

LMKG: a flexible method for ontology-aware Knowledge Graph Extraction from text

Gianmarco Pappacoda¹ and Paolo Torroni¹

¹DISI, University of Bologna
{gianmarco.pappacoda, p.torroni}@unibo.it

Abstract

Knowledge Graphs (KGs) are a practical representation of information that combines structural flexibility with expressiveness. With the recent advent of Large Language Models (LLMs), it has become possible to extract KGs from plain text. However, extracting facts from text is only one part of building a usable knowledge graph. A construction system must also reconcile entity mentions, control its relation vocabulary, integrate information across passages, and preserve a traceable connection between graph facts and their source. We present LMKG, a modular LLM-based pipeline that supports both open and ontology-constrained extraction. We also propose two construction strategies for long documents: *fold*, which accumulates a graph over successive chunks, and *enrich*, which combines whole-document and chunk-level extraction before graph refinement. We evaluate LMKG on three standard benchmarks: Text2KGBench, MINE-1, and HotpotQA. We find that our ontology conditioning substantially improves conformance to a given ontology in Text2KGBench. Our system consistently recovers facts, reaching the highest reported fact recovery of 90.80% in MINE-1. We observe degraded downstream question-answering performance on HotpotQA when supplying graphs rather than full passages.

1 Introduction

Most of the knowledge in the world is stored as unstructured text. News and scientific articles, technical documentation, and online social media texts all fall into this category. *Knowledge Graphs* (KGs) offer a different, structured representation of such information. The information in KGs usually follows an *ontology*: a schema of the types and relations allowed in the graph. This representation is auditable and presents textual facts in a structured form that can also be used by retrieval-augmented generation systems (Lewis et al., 2020). Nevertheless,

most general-purpose knowledge graphs, such as DBpedia, YAGO, and Wikidata (Auer et al., 2007; Suchanek et al., 2007; Vrandečić and Krötzsch, 2014), are incomplete and difficult for humans to inspect without purpose-built software. By contrast, small, high-quality, hand-curated graphs are more focused on a specific task but remain expensive to construct.

With the rise of LLMs, it has become possible to tackle the problem of text-to-KG extraction without employing expensive hand-crafted ontologies. Classical closed Information Extraction (cIE) assumes a predefined inventory of entity and relation types, often derived from an existing KG or ontology. Such constraints produce a consistent representation, but make the extractor difficult to transfer to new domains and unable to express facts outside its schema. Open Information Extraction (OpenIE) instead derives entities and relations directly from text (Angeli et al., 2015). Recent language models make this setting particularly attractive because they can generate structured information without task-specific training data.

Flexibility alone, however, does not produce a usable graph. Open extractors may represent the same entity under several surface forms, generate many overlapping relation labels, or fail to reconcile facts extracted from different passages. As these variations accumulate, the resulting graph becomes sparse and fragmented. Moreover, generated entities and relations may no longer retain a clear connection to the passages that support them. These problems weaken the consistency, interpretability, and auditability that motivate KG construction in the first place.

We present LMKG, a modular language-model-based pipeline for both open and ontology-constrained KG construction. The pipeline separates entity extraction, entity canonicalization, relation extraction, evidence resolution, and evidence refinement. An optional ontology constrains entity

extraction, relation extraction, and refinement without changing the remaining construction process. Every predicted entity and triple is also paired with a model-proposed source quote, which deterministic matching resolves to a character span when possible. This gives the resulting graph an explicit connection to its textual evidence.

Constructing a graph from a long document introduces an additional integration problem: local chunks provide manageable extraction contexts, but important entities and relations may cross chunk boundaries. Therefore, we also introduce two document-level strategies.

We evaluate LMKG through three questions:

RQ1: Ontology control. How does supplying an ontology affect extraction accuracy, conformance, and hallucination?

RQ2: Long-document construction. How do sequential and global-first construction strategies differ in fact recovery and graph size?

RQ3: Graph utility. Is the extracted knowledge graph useful for a downstream task when compared to the original text it was extracted from?

2 Related Work

From extraction to graph construction. OpenIE extracts relational tuples without a predefined inventory (Angeli et al., 2015), while supervised document-level systems such as DyGIE++ jointly model entities, relations, and events under task-specific annotations (Wadden et al., 2019). Generative IE serializes structured outputs: REBEL predicts relation triples over a broad inventory (Huguet Cabot and Navigli, 2021), and GenIE constrains decoding to a predefined KB schema (Josifoski et al., 2022).

LLM-based construction and canonicalization. EDC organizes open construction by extracting, defining, and canonicalizing a schema, then mapping open predicates into it (Zhang and Soh, 2024). KGGen separates entity and relation extraction from entity and edge resolution, then evaluates whether its graph retains source information with MINE-1 and its downstream usefulness with MINE-2 (Mo et al., 2025). GraphRAG constructs an entity graph and community summaries for corpus-level retrieval and query-focused summarization (Edge et al., 2024). COMBO distinguishes entity, relation, and ontology canonicalization in open KGs (Jiang et al., 2023). Wikontic instead uses Wikidata-aligned type and relation con-

straints together with entity normalization (Chepurova et al., 2026), and schema-adaptable KGC studies generalization when extraction schemas change (Ye et al., 2023). We develop LMKG as an intersection of these settings: one representation and program support both open and supplied-schema extraction, while canonicalization, refinement and knowledge graph construction strategies remain interchangeable.

Grounding, refinement, and evaluation. Iterative construction can improve coverage but also amplify unsupported edges. SAC-KG uses a generator-verifier-pruner loop for domain KG construction (Chen et al., 2024); RefChecker extracts claim triplets and verifies them against a reference (Hu et al., 2024). We propose a narrower mechanism that asks for a local quote with each prediction and resolves it to source offsets deterministically before source-conditioned KG refinement. This provides an anchor in the actual text for extracted facts.

3 Methodology

3.1 Knowledge Graph and Ontology Representations

A knowledge graph is a pair $G = (\mathcal{E}, \mathcal{R})$. The set $\mathcal{E}$ holds the entities, each with a name and a type from an entity-type vocabulary $\mathcal{T}_{\text{ent}}$. The set $\mathcal{R} \subseteq \mathcal{E} \times \mathcal{E}$ holds the relations. A relation $re \in \mathcal{R}$ is paired with a type from a relation-type vocabulary $\mathcal{T}_{\text{rel}}$. Each relation is therefore expressed either as a triple of the form $re = (s, r, o)$ or in the functional form $re = r(s, o)$ with $s, o \in \mathcal{E}$, $r \in \mathcal{T}_{\text{rel}}$. An ontology is the triple $\mathcal{O} = (\mathcal{T}_{\text{ent}}, \mathcal{T}_{\text{rel}}, \mathcal{T}_{\text{sig}})$, where the set of allowed signatures is defined as $\mathcal{T}_{\text{sig}} \subseteq \mathcal{T}_{\text{rel}} \times \mathcal{T}_{\text{ent}} \times \mathcal{T}_{\text{ent}}$. Under a given ontology $\mathcal{O}$, entity recognition should type every mention with a type from $\mathcal{T}_{\text{ent}}$, and relation extraction may use only relation types from $\mathcal{T}_{\text{rel}}$ and signatures allowed by $\mathcal{T}_{\text{sig}}$.

3.2 LMKG

We developed our system as a streamlined pipeline that uses a mixture of components: reusable DSPy modules for LLM calls (Khatab et al., 2024), deterministic evidence resolution, and adaptable chunking strategies. In our system, every entity and every triple additionally carries provenance (q, c, p) : an evidence quote q , the chunk c it came from, and a character span p in the source document. The

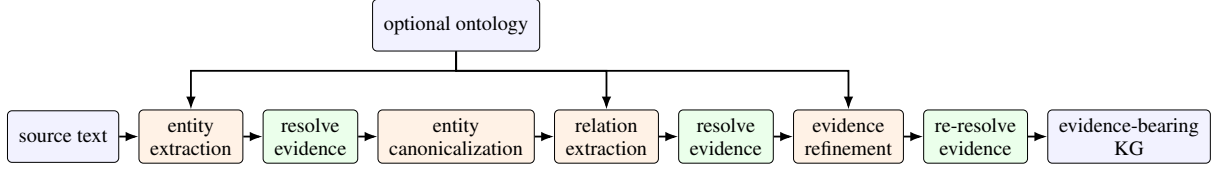


Figure 1: The LMKG extraction pipeline. Orange boxes contain language-model decisions; green boxes map proposed evidence quotes to source offsets deterministically. The optional ontology constrains the label vocabulary. Fold and enrich compose this pipeline differently for long documents (Figures 2 and 3).

model supplies q alongside the prediction it justifies. The deterministic resolver locates q in the source and records its character span p ; when no match exists, p remains empty rather than receiving an LLM-guessed offset. Table 1 shows two triples with their quotes and spans. We define the pipeline as shown in Figure 1 as the following series of steps:

Named Entity Recognition (NER) recognizes named entities in the supplied text.

Evidence resolution is a deterministic mechanism that maps each entity to a given quote, document chunk and numeric span.

Canonicalization merges coreferent mentions in the text and expresses each cluster using a single canonical name. The graph is then deterministically re-derived to account for the entity clusters while remapping the evidence. Furthermore, if canonicalization occurs after relations are already present, it accounts for the changes to the entities.

Relation Extraction (RE) recognizes relations starting from entities extracted in the previous step and the supplied text.

Evidence resolution is the same mechanism as above also applied to extracted relations.

Evidence refinement is a step that takes as input a graph in which each entity or triple is associated with provenance and attempts to add missing triples and remove unsupported ones through critique. This step can be repeated multiple times.

Evidence resolution is the same mechanism as above applied to the whole graph.

3.3 Local Extraction with Evidence

The pipeline alternates model decisions with deterministic resolution (Figure 1). A language model identifies the entities/triples and returns a short verbatim quote for each; the resolver then maps

Triple	Evidence quote	Span
(Joshua Parlby, born_in, Longton)	“born 1855 in Longton, Staffordshire”	30–65
(Joshua Parlby, managed, Manchester City)	“managed Manchester City in the 1890s”	103–139

Table 1: Two triples from a constructed HotpotQA graph. The quote is requested from the model with the proposition; the character span is assigned by deterministic matching against the source and is absent when no match is found.

those quotes to character offsets, trying exact, case-insensitive, and whitespace-tolerant matching.

An *evidence refiner* iteratively critiques entities and triples against the source. In each iteration it can remove unsupported items and add missing ones, stopping at a configured iteration limit.

3.4 Long Documents: Fold and Enrich

A prominent problem when extracting KGs from text using LLMs is posed by long documents. The long context may hinder the capabilities of an LLM to perform extraction. We propose two strategies to tackle this problem: *Fold* and *Enrich*.

Both long-document strategies split the text into chunks with recorded starting offsets and perform evidence-aware extraction within each chunk. The difference lies in how KGs are constructed and aggregated. *Fold* (Figure 2) accumulates graph triples and conditions each subsequent chunk extraction; *Enrich* (Figure 3) performs a whole-document extraction without refinement. Both strategies then perform document-level entity canonicalization followed by evidence resolution, aligning chunks with the original document spans.

Fold (sequential). The system starts with an empty graph and visits chunks in order. Triples from earlier chunks are supplied as anchors to subsequent extraction in later chunks. This is impor-

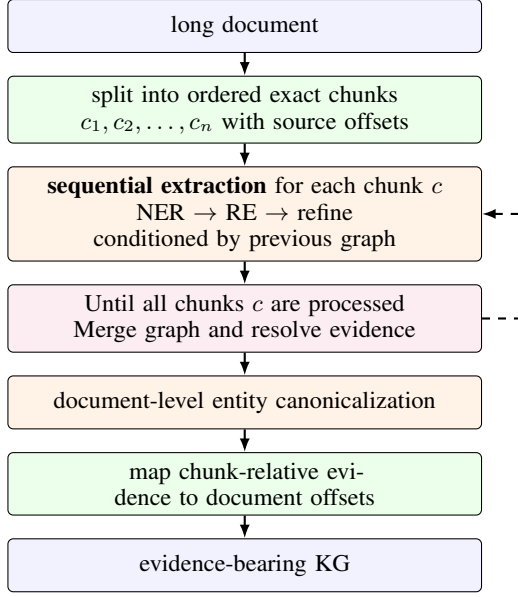


Figure 2: **Fold** strategy visits chunks sequentially. Later relation extraction can refer to entities accumulated from earlier chunks; canonicalization and final evidence resolution happen once after all chunks.

tant for RE because it allows a later proposition to point back to an earlier mention. Locally extracted entities/triples accumulate gradually and are followed by one final document-level canonicalization pass.

Enrich (global-first). The system first runs the complete pipeline over the entire document. It then extracts every chunk independently and merges local results with the base graph. After global canonicalization, a structure-only graph refiner proposes strongly implied missing edges among existing nodes. Enrich therefore uses more model calls and can add more coverage, but its structure-only completion is not source-grounded and must be evaluated for precision separately.

4 Experimental Setup

Every experiment uses gpt-5.4-mini¹ as the extraction and question-answering model.

We select three benchmarks with which to evaluate our system and answer our research questions:

- **Text2KGBench**: a text-to-knowledge-graph benchmark that also includes ontologies (RQ1).
- **MINE-1**: a recall-oriented benchmark that

¹Accessed via the OpenAI API (GPT-5.4-mini); see <https://platform.openai.com/docs/models>, July 2026.

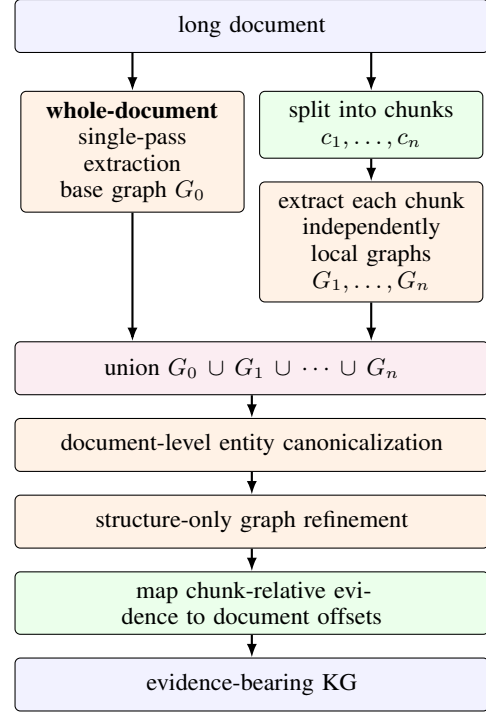


Figure 3: **Enrich** combines a whole-document base graph with independent chunk graphs before global canonicalization. It then adds a structure-only completion pass (i.e., without the document as context).

measures fact retrieval in long documents (RQ2).

- **HotpotQA**: a QA benchmark that supplies a document as context for the question (RQ3).

We reproduced all experiments as faithfully as possible, given the availability of the original data and upstream code.

4.1 Text2KGBench

We compare the same extraction pipeline firstly omitting the ontology and secondly providing such ontology on both Text2KGBench splits: Wikidata-TekGen ($n = 4062$, 19 ontologies), DBpedia-WebNLG ($n = 2014$, 10 ontologies). Metrics are the benchmark’s macro-averaged triple precision, recall, and F1, ontology conformance, and subject, relation, and object hallucination rates. For definitions of these metrics, we refer the reader to the original paper (Mihindukulasooriya et al., 2023). Both conditions follow the benchmark’s own prompting protocol: one in-context example per test sentence. We selected them using the benchmark’s precomputed test–train similarity ranking. Because the input is sentence-level, we run the extractor without chunking, evidence

refinement, or canonicalization.

4.2 MINE-1

MINE-1 contains 100 articles with 15 annotated facts each (Mo et al., 2025). We construct one open-schema graph per article with fold and enrich, using structure-aware chunks (i.e., taking into account paragraphs and blank lines). We use two evidence refinement rounds.

Scoring follows the benchmark’s retrieve-then-judge procedure: each fact is embedded with a SentenceTransformers encoder (Reimers and Gurevych, 2019)². The eight most similar nodes are retrieved, their two-hop neighborhoods are rendered as text, and a gpt-5³ judge at high reasoning effort adjudicates entailment; this configuration reproduces the reference judge. The outcome is the percentage of source facts recoverable from the graph, averaged by article.

For external comparison, we include the KGGen, GraphRAG (Edge et al., 2024), and OpenIE (Angeli et al., 2015) results reported by Mo et al. (2025), as well as the Wikontic results reported by Chepurova et al. (2026).

4.3 HotpotQA

We use a subset ($n = 2500$) of the HotpotQA (Yang et al., 2018) distractor set in a closed-book per-item setting. We run two experiments: one in which the full passages are provided to establish a baseline and another in which a KG produced by LMKG is provided instead. For each question a graph is built from the passages supplied with that question, and the answerer receives graph facts instead of the passages. The graph is built on open schema by a single full-document extraction pass with two evidence refinement rounds and LLM canonicalization. The text configuration emits no sentence-level attribution, so its supporting-fact and joint scores are structurally zero and are not useful for comparison.

5 Results and discussion

5.1 Ontology conditioning: Text2KGBench (RQ1)

Ontology conditioning aligns the produced vocabulary with the target ontology schema. We can

²We used all-MiniLM-L6-v2, the original embedding model used in KGGen.

³Accessed via the OpenAI API (GPT-5); see <https://platform.openai.com/docs/models>, July 2026

observe that this matters most in Text2KGBench, in which labels are constrained and not easily inferable (Table 2). LMKG without ontology conditioning performs poorly (F1 0.08–0.17) compared to the same system with ontology conditioning (F1 0.31–0.39). Part of this gap is definitional. Conformance and relation hallucination are defined against the supplied ontology, so an open-schema run that produces its own predicate labels is almost certain to score badly on both: the 0.11 and 0.37 conformance figures are largely attributed to open extraction not using the ontology’s vocabulary. All four experiments issue about the same number of model calls per sentence (2.15–2.23), and cost per sentence differs by under 9% in either direction: TekGen (\$0.0031 versus \$0.0034); DBpedia (\$0.0034 versus \$0.0036).

5.2 Long documents and factual recall: MINE-1 (RQ2)

MINE-1 asks whether a graph contains enough information to recover known source facts. Our system consistently recovers facts from the text, achieving a high MINE-1 score with both strategies. Enrich recovers more source facts than fold: 90.80% against 84.87% (Table 3), the highest reported score on this benchmark. This gain comes with substantially larger graphs: enrich yields 80.1 edges per article on average against fold’s 46.9, and 81.4 versus 67.2 entities. Because MINE-1 measures fact recovery and it does not penalize unsupported edges, this score should be read as coverage.

5.3 Question Answering: HotpotQA (RQ3)

The HotpotQA experiment shows that providing the full passages to the LLM produces significantly better results than providing LMKG-produced graphs as context (Table 4). Two explanations are possible: the graphs may omit information needed to answer, or the graph representation may be unsuitable for LLM consumption. Indirect evidence favours the latter as a contributing factor. On MINE-1, LMKG recovers 90.80% of source facts, indicating that its graphs preserve most salient content; and in the graph condition here the supporting-fact F1 is 71.67, so the facts needed to justify an answer may frequently be present in the graph even when the final answer is wrong. Moreover, we provided as input to the LLMs all the passages and the whole graph in each respective experiment. Accessing the whole knowledge graph rather than a

Dataset	System	Ont.	P $\uparrow$	R $\uparrow$	F1 $\uparrow$	Conf. $\uparrow$	Sub.H. $\downarrow$	Rel.H. $\downarrow$	Obj.H. $\downarrow$
Wikidata–TekGen	Vicuna-13B [†]	on	0.38	0.34	0.35	0.83	0.17	0.17	0.17
	Alpaca-LoRA-13B [†]	on	0.32	0.26	0.27	0.87	0.18	0.13	0.17
	LMKG	off	0.10	0.08	0.08	0.11	0.01	0.89	0.03
	LMKG	on	0.33	0.31	0.31	0.97	0.01	0.03	0.02
DBpedia–WebNLG	Vicuna-13B [†]	on	0.34	0.27	0.30	0.93	0.12	0.07	0.28
	Alpaca-LoRA-13B [†]	on	0.32	0.23	0.25	0.91	0.16	0.09	0.38
	LMKG	off	0.25	0.14	0.17	0.37	0.00	0.63	0.02
	LMKG	on	0.43	0.36	0.39	1.00	0.00	0.00	0.02

Table 2: Text2KGBench published baselines and full LMKG runs. Values are in the 0–1 scale and are macro averages across ontologies. Conf. is ontology conformance; Sub.H., Rel.H., and Obj.H. are subject, relation, and object hallucination rates. Bold marks the better of our two settings where the paired difference is significant. [†] Reported from Table 2 of [Mihindukulasooriya et al. \(2023\)](#)—All variant.

System	MINE-1 $\uparrow$
OpenIE [†]	29.29
GraphRAG [†]	47.94
KGGen [†]	65.41
WIKONTIC (gpt4o) [†]	84
WIKONTIC (gpt4.1-mini) [†]	86
Ours, fold	84.87
Ours, enrich	90.80

Table 3: MINE-1 fact recovery. Score is the mean per-article fact-recovery percentage. [†] Reported.

Context	Ans. EM $\uparrow$	Ans. F1 $\uparrow$	Sup. F1 $\uparrow$
Passages	65.36	79.59	n/a
Graph	55.40	68.48	71.67

Table 4: Comparison on 2,500 samples from the HotpotQA distractor subset.

selected subgraph (as happens in MINE-1) may partially explain this result. We do not test this directly and leave a controlled comparison to future work.

5.4 Evidence as audit interface

The provenance fields give the refiner a local proposition to inspect, allow deterministic code to recover document offsets, and allow downstream outputs to refer back to source units. This supports auditability by allowing a human reviewer to trace each entity or triple back to the original text.

6 Conclusion

We introduce LMKG, a modular and ontology-aware KG extraction pipeline that allows for flexible extraction by exposing modular components. We presented and evaluated three strategies for handling documents of different lengths: a simple strat-

egy for sentences and two strategies for long documents. We tested our system in three different settings, two for evaluating KG quality and one for downstream usefulness. Our system reaches the highest reported score on the MINE-1 factuality recall benchmark, while demonstrating the importance of ontology awareness for LLMs whenever a target ontology is available. We observe a stark contrast between this strong fact recovery and the weaker downstream usability of the graphs produced by LMKG. This prompts us to question both the quality of the graphs and the LLM’s ability to leverage them in the downstream task (QA), both of which will be addressed in future work.

Limitations

The comparison between *fold* and *enrich* on MINE-1 does not isolate the factors contributing to their performance. Because MINE-1 measures fact recovery (recall) and does not penalize unsupported edges, *enrich*’s higher score reflects the additional coverage. A future ablation study would help isolate and quantify the sources of contribution and precision cost of each method.

Structure-only graph completion in *enrich* is especially vulnerable to plausible but unsupported edges. Canonicalization can merge distinct entities, and ontology constraints can discard valid facts that fall outside a schema. These errors matter in high-stakes domains; the resulting graphs should remain auditable and should not be treated as verified without human adjudication.

Although all selected benchmarks present the model with sentence-level or document-level extraction tasks, our system can also be used in a multi-document setting, which we plan to evaluate in future work.

LMKG operates on English text in the present study. Its evidence critic is a language-model self-check rather than an independent graph verifier. Although our evidence resolution mechanism partially mitigates this limitation, a stronger, language-independent external verification mechanism would be desirable.

All experiments use a single extraction and question-answering model. Our findings may therefore not transfer to other language models, and a systematic study of robustness across model families and sizes remains future work.

Acknowledgements

Thanks to DecisionBrain for providing computational resources to run the experiments.⁴

AI assistance. Generative AI tools were used to assist with language editing and improve the clarity of the manuscript. The authors reviewed and revised all generated suggestions and take full responsibility for the content of the paper.

References

- Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. [Leveraging linguistic structure for open domain information extraction](#). In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 344–354, Beijing, China. Association for Computational Linguistics.
- Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. [DBpedia: A nucleus for a web of open data](#). In *The Semantic Web*, volume 4825 of *Lecture Notes in Computer Science*, pages 722–735, Berlin, Heidelberg. Springer.
- 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, Bangkok, Thailand. Association for Computational Linguistics.
- Alla Chepurova, Aydar Bulatov, Mikhail Burtsev, and Yuri Kuratov. 2026. [Wikontic: Constructing wikidata-aligned, ontology-aware knowledge graphs with large language models](#). In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8304–8319, Rabat, Morocco. Association for Computational Linguistics.
- 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](#). *Preprint*, arXiv:2404.16130.
- Xiangkun Hu, Dongyu Ru, Lin Qiu, Qipeng Guo, Tianhang Zhang, Yang Xu, Yun Luo, Pengfei Liu, Yue Zhang, and Zheng Zhang. 2024. [Knowledge-centric hallucination detection](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 6953–6975, Miami, Florida, USA. Association for Computational Linguistics.
- Pere-Lluís Hugué 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, Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Chengyue Jiang, Yong Jiang, Weiqi Wu, Yuting Zheng, Pengjun Xie, and Kewei Tu. 2023. [COMBO: A complete benchmark for open KG canonicalization](#). In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 340–357, Dubrovnik, Croatia. Association for Computational Linguistics.
- 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.
- Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. 2024. [DSPy: Compiling declarative language model calls into state-of-the-art pipelines](#). In *The Twelfth International Conference on Learning Representations*.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. [Retrieval-augmented generation for knowledge-intensive NLP tasks](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.
- Nandana Mihindukulasooriya, Sanju Tiwari, Carlos F. Enguix, and Kusum Lata. 2023. [Text2KGBench: A benchmark for ontology-driven knowledge graph generation from text](#). In *The Semantic Web – ISWC 2023*, volume 14266 of *Lecture Notes in Computer Science*, pages 247–265, Cham. Springer.

⁴<https://decisionbrain.com>

Belinda Mo, Kyssen Yu, Joshua Kazdan, Joan Cabezas, Proud Mpala, Lisa Yu, Chris Cundy, Charilaos Kanatsoulis, and Sanmi Koyejo. 2025. [KGGen: Extracting knowledge graphs from plain text with language models](#). In *Advances in Neural Information Processing Systems*, volume 38, pages 33962–33985. Neural Information Processing Systems Foundation.

Nils Reimers and Iryna Gurevych. 2019. [Sentence-BERT: Sentence embeddings using siamese BERT-networks](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. [YAGO: A core of semantic knowledge](#). In *Proceedings of the 16th International Conference on World Wide Web, WWW '07*, pages 697–706, New York, NY, USA. Association for Computing Machinery.

Denny Vrandečić and Markus Krötzsch. 2014. [Wiki-data: A free collaborative knowledgebase](#). *Communications of the ACM*, 57(10):78–85.

David Wadden, Ulme Wennberg, Yi Luan, and Hananeh Hajishirzi. 2019. [Entity, relation, and event extraction with contextualized span representations](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5784–5789, Hong Kong, China. Association for Computational Linguistics.

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, Brussels, Belgium. Association for Computational Linguistics.

Hongbin Ye, Honghao Gui, Xin Xu, Xi Chen, Huajun Chen, and Ningyu Zhang. 2023. [Schema-adaptable knowledge graph construction](#). In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 6408–6431, Singapore. Association for Computational Linguistics.

Bowen Zhang and Harold Soh. 2024. [Extract, define, canonicalize: An LLM-based framework for knowledge graph construction](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9820–9836, Miami, Florida, USA. Association for Computational Linguistics.

A Model Prompts and Signatures

This appendix reports the instruction string and input–output signature of every model-facing com-

ponent used by the system and the reported evaluations. DSPy (Khattab et al., 2024) serializes runtime values and structured types around these instructions. Components marked *CoT* use ChainOfThought; those marked *direct* use Predict. Text, ontology values, graph contents, and test-specific demonstrations are runtime inputs rather than fixed prompt text.

A.1 Modules

Entity extraction (CoT).

Extract the named entities mentioned in the text. If a list of allowed types is given, assign each entity exactly one of those types; if ‘open’, assign a suitable concise type. For each entity, set ‘evidence’ to a short verbatim quote copied from the text that mentions it (its supporting span).

Inputs. text (*string*); entity_types (*string: comma-separated allowed types, or ‘open’*).

Output. entities (*list of entities, each carrying a verbatim evidence quote*).

Relation extraction (CoT).

Extract every explicit subject-relation-object proposition in the text. Use the given entities as anchors, but preserve the actual entity or value asserted as the object. Each triple must express one fact. Its relation must name the semantic role of its object, never a copula; split descriptions or values that express multiple roles. Do not add outside facts. If allowed relations are given, use only those labels and skip propositions that do not fit; if ‘open’, use concise relation labels. Give every triple a short verbatim evidence quote supporting the whole proposition.

Inputs. text (*string*); entities (*list of entity anchors containing names and types only*); relation_types (*string: allowed relations, or ‘open’*).

Output. triples (*list of triples, each carrying a verbatim evidence quote*).

Entity canonicalization (CoT).

Group only entity mentions that unambiguously refer to the same entity. Keep uncertain mentions separate and choose an existing name for each group.

Input. entities (*list of entities containing names and types only*).

Output. clusters (*list of entity clusters*).

Evidence refinement (CoT).

Verify each triple against the text, which is the source of truth. Remove unsupported triples and replace copula predicates or multi-role values with atomic propositions whose relations name

their objects' semantic roles. If allowed relations are supplied, retain and add only those exact semantic labels; skip a proposition that does not fit rather than inventing a related label. Add explicit propositions that are missing; objects may be stated values rather than named entities. Give every added triple a verbatim evidence quote supporting the whole proposition.

Inputs. text (string); triples (list of triples carrying evidence spans); relation_types (string: allowed relation labels, or 'open').

Outputs. unsupported (triples to remove); missing (triples to add, each with an evidence quote).

Structure-only graph completion (CoT).

You are given the entities of a knowledge graph and the relations already extracted between them. Propose additional relation triples that are missing but strongly implied by the existing structure—connecting entities that clearly should be linked given the relations already present. Use only the given entities as subjects and objects, and reuse existing relation labels where they fit. Do NOT restate triples already present. Return only the new triples.

Inputs. entities (graph nodes containing names and types only); triples (existing relations without evidence).

Output. missing (new triples connecting existing nodes).

A.2 Evaluation

HotpotQA answer generation (CoT).

Answer the question using ONLY the numbered facts provided.

The facts come from a knowledge graph extracted from the source document; they are all you have. If they do not contain the answer, say so rather than guessing from your own knowledge—an invented answer is worse than an admitted gap.

Answers may be an entity/span or exactly "yes" or "no". Return only that short answer. Never infer facts that are absent from the provided graph facts.

Inputs. question (string); facts (numbered subject | relation | object).

Outputs. used_facts (list of supporting fact numbers, or an empty list); answer (short name, phrase, number, or date, without explanation).