs may be semantically redundant or structurally inconsistent. The entity and relation names may not align with existing entities and relations already present in the KG. For example, an LLM might extract "Nolan" as the subject from one input text and extract "Christopher Nolan" from the other one; or relations like "directed" vs. "director" might appear in different grammatical forms or inverse directions in different input texts (see Figure 1, bottom). Without additional correction, these inconsistencies lead to entity duplication and increased KG size, which degrades both storage efficiency and downstream reasoning. Thus, to improve consistency and reduce redundancy, the next steps of the pipeline validate extracted triplets using Wikidata's ontology and normalize entity names.

Stage 2: Ontology-aware Refinement. At this stage, each candidate triplet is refined using the schema and constraints of the Wikidata ontology:

Entity typing: For both subject and object, we retrieve the top-10 candidate types from the dense retrieval index. The LLM then selects the most plausible type. We then add supertypes from the taxonomy to ensure coverage when constraints are defined at higher abstraction levels.

Relation validation: Using Wikidata constraints, we identify all relations that can legally connect selected subject and object types, including inverse combinations of subject and object types (e.g., directed vs. director). These candidate relations are ranked by cosine similarity to the originally extracted relation.

Triplet backbone reconstruction: The text, triplet, and valid relations are passed to the LLM, which selects the most plausible ontology-valid configuration, yielding a refined triplet backbone.

This stage enforces structural validity, semantic alignment, and consistency with Wikidata's ontology. A worked example is shown in Figure 5 (Appendix A.6).

Stage 3: Entity Normalization and Alias-aware Deduplication. While the focus of the previous step is validating triplet structure and semantics, this step aligns entity names to a unified vocabulary of existing KG entries to reduce duplication and ensure consistency of the constructed KG.

For each refined triplet, we link its subject and object names to existing entities in the KG that share the same entity type or a compatible parent type from the taxonomy. Using precomputed embeddings of entity aliases from the KG, we retrieve top-10 candidates and rank them by cosine similarity to the surface forms of the extracted mentions. The top-10 candidates, together with their types, are passed to the LLM to determine whether the extracted entity is synonymous with one of the existing entries. On a match, we replace the mention with the canonical KG label and store the surface form as an alias; otherwise, we preserve a new entity and add its surface form to the alias collection.

This step aims to ensure that the resulting KG is compact by avoiding redundant entities with different surface forms and evolving by supporting incremental updates with the discovery of new entities. A detailed example for the second step is provided in Figure 6 in Appendix A.6.

We implement **a final ontology verification step**

to ensure that extracted triplets comply with the structural and semantic constraints of the target KG. A triplet is verified if (i) its subject and object types, together with the relation, are defined in the ontology, and (ii) the relation's domain and range constraints are satisfied. Triplets that fail these checks are flagged as ontology-misaligned but retained, as they remain linked to the main KG through the entity name refinement step. Preserving them enables computation of an ontology alignment score and provides interpretable cues for identifying or revising schema-inconsistent facts.

2.3 Retrieval for QA

We address multi-hop question answering with the constructed KG via an iterative retrieval that decomposes the question into subquestions, grounding each step in the retrieved KG context.

Given a question, the LLM decomposes it into the first 1-hop subquestion. For each subquestion the LLM (1) identifies explicitly mentioned or potentially relevant entities; (2) links extracted entities to KG nodes and selects those most relevant for the current step; (3) given the retrieved subgraph formed by the neighborhood of the selected entities, generates an answer to the subquestion; (4) conditioned on the previous answer, the LLM formulates the next subquestion. This iterative process continues for up to five subquestions, after which the LLM produces the final answer. Implementation details and prompt templates are provided in Appendix A.5.

2.4 Evaluation

Existing benchmarks for triplet extraction suffer from substantial limitations: annotated closed IE datasets are small-scale, noisy, and often incomplete (Josifoski et al., 2023, 2022; Huguet Cabot and Navigli, 2021), while open IE corpora are difficult to align with real-world KGs and provide unreliable ground truth for evaluation (Stanovsky et al., 2018). Constructing high-quality datasets is costly, as annotators must not only identify all explicit and implicit entity and relation mentions but also align them to the complex schemas of large KGs such as Wikidata, which contain thousands of entity types and relations. Recently, the MINE benchmark (Mo et al., 2025) was introduced to address some of these issues by evaluating KGs through *information retention* rather than exact triplet-level supervision. We evaluated Wikontic on the MINE-1 task and observed much higher information retention

performance. However, while MINE provides a useful and scalable proxy for KG quality, it does not include complete ground-truth triplets. Therefore, it cannot measure classical precision or recall and instead captures only the degree to which a constructed KG preserves factual information.

To further address this, we adopt an alternative evaluation strategy. We measure KG quality through (1) structural compactness (i.e., non-redundancy and deduplication) and (2) performance in downstream multi-hop QA. In this setup, the LLM must answer factual questions in a text-free setting using only the constructed KG, without access to the original source texts, unlike retrieval-augmented methods such as HippoRAG (Gutierrez et al., 2024), AriGraph (Anokhin et al., 2025), and Holmes (Panda et al., 2024). This design makes QA a functional proxy for two key properties of an extracted KG: (a) factual correctness, since noisy or invalid triplets directly impede correct answers, and (b) coverage and completeness, since incomplete graphs restrict multi-hop reasoning. Despite existing advances in aligning KGs with LLMs and adapting for QA (Han and Shareghi, 2022; Dai et al., 2025; Sui et al., 2025; Pan et al., 2024), we deliberately refrain from training additional models to estimate the KG quality itself.

To compare predicted answers with ground truth, we apply a normalization procedure that lowercases all strings and removes punctuation. To account for lexical variation, we further expand entity matching using the alias mappings stored in the KG. If the model's predicted answer matches any canonical entity or one of its aliases, the corresponding alias set is treated as the set of valid candidate answers.

We perform evaluations on KGs extracted using different LLMs: `gpt-4.1`, `gpt-4.1-mini`, `gpt-4o-mini`[2], and `Llama-3.3-70b-Instruct`[3], and assess the quality of the resulting KG on two multi-hop QA datasets: MuSiQue (Trivedi et al., 2022) and HotpotQA (Yang et al., 2018). We used the same questions and candidate passages, including both supporting and distractor passages, used in HippoRAG (Gutierrez et al., 2024) and AriGraph (Anokhin et al., 2025) to compare the results with existing methods.
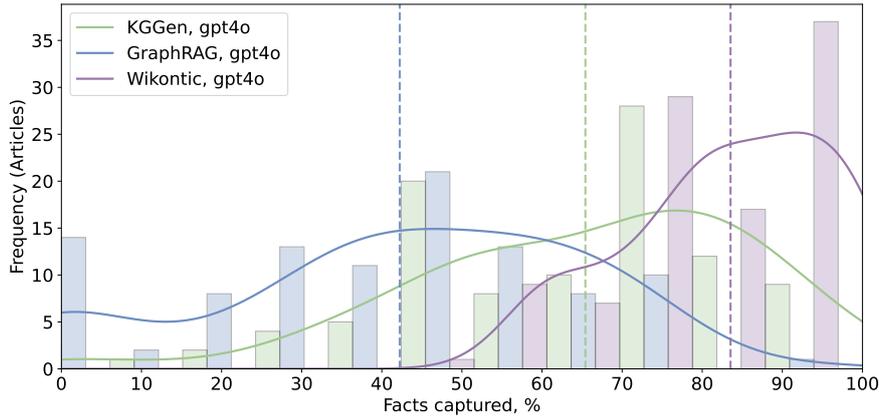

Figure 2: Distribution of MINE-1 scores across 100 articles for GraphRAG, KGGen, and Wikontic. Dotted vertical lines are averaged scores. Wikontic scored 84% on average, substantially outperforming GraphRAG 47.80% and KGGen 66%.

3 Results

3.1 Evaluations on MINE-1

We evaluated Wikontic on the MINE-1 benchmark, which measures how much factual information from the source text is retained in the constructed KGs using an LLM-as-a-judge protocol from the original study (Mo et al., 2025). Figure 2 displays the retention scores distribution in articles of MINE-1 for KGGen, GraphRAG, and Wikontic. Table 1 demonstrates the results for both KGGen and Wikontic with different LLM backbones. Wikontic consistently outperforms KGGen, reaching 84% with `gpt-4o` and 86% with `gpt-4.1-mini`, compared to KGGen's best score of 73% (Claude Sonnet 3.5). These results demonstrate that Wikontic effectively preserves factual information during the construction of the KG.

Method	MINE-1 Score (%)
KGGen, Claude Sonnet 3.5	73
KGGen, GPT-4o	66
KGGen, Gemini 2.0 Flash	44
GraphRAG, gpt4o	48
Wikontic, gpt4o	**84**
Wikontic, gpt4.1-mini	**86**

Table 1: MINE-1 information-retention scores for KGGen, GraphRAG, and Wikontic. Wikontic achieves the highest retention performance across all evaluated LLMs.

3.2 Graph Quality Analysis

We aim to comprehensively evaluate the structure, information content, and usability of knowledge

graphs created by Wikontic in a challenging information extraction setting. Given the limited availability of approaches that assess the KG quality directly, we use a proxy evaluation methodology based on the MuSiQue QA dataset to examine how effectively the knowledge is represented in the resulting KG.

A KG can be formally represented as $\mathcal{G} = (\mathcal{T}, \mathcal{E}, \mathcal{R})$, where $\mathcal{E}$ is the set of entities e, $\mathcal{R}$ is the set of relations r and $\mathcal{T}$ is the set of triplets: $\mathcal{T} \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$. For efficient knowledge storage and retrieval, the KG should satisfy the *size*, *density*, and *diversity* desiderata, which can be directly evaluated using graph statistics (Table 2).

Method	$|\mathcal{E}|$	$|\mathcal{R}|$	Avg. e degree	Unique e per r	r diversity per $2 \times e$
HippoRAG	234.9	130.1	4.0	1.8	1.1
AriGraph	228.0	115.6	3.9	2.0	1.01
Wikontic (1-3)	248.8	104.8	4.3	2.5	1.03
w/o ontology (2)	232.4	106.7	4.4	2.6	1.06
w/o ontology (2) and normalization (3)	273.0	140.9	4.2	2.3	1.09
w/o ontology-misaligned triplets	239.9	99.5	4.3	2.6	1.0

Table 2: KGs structural statistics for MuSiQue QA corpus: the number of unique entities ($|\mathcal{E}|$) and relations ($|\mathcal{R}|$), average entity degree (Avg. e degree), the number of unique entities per relation (Unique e per r) and the average relation diversity per two entities (r diversity per $2 \times e$).

Graph size is directly connected to the number of stored facts, and can be represented by the average number of entities and relations per MuSiQue sample, denoted by $|\mathcal{E}|$ and $|\mathcal{R}|$. Wikontic without ontology (Stage 2) and normalization (Stage
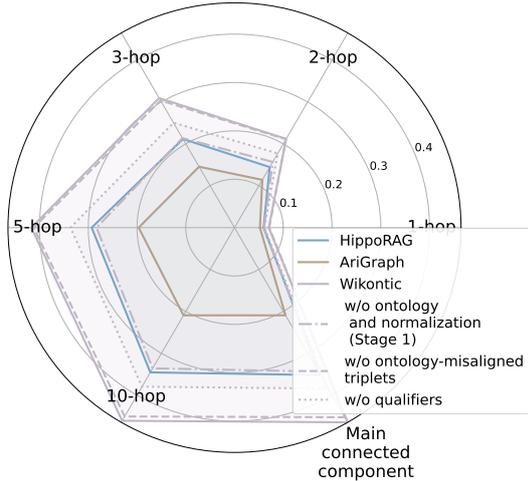
Figure 3: Wikontic produces the most dense KGs for MuSiQue questions. For each question, subgraphs are constructed around its entities, and their sizes are reported relative to the full KG. The figure shows the relative sizes of 1– to 10-hop neighborhoods and the entire connected component containing the question, defined as all nodes reachable from any question node.

3) yields diverse KGs with the highest number of unique entities and relations, followed by HippoRAG. However, the sheer volume of relations does not necessarily make retrieval more informative. Each relation should also be well-represented across various unique entities to ensure that relation names are standardized and meaningful. This property is reflected by the average number of unique entities per relation, which is significantly higher in refinement-augmented Wikontic versions.

High KG connectivity, represented by the average entity degree, ensures efficient retrieval, especially with limited search depth. Entity normalization (Stage 3) is a key to building KGs with the highest density among the compared methods. The relation diversity, or the number of unique relations per two entities, captures a variety of information stored in the KG. Here, Wikontic versions with relaxed ontology constraints remain the most diverse.

In a dense graph, important nodes should have many neighbors. We select the entities from MuSiQue questions and assess their neighborhood in graphs built by various methods (Figure 3). The largest possible neighborhood is the main connected component containing the given entities. In KGs built by the full Wikontic pipeline, each neighborhood contains the greatest number of entities, again underscoring strong KG connectivity and the importance of ontology.

3.3 Answer Coverage

In the task of question answering, the primary purpose of the extracted KG is to extract as much relevant information from the context as possible. Ideally, the resulting graph should contain all entities mentioned in the question as well as the answer to the question. In multi-hop question answering, entities in question will unlikely be in the same context as the answer entity, meaning they may not be direct neighbors in the resulting KG. However, in a KG with sufficient coverage, there has to be a reasonably short path connecting these two entities.

To measure the overall coverage of various KG construction methods, we estimate whether the answer to the question is present in the KG as an entity and whether the path from the question to the answer exists in the graph. Due to differences in pipelines, we cast all entity and relation names to lowercase and remove punctuation to standardize their format. To account for possible differences in entity naming, we consider two entities matching if one is a substring of the other.

Table 3 presents coverage and size metrics for KGs built by our pipeline, AriGraph, and HippoRAG on the MuSiQue dataset. Since baseline KGs are available only for the gpt4o-mini model and 80 common test samples, we use the same configuration to ensure a fair comparison. We estimated the standard deviation using bootstrapping. "Contains Answer" represents the percentage of cases when the answer entity is present in the neighborhood of entities from the MuSiQue question. We ablate Wikontic by removing one or multiple pipeline steps at a time. "Ontology Entailment" is

| Method | Contains Answer (%) | | | Ontology |
	Total	5-hop	10-hop	Entailment (%)
HippoRAG	$96.3_{\pm 2.1}$	$\mathbf{67.5}_{\pm 5.3}$	$68.8_{\pm 5.2}$	-
AriGraph	$79.9_{\pm 4.5}$	$40.0_{\pm 5.5}$	$41.3_{\pm 5.5}$	-
Wikontic	$96.2_{\pm 2.1}$	$66.3_{\pm 5.3}$	$68.8_{\pm 5.2}$	96.5
w/o ontology (2)	$\mathbf{97.5}_{\pm 1.7}$	$66.3_{\pm 5.3}$	$\mathbf{70.0}_{\pm 5.2}$	15.2
w/o ontology (2) and normalization (3)	$96.2_{\pm 2.1}$	$65.0_{\pm 5.4}$	$68.8_{\pm 5.2}$	12.4
w/o ontology-misaligned triplets	$93.8_{\pm 2.7}$	$63.8_{\pm 5.4}$	$66.3_{\pm 5.3}$	$\mathbf{100.0}$
w/o qualifiers	$85.0_{\pm 4.0}$	$51.3_{\pm 5.7}$	$53.8_{\pm 5.6}$	96.5

Table 3: Knowledge coverage of graphs built using different extraction methods on the MuSiQue dataset with mini-sized models. (Left) Percentage of cases where the correct answer to a question appears in the full constructed graph or within the 5-and 10-hop neighborhoods of the question nodes. Wikontic pipelines provide the best answer coverage while maintaining high ontology agreement. (Right) Average percentage of triplets in each sample that are entailed by the Wikidata ontology.

Method	MuSiQue		HotpotQA	
	EM	F1	EM	F1
Wikontic, gpt4.1	$46.8_{\pm 0.8}$	$\mathbf{59.8}_{\pm 0.3}$	$64.5_{\pm 0.4}$	$76.0_{\pm 0.4}$
Wikontic, gpt4.1-mini	$42.6_{\pm 0.7}$	$55.9_{\pm 0.3}$	$59.7_{\pm 0.6}$	$71.7_{\pm 0.8}$
Wikontic, gpt4o-mini	$42.1_{\pm 0.1}$	$53.3_{\pm 0.1}$	$53.7_{\pm 1.2}$	$65.8_{\pm 1.0}$
Full context, gpt4	33.5	42.7	53.0	68.4
Supporting facts, gpt4	45.0	56.0	57.0	73.8
ReadAgent (Lee et al., 2024), gpt4	35.0	45.1	48.9	62.0
GraphReader (Li et al., 2024), gpt4	38.0	47.4	55.0	70.0
GraphRAG (Edge et al., 2024), gpt4o-mini	40.0	53.5	58.7	63.3
AriGraph (Anokhin et al., 2025), gpt4o-mini	36.5	47.9	60.0	68.0
AriGraph (Anokhin et al., 2025), gpt4	45.0	57.0	**68.0**	74.7
HOLMES (Panda et al., 2024), gpt4	**48.0**	58.0	66.0	**78.0**
Wikontic, Llama 3.3	$37.7_{\pm 0.6}$	$\mathbf{49.7}_{\pm 0.4}$	$55.1_{\pm 0.5}$	$67.4_{\pm 0.5}$
HippoRAG v2 (Gutiérrez et al., 2025), Llama 3.3	37.2	48.6	**62.7**	**75.5**

Table 4: Exact Match (EM) and F1 scores on the MuSiQue and HotpotQA. Wikontic operates solely on KG triplets without accessing the source text, yet achieves performance comparable to or even exceeding both raw-text baselines (Full Context, Supporting Facts) and retrieval-augmented KG approaches that still rely on source text access.

the percentage of triplets where subject, object, and relation match the ontology.

Both Wikontic and HippoRAG achieve over 96% answer coverage, surpassing AriGraph's 79.9%. Notably, only 3.5% of triples in Wikontic are ontology-misaligned, confirming that the generated knowledge is largely schema-consistent; only a small portion of produced triplets requires correction to fully satisfy ontology constraints. Without ontology constraints, Wikontic reaches the highest answer coverage (97.5%), avoiding mismatches between Wikidata and MuSiQue entities and making it particularly effective for open-domain QA. When ontology consistency is required, the Wikontic variant that excludes ontology-misaligned triplets (100% ontology entailment) attains a competitive answer coverage of 93.8%.

These findings suggest that Wikontic variants provide the best solution both with and without the ontology. Ontology and misaligned triplets exclusion help to achieve the most standardized graph and strong answer coverage by thorough triplet refinement and deduplication strategies. Wikontic achieves the best coverage of information essential for question answering, while maintaining the highest connectivity levels, crucial for enabling graph search.

3.4 Computational Efficiency

We evaluated the computational efficiency of different KG-construction methods by counting the number of input (prompt) and output (completion) tokens required to build a KG from a single paragraph in the MuSiQue dataset. Table 5 presents

the estimated token-based costs for Wikontic, AriGraph, and GraphRAG, based on publicly available data and original implementations.

Tokens	Wikontic	AriGraph	GraphRAG
Prompt	12,687	11,000	115,000
Completion	881	2,500	20,000

Table 5: Mean token efficiency for KG construction per text paragraph of Wikontic compared with AriGraph and GraphRAG on the MuSiQue dataset.

A key indicator of computational cost is the number of *completion tokens*, which are typically around 3-5 times more expensive[4,5] and computationally intensive than input tokens (Zhou et al., 2024). Under this metric, Wikontic is more efficient, producing KGs with roughly three times fewer output tokens than AriGraph (881 vs 2,500) and about twenty times fewer than GraphRAG (881 vs 20,000). This shows that Wikontic achieves comparable KG construction quality while using significantly fewer output tokens.

3.5 Performance on QA Tasks

Table 4 reports Exact Match (EM) and F1 scores on the MuSiQue and HotpotQA datasets. Unlike retrieval-augmented approaches such as HippoRAG and AriGraph, which use KGs primarily to retrieve and process relevant text passages, our method performs reasoning directly over structured triplets without accessing the original documents.

Despite this constraint, using our triplet-only

---
[4] https://claude.com/platform/api/
[5] https://openai.com/api/pricing/

contexts for `gpt-4.1`, it achieves strong results, reaching 64.5 EM and 76.0 F1 on HotpotQA and 46.8 EM and 59.8 F1 on MuSiQue. These scores surpass several retrieval-based methods, including ReadAgent and GraphReader, and are comparable to more resource-intensive systems such as AriGraph and HOLMES that rely on richer textual context. This demonstrates that complete and well-structured symbolic representations of KGs can serve as a sufficient and reliable information source for multi-hop reasoning.

3.6   Ablation Study

Method variant	EM	F1
Wikontic (gpt4.1-mini)	$42.6_{\pm 0.7}$	$55.9_{\pm 0.3}$
w/o qualifiers	$23.9_{\pm 0.2}$	$39.4_{\pm 0.4}$
w/o aliases	$36.5_{\pm 0.8}$	$50.0_{\pm 1.4}$
w/o ontology (2)	$36.3_{\pm 1.2}$	$48.8_{\pm 1.0}$
w/o ontology (2) and normalization (3)	$27.0_{\pm 1.2}$	$36.9_{\pm 2.2}$
Single-step QA	$31.3_{\pm 0.6}$	$43.4_{\pm 0.6}$

Table 6: Ablations of the Wikontic pipeline on MuSiQue. "Single-step QA" omits iterative subquestion reasoning. Removing ontology or entity normalization yields the largest drop, highlighting their importance for accurate reasoning over the constructed KG.

To evaluate the contribution of individual Wikontic components, we conducted ablations (Table 6). Removing qualifiers leads to a substantial performance drop (–15.9 EM, –15.7 F1) on MuSiQue, indicating that qualifier information is essential for capturing fine-grained relational context. Excluding aliases (introduced at Stage 3) moderately decreases performance, confirming that alias expansion improves entity matching in question answering. Eliminating ontology integration (Stage 2) reduces both EM and F1, demonstrating the importance of type and schema constraints for consistent KG construction. When both ontology and entity normalization are removed (Stages 2 and 3), performance degrades most severely. The single-step QA variant also performs significantly worse, confirming that multi-hop question decomposition is essential for effective reasoning over the constructed KG. Overall, these findings show that each component contributes meaningfully to Wikontic's performance, with ontology-guided refinement and iterative retrieval being the most critical for downstream reasoning over the constructed KG.

4   Conclusions

We introduced Wikontic, a fully automated pipeline that uses LLMs to construct KGs from unstructured text. The pipeline produces compact, internally consistent graphs by aligning extracted triplets with the Wikidata ontology and deduplicating entities and relations. While using KGs as the sole knowledge source for multi-hop question answering, Wikontic achieves competitive performance with retrieval-augmented and KG-based baselines that still rely on source texts. Thus, ontology-guided KG construction is a viable alternative to passage-level retrieval. On MuSiQue, Wikontic includes 38–45 more unique entities than HippoRAG and AriGraph and contains the correct answer entity in 97.5% of cases. Within a 10-hop subgraph, it maintains 70% answer coverage and denser local connectivity. For QA, it attains 64.5 EM / 76.0 F1 on HotpotQA and 46.8 EM / 59.8 F1 on MuSiQue, outperforming text-reliant systems (ReadAgent, GraphReader) and approaching larger text-dependent methods (AriGraph, HOLMES). Moreover, Wikontic achieves state-of-the-art results on the MINE-1 benchmark, achieving 84–86% information-retention scores and surpassing GraphRAG and KGGen. These findings indicate that the proposed pipeline preserves a substantial amount of factual information from source texts.

Our approach is also token-efficient: during KG construction, Wikontic uses under 1,000 output tokens, about $3\times$ fewer than AriGraph and $1/20$ of GraphRAG, while preserving accuracy. Moreover, only 3.5% of extracted triplets are flagged as ontology-misaligned, indicating that nearly all generated knowledge is schema-consistent, minimizing the need for manual correction and significantly reducing annotation overhead. Beyond efficiency, the pipeline is adaptable: it can operate without an ontology or integrate domain-specific ontologies, enabling applications across specialized domains. Moreover, as LLMs increasingly serve as data generators, Wikontic provides a principled foundation for producing verified, structured KG data suitable for fine-tuning smaller task-specific models.

Our findings show that ontology-aware KG construction enables scalable, interpretable, and verifiable transformation of unstructured text into structured knowledge, bridging symbolic reasoning and generative language modeling.

Limitations

Our experiments are restricted to proprietary OpenAI models (GPT-4.1, GPT-4.1-mini, GPT-4o-mini) and the open-source Llama 3.3-70B. Token efficiency is measured as model-generated tokens during KG construction. This metric reflects provider billing but not end-to-end latency or throughput. Every stage of the pipeline currently uses LLMs with instructions and in-context examples prompting. Because the pipeline now yields its own annotated data, several stages could be replaced in future work by smaller, task-specific models fine-tuned on this data, thereby improving efficiency and lowering computational cost.

In the current work, we focus only on Wikidata due to its size, quality, and rich ontology. However, the proposed pipeline is flexible and can be adapted to any domain and ontology with a matching format of triplet constraints and entity types.

References

Petr Anokhin, Nikita Semenov, Artyom Sorokin, Dmitry Evseev, Andrey Kravchenko, Mikhail Burtsev, and Evgeny Burnaev. 2025. Arigraph: Learning knowledge graph world models with episodic memory for llm agents. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, pages 12–20. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Pere-Lluís Huguet Cabot and Roberto Navigli. 2021. Rebel: Relation extraction by end-to-end language generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2370–2381.

Hanzhu Chen, Xu Shen, Qitan Lv, Jie Wang, Xiaoqi Ni, and Jieping Ye. 2024. Sac-kg: Exploiting large language models as skilled automatic constructors for domain knowledge graph. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4345–4360.

Alla Chepurova, Yurii Kuratov, Aydar Bulatov, and Mikhail Burtsev. 2024. Prompt me one more time: A two-step knowledge extraction pipeline with ontology-based verification. In *Proceedings of TextGraphs-17: Graph-based Methods for Natural Language Processing*, pages 61–77.

Prafulla Kumar Choubey, Xin Su, Man Luo, Xiangyu Peng, Caiming Xiong, Tiep Le, Shachar Rosenman, Vasudev Lal, Phil Mui, Ricky Ho, and 1 others. 2024. Distill-synthkg: Distilling knowledge graph synthesis workflow for improved coverage and efficiency. *arXiv preprint arXiv:2410.16597*.

Xinbang Dai, Yuncheng Hua, Tongtong Wu, Yang Sheng, Qiu Ji, and Guilin Qi. 2025. Large language models can better understand knowledge graphs than we thought. *Knowledge-Based Systems*, 312:113060.

Bayu Distiawan, Gerhard Weikum, Jianzhong Qi, and Rui Zhang. 2019. Neural relation extraction for knowledge base enrichment. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 229–240.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2024. Lightrag: Simple and fast retrieval-augmented generation.

Bernal Gutierrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. Hipporag: Neurobiologically inspired long-term memory for large language models. volume 37, pages 59532–59569.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. 2025. From rag to memory: Non-parametric continual learning for large language models. *arXiv preprint arXiv:2502.14802*.

Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh Halappanavar, Ryan A Rossi, Subhabrata Mukherjee, Xianfeng Tang, and 1 others. 2024. Retrieval-augmented generation with graphs (graphrag). *arXiv preprint arXiv:2501.00309*.

Jiuzhou Han and Ehsan Shareghi. 2022. Self-supervised graph masking pre-training for graph-to-text generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4845–4853, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Pere-Lluís Huguet Cabot and Roberto Navigli. 2021. Rebel: Relation extraction by end-to-end language generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, Online and in the Barceló Bávaro Convention Centre, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research*.

Martin Josifoski, Nicola De Cao, Maxime Peyrard, Fabio Petroni, and Robert West. 2022. GenIE: Generative information extraction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4626–4643, Seattle, United States. Association for Computational Linguistics.

Martin Josifoski, Marija Sakota, Maxime Peyrard, and Robert West. 2023. Exploiting asymmetry for synthetic training data generation: SynthIE and the case of information extraction. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1555–1574, Singapore. Association for Computational Linguistics.

Kuang-Huei Lee, Xinyun Chen, Hiroki Furuta, John Canny, and Ian Fischer. 2024. A human-inspired reading agent with gist memory of very long contexts. In *Forty-first International Conference on Machine Learning*.

Shilong Li, Yancheng He, Hangyu Guo, Xingyuan Bu, Ge Bai, Jie Liu, Jiaheng Liu, Xingwei Qu, Yangguang Li, Wanli Ouyang, Wenbo Su, and Bo Zheng. 2024. GraphReader: Building graph-based agent to enhance long-context abilities of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 12758–12786, Miami, Florida, USA. Association for Computational Linguistics.

Belinda Mo, Kyssen Yu, Joshua Kazdan, Proud Mpala, Lisa Yu, Charilaos I. Kanatsoulis, and Sanmi Koyejo. 2025. KGGen: Extracting knowledge graphs from plain text with language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*, 36(7):3580–3599.

Pranoy Panda, Ankush Agarwal, Chaitanya Devaguptapu, Manohar Kaul, and Prathosh Ap. 2024. Holmes: Hyper-relational knowledge graphs for multi-hop question answering using llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13263–13282.

Fina Polat, Ilaria Tiddi, and Paul Groth. 2025. Testing prompt engineering methods for knowledge extraction from text. *Semantic Web*, 16(2):SW–243719.

Gabriel Stanovsky, Julian Michael, Luke Zettlemoyer, and Ido Dagan. 2018. Supervised open information extraction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 885–895.

Yuan Sui, Yufei He, Zifeng Ding, and Bryan Hooi. 2025. Can knowledge graphs make large language models more trustworthy? an empirical study over open-ended question answering. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12685–12701, Vienna, Austria. Association for Computational Linguistics.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Musique: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Denny Vrandečić. 2012. Wikidata: a new platform for collaborative data collection. In *Proceedings of the 21st International Conference on World Wide Web*, page 1063–1064, New York, NY, USA. Association for Computing Machinery.

Chenguang Wang, Xiao Liu, and Dawn Song. 2020. Language models are open knowledge graphs. *arXiv preprint arXiv:2010.11967*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Kg-bert: Bert for knowledge graph completion. *arXiv preprint arXiv:1909.03193*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th international conference on computational linguistics: technical papers*, pages 2335–2344.

Ranran Haoran Zhang, Qianying Liu, Aysa Xuemo Fan, Heng Ji, Daojian Zeng, Fei Cheng, Daisuke Kawahara, and Sadao Kurohashi. 2020. Minimize exposure bias of seq2seq models in joint entity and relation extraction. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 236–246.

Zixuan Zhou, Xuefei Ning, Ke Hong, Tianyu Fu, Jiaming Xu, Shiyao Li, Yuming Lou, Luning Wang, Zhihang Yuan, Xiuhong Li, and 1 others. 2024. A survey on efficient inference for large language models. *arXiv preprint arXiv:2404.14294*.

A Appendix

A.1 Reproducibility

We release[6]: (1) prompts and source code for each pipeline stage; (2) scripts to build Wikidata-derived ontology used for ontology-aware stages; (3) the KG-only multi-hop QA component.

We reported metrics averaged across multiple runs (Tables 4, 6) with standard deviation. For Table 3, standard deviations are estimated via bootstrap resampling. The exact prompts used at each stage are provided in the Appendix A.4, A.5.

A.2 Dataset Statistics

The test splits of HotpotQA and MuSiQue consist of 1000 samples each. We provide the QA evaluation results for Wikontic and HippoRAG for the whole evaluation set, and for AriGraph, we report the openly available results for 200 test samples. To compare the statistics of KGs, we use 80 MuSiQue samples that are commonly available for all compared methods.

A.3 Computational resources

All experiments on knowledge graph (KG) construction and question answering (QA) were conducted using the OpenAI and OpenRouter APIs. Across all datasets, models, and ablation studies, KG construction and QA (each QA experiment repeated three times to compute mean and standard deviation) required a total cost of approximately $500.

A.4 Prompts for triplet extraction

Here we provide excerpts of prompts that were used in our KG construction pipeline. Prompt 1 was used for candidate triplet extraction. Subsequently, Prompt 2 was used to refine entity types for both subject and object entities. Prompt 3 was used for choosing relations among those that can legally connect chosen entity types by Wikidata constraints. Finally, Prompt 6 was used to refine surface forms of subject and object. All prompts, instructions, and in-context examples are available with the code.

---

[6]https://github.com/screemix/Wikontic

Prompt 1. Candidate Triplet Extraction

You are an algorithm designed to extract structured knowledge from texts to build a Wikidata-like knowledge graph. A knowledge graph consists of **triplets** in the format (subject, relation, object), where:

- Subject: A named entity or concept describing a group of people, events, or abstract objects that serves as the source of the relation.
- Relation: A Wikidata-style predicate connecting the subject and the object.
- Object: A named entity or concept describing a group of people, events, or abstract objects related to the subject.

Additionally, some triplets may have **qualifiers** that provide more context (e.g., date, place, or other attributes). Qualifiers should have relations and objects like triplets do, but instead of a subject, their relation connects an object and the triplet they qualify. Qualifiers must always be attached to a triplet and never exist as standalone triplets.

You will receive a text labeled **"Text:"**. Your task is to extract meaningful triplets that represent factual relationships.

Output Format. Return only triplets in **JSON format** as a list of dictionaries:

- "subject": Subject entity.
- "relation": Relation connecting subject and object.
- "object": Object entity.
- "qualifiers": List of dictionaries, each with:
    - "relation": Relation connecting triplet and object.
    - "object": Object entity connected to the main triplet.
- "subject_type": Class that describes the subject.
- "object_type": Class that describes the object.

Prompt 2. Triplet Backbone Refinement - choosing Relevant Entity Types

You are given a factual triplet extracted from text. The triplet follows the format (subject, relation, object), where:

- Subject: A named entity or concept that represents a person, group, event, or abstract entity serving as the source of the relation.

- Relation: A Wikidata-style predicate that defines the connection between the subject and the object.

- Object: A named entity or concept that represents a person, group, event, or abstract entity related to the subject.

- Subject type: a class that describes the object.

- Object type: a class that describes the subject.

The extracted entity types of both subject and object were mapped to a set of similar Wikidata-style entity types based on semantic similarity.

Your Task:

You will be provided with the following:

- Text: The original sentence or passage from which the triplet was extracted.

- Extracted Triplet: The factual triplet derived from the text.

- Candidate subject types: similar entity types for subject type of extracted triplet retrieved from Wikidata.

- Candidate object types: similar entity types for object type of extracted triplet retrieved from Wikidata.

Select the most appropriate candidate entity types for both subject and object from the provided candidates that best match the meaning of previously extracted triplet and original text.

Provide ONLY an answer in JSON format with the following keys:

- "subject_type": Selected subject type candidate.

- "object_type": Selected object type candidate.

Prompt 3. Triplet Backbone Refinement - choosing Relevant Relation

You are given a factual triplet extracted from text. The triplet follows the format (subject, relation, object), where:

- Subject: A named entity or concept that represents a person, group, event, or abstract entity serving as the source of the relation.

- Relation: A Wikidata-style predicate that defines the connection between the subject and the object.

- Object: A named entity or concept that represents a person, group, event, or abstract entity related to the subject.

- Subject type: a class that describes the object.

- Object type: a class that describes the subject.

The extracted relation has been mapped to a set of similar Wikidata-style relations based on semantic similarity and the entity types they can connect.

Your Task:

You will be provided with the following:

- Text: The original sentence or passage from which the triplet was extracted.

- Extracted Triplet: The factual triplet derived from the text.

- Candidate relations: list of relation (or in other words property) names similar to the extracted relation from triplet retrieved from Wikidata.

- Candidate relations: list of relation (or in other words property) names similar to the extracted relation from triplet retrieved from Wikidata.

Select the most appropriate relation candidate from the provided candidate triplets that best match the meaning of previously extracted triplet and original text.

Provide only an answer in JSON format with the following keys:

- "relation": Relation for the selected triplet.

Prompt 4. Entity Names Refinement

In the previous step, there was extracted a triplet akin to one in Wikidata knowledge graph from the text. Triplet contains two entities (subject and object) and one relation that connects these subject and object. Using semantic similarity, we linked subject name with top similar exact names from the knowledge graph built from previously seen texts.

You will be provided with the following:

- Text: The original sentence or passage from which the triplet was extracted.

- Extracted Triplet: A structured representation in the format "subject": "...", "relation": "...", "object": "..." .

- Original Subject: A subject name that needs refinement.

- Candidate Subjects: A list of possible entity names from previously seen texts.

Your Task:

Select the most contextually appropriate subject name from the Candidate Subjects list that best matches subject from extracted triplet and context of the given Text.

- If an exact or semantically appropriate match is found, return the corresponding name exactly as it appears in the list.

- If no suitable match exists, return the string "None".

- Do not modify name from the candidate list in case of match, add explanations, or provide any additional text.

A.5 Prompts for question answering

Here, we provide excerpts of prompts that were used for KG grounded question answering. Prompt 5 was used to extract entities relevant to the question. Then, with Prompt 6 the LLM was instructed to choose relevant entities among entities similar to the extracted ones in the KG. Prompt 7 was used to decompose the question on a single-hop subquestion conditioned on extracted entities or previously answered subquestions. Prompt 8 was used to check if a question is answered by a sequence of subquestions and corresponding answers. All prompts, instructions, and in-context examples are available with the code.

Propmt 5. Entity extraction for question answering

Extract wikidata-like entities from the question below. It is guaranteed that there is at least one mentioned entity.

Extract any entity, whether name entity or an abstract entity, that might help retrieve the information to answer the question.

Provide output in json format, no additional symbols. Output should be represented as a LIST of extracted entities' names.

Prompt 6. Entity linking for question answering

Task: Identify relevant entities from a pre-constructed knowledge graph that might help to answer a provided question.

Input Structure:

- The question will be labeled as "Question:".

- A list of entities from the knowledge graph will be labeled as "Entities:".

Selection Criteria:

- Relevance means an entity is directly or indirectly useful for answering the question. Look for names, events, dates, and other related concepts or entities that match or connect to key concepts in the question.

- Do not ignore possible indirect relevance (e.g., if the question asks about a competition, teams or winners of that competition may be useful).

Response Format:

- Always return at least one relevant entity. It is guaranteed that there is at least one.

- The output must be a JSON list of dictionaries, where each dictionary contains a key "entity": the name of the chosen relevant entity

- Do not return an empty list. Select the best possible options.

Prompt 7. Question decomposition

You are an assistant for stepwise question decomposition.

You will be given three inputs:

- An original multi-hop question.

- A 1-hop sub-question that has already been answered.

- The answer to that 1-hop sub-question.

Your task:
 Reformulate the original multi-hop question by integrating obtained answer from sub-question, so the new question has (n-1) hops.

Rules:

- Only perform one reasoning hop at a time. Do not generate additional reasoning steps beyond this hop.

- Do not include explanations or text, just reformulated question.

Prompt 8. Check if a question is answered

You are a reasoning assistant for multi-hop question answering.

Your task: Decide whether a list of subquestions and their answers fully resolves the original multi-hop question.

Input format:

- Original multi-hop question: <text>

- Question->answer sequence: [a list of subquestions and their answers, ending with the most recent one]

Output rules:

- If the sequence of subquestions and answers completely and directly resolves the original multi-hop question, output only the final answer to the original multi-hop question (not just the last subanswer, i.e. answer the original question).

- If the sequence is not sufficient and more reasoning or hops are needed, output exactly: NOT FINAL

Do not include any prefixes like "Final answer:", "Answer:", suffixes, formatting, original questions or explanations.

Output must be a single line: either string with the final answer to the original multi-hop question or the exact string NOT FINAL.

<example>

 Original multi-hop question: Who was the spouse of the person who wrote The Iron Heel?

 Question->answer sequence:

  Who wrote The Iron Heel? → Jack London

  Who was the spouse of Jack London? → Charmian London

 Expected output:

 Charmian London
</example>

<example>

 Original multi-hop question: Which country's capital is closest to the birthplace of Nikola Tesla?

 Question->answer sequence:

  Where was Nikola Tesla born? → Smiljan, Croatia

Expected output:

 NOT FINAL
</example>
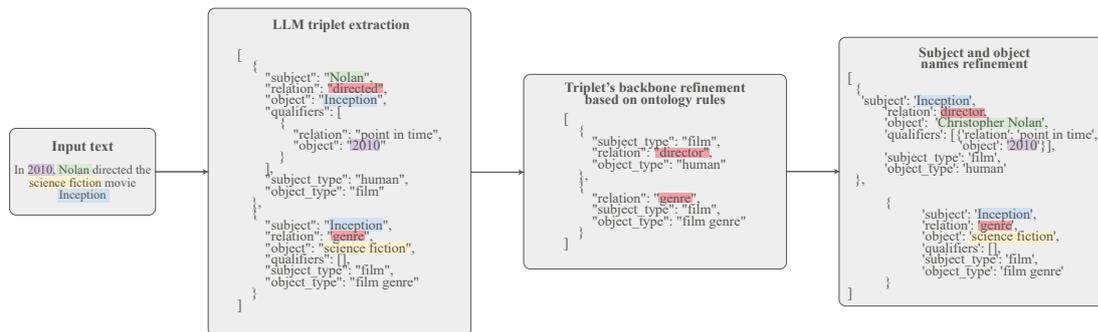
A.6 Triplet extraction pipeline examples

Figure 4: Overview of the multi-stage pipeline for KG extraction from unstructured text. The process consists of (1) LLM-based triplet extraction, (2) ontology-based validation of triplet structure, and (3) entity linking and normalization.
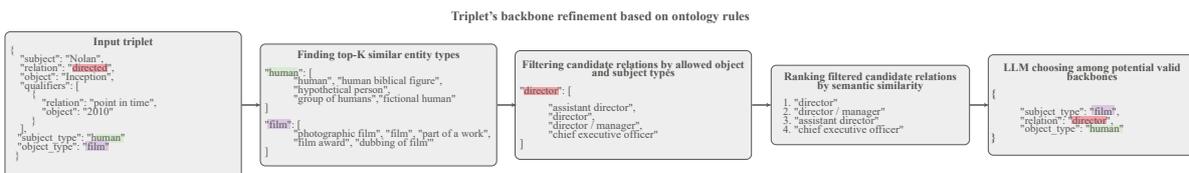

Figure 5: Ontology-based triplet refinement process. For each extracted triplet, we retrieve and extend candidate entity types using Wikidata's type hierarchy, identify valid relations allowed to use between extracted entities based on ontology constraints, and re-rank relation candidates using semantic similarity. The final triplet configuration is selected by an LLM.
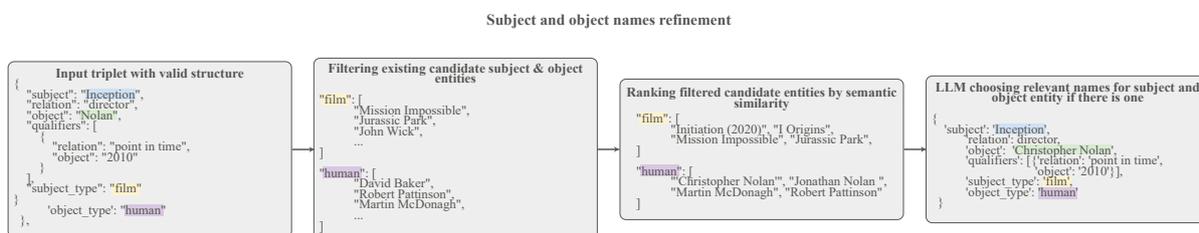

Figure 6: Entity refinement step for KG construction. For each refined triplet, candidate subject and object entities are retrieved from the existing KG based on their type and semantic similarity. An LLM determines whether the extracted entity matches an existing one or should be preserved as a new entry. This process reduces redundancy and supports incremental KG updates.