SAKGC: An Iterative RAG-Powered Framework for Stereo Knowledge Graph Construction

Anonymous ACL submission

Abstract

Research on LLM-based knowledge-graph (KG) automation is accelerating, while retrieval-augmented generation (RAG) is becoming the de-facto strategy for grounding large language models in external facts. Together these trends highlight a pressing demand for KGs that are not only domain-specialised but also endowed with multi-layer explanations that an LLM can traverse when reasoning.

We introduce **SAKGC**, an iterative two-phase framework that (i) extracts and organises large volumes of heterogeneous data into a compact horizontal KG and (ii) uses RAG to attach complexity-aware, hierarchical explanations to every non-trivial entity. Extensive experiments on three corpora of increasing scale show that SAKGC improves triple accuracy, reduces redundancy and internal-knowledge leakage, and boosts answer correctness and chain-of-thought clarity in downstream QA. Code and data are available at https://anonymous.4open.science/r/SAKGC-3E67.

1 Introduction

Knowledge graphs (KGs) are structured collections of factual triples, typically expressed as (*head entity*, *relation*, *tail entity*), that represent human knowledge in a machine-readable format. Recent research has increasingly explored the synergy between large language models (LLMs) and KGs (Yang et al., 2024) (Pan et al., 2024). On one hand, LLMs have been used to *automatically* construct KGs from raw text corpora; on the other, KGs have been integrated into *KG-based retrieval-augmented generation* (KG-RAG) pipelines to improve the factual accuracy and consistency of LLM outputs. These advances highlight significant potential: heterogeneous Web data can be transformed into structured, queryable KGs, while LLMsenhanced by subgraph matching and related reasoning algorithmscan accurately retrieve relevant facts to support complex queries. This promise is especially salient in the context of **domain-specific knowledge graphs**, where high-precision factual content is essential for expert-level question answering and decision support (Santos et al., 2022) (Zhang et al., 2022). As a result, the **central challenge** becomes the *efficient, domain-specialised, and fully automatic construction of KGs that are tightly aligned with the capabilities and reasoning patterns of LLMs*.

Despite promising progress, current LLM-centric approaches remain insufficient for real-world deployment. Domain-specific corpora are typically vast and repetitive, leading models to generate multiple triples that express the *same meaning* in slightly varied forms. This redundancy inflates both storage and retrieval costs without yielding meaningful performance gains. In addition, some methods rely exclusively on the reasoning capacity of LLMs while ignoring knowledge already internalized by the model, resulting in repeated regeneration of known facts (Cao, 2023). Most automatically constructed KGs also limit themselves to surface-level triple extraction from factual text, without deeper semantic structuring or abstractionthereby restricting downstream explainability. Combined with contextual noise in the input, these limitations degrade extraction quality and increase the risk of hallucinated outputs when grounding is weak or inconsistent (Zhang et al., 2023).

To address these challenges, we propose **Stereo Automatic Knowledge Graph Construction (SAKGC)**, a framework that constructs domain-specific knowledge graphs from raw text along two complementary dimensions. The *horizontal dimension* performs an iterative extraction process to mine high-quality factual triples from domain corpora. A **redundancy-merging module** consolidates semantically equivalent triples, while

an **internal-knowledge pruning module** removes facts already embedded in the LLM, thereby reducing storage cost and improving retrieval efficiency. The *vertical dimension* enriches each entity in the horizontal KG by generating hierarchical, complexity-aware explanations using retrieval-augmented generation (RAG), guided by a proposed *entity complexity* metric. We further design a reliable evaluation scheme to validate the effectiveness of SAKGC through extensive experiments across multiple settings.

In summary, this paper makes the following main contributions:

1. **SAKGC framework.** We introduce Stereo Automatic Knowledge Construction, a two-dimensional pipeline that couples horizontal factual extraction with vertical hierarchical explanation, aligning the graphs structure with the information-seeking behaviour of LLMs.

2. **Redundancy-aware compression.** A dedicated merging module detects and fuses semantically equivalent triples, and an internal-knowledge pruning component eliminates facts already memorised by the LLM, jointly reducing storage cost while preserving recall.

3. **Entity-complexitydriven explanation.** We formalise entity complexity and leverage it to trigger retrieval-augmented generation, yielding multi-level explanations that enhance the interpretability of KG-assisted reasoning.

4. **Evaluation protocol.** We survey existing evaluation practices in knowledge graph construction, propose a more comprehensive and domain-oriented evaluation scheme, and conduct extensive experiments based on this protocol to demonstrate the advantages of SAKGC.

2 Related Work

2.1 LLM-Based Automatic Knowledge Graph Construction

Recent studies show that large language models (LLMs) can drive *fully automated* KG construction by directly extracting, categorising, and linking entities from unstructured text. With a predefined target schemaor even an induced schema when none existsan LLM turns raw documents into triples, enabling nearreal-time graph updates with minimal human oversight (Zhang and Soh, 2024). This ability has already yielded practical gains in specialised domains: in healthcare and biomedicine, LLM pipelines distil entities and relations from electronic health records and scholarly articles, slashing annotation time while maintaining high recall (Xu et al., 2024; Arsenyan et al., 2023); in mental-health research, depression-related corpora are organised cheaply and rapidly for downstream analysis (Park et al., 2024).

Building on these extraction capabilities, end-to-end frameworks integrate knowledge extraction, refinement and updating into a unified pipeline. Auto-KGQA supplies KG fragments as context so that an LLM can convert natural-language queries into structured SPARQL, overcoming token-length limits (Avila et al., 2024). Semi-automated toolkits assist ontology expansion with limited expert input (Kommineni et al., 2024b,a), while KELDaR improves graph fidelity through competency-question generation and retrieval-based verification (Li et al., 2024). Complementary efforts tackle hallucination by cross-checking model outputs against trusted graphs (KGR) (Guan et al., 2024), align knowledge and rerank entities without fine-tuning (Chen et al., 2024), benchmark LLMs on RDF-centric formats (Frey et al., 2024), and adapt construction to streaming or temporal data via Text-to-KG and dynamic TKG pipelines (Ghanem and Cruz, 2024; Di Maio et al., 2024).

While LLMs have unlocked promising levels of automation, several challenges remain. *Hallucination* is foremost: models can invent facts when faced with noisy inputs, sparse evidence or ambiguous wording, thereby degrading KG reliability (Guan et al., 2024). Furthermore, most pipelines stop at surface-level triples and omit deeper semantic explanations, limiting the interpretability of downstream reasoning. Finally, empirical studies reveal that different LLMs exhibit highly variable performance across RDF parsing, SPARQL generation and temporal reasoning tasks, making robust model selection an open research question.

Based on these findings, our work targets hallucination and explainability by producing lean, hierarchically explained graphs that align more closely with LLM reasoning.

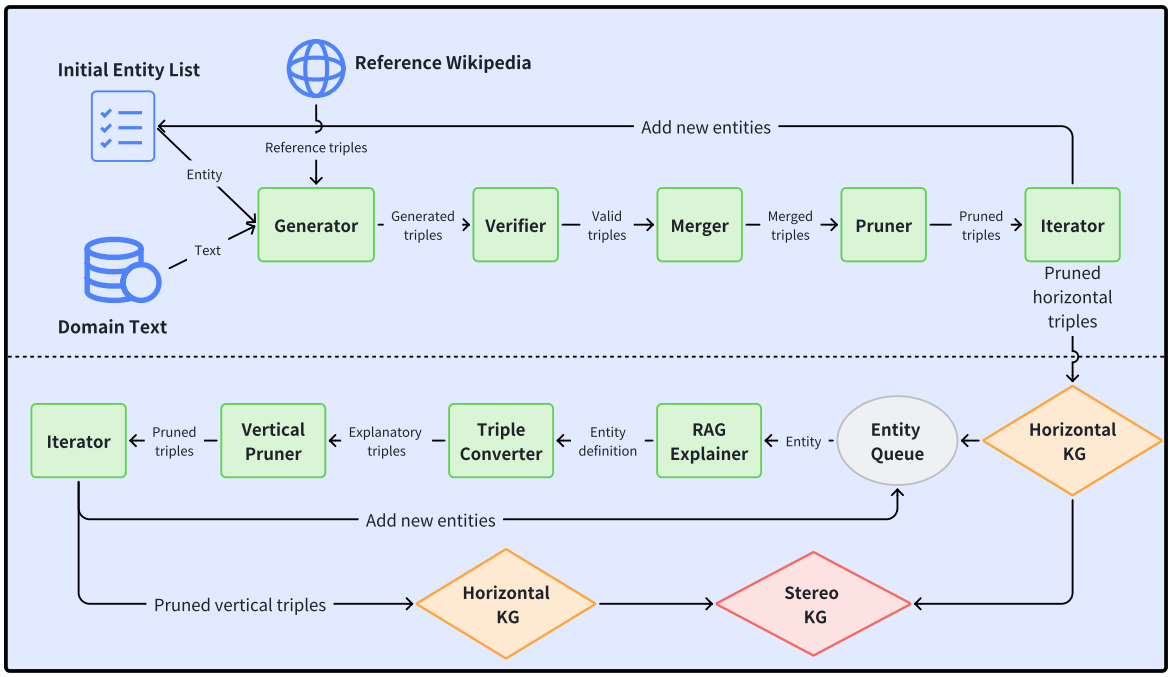

Figure 1: The freamwork of SAKGC. SAKGC consists of two components: horizontal knowledge graph expansion (upper part) and vertical knowledge graph expansion (lower part).

2.2 Retrieval-Augmented Generation (RAG)

Text-centric RAG. Retrieval-augmented generation combines neural language models with non-parametric memory so that the model can consult external documents at inference time, thereby improving factual accuracy and reducing parametric load (Li et al., 2022; Khandelwal et al., 2019a). Early instantiations such as REALM (Guu et al., 2020) and RAG (Lewis et al., 2020) jointly train a dense retriever and a generator by treating retrieved documents as latent variables. Subsequent work scales memory and architecture: kNN-LM replaces LSTMs with Transformers and indexes billions of tokens (Khandelwal et al., 2019b), while RETRO pushes the retrieval store to the trillion-token scale and feeds the retrieved passages directly into the decoder (Borgeaud et al., 2022). More recent systems query live search engines to obtain up-to-date evidence before generation (Shuster et al., 2022; Lazaridou et al., 2022).

Graph-based RAG. Traditional RAG treats the knowledge source as flat text, which under-utilises the rich relational structure present in many domains. GraphRAG addresses this gap by retrieving subgraphsnot passagesfrom a graph database and injecting their structured context into the language model. By coupling graph neural networks with LLMs, GraphRAG can exploit non-Euclidean topology, yielding superior accuracy on tasks such as node and edge classification (Zhu et al., 2024). Empirical studies report consistent gains when relational signals are provided alongside textual cues, especially for multi-hop or multi-entity questions.

KG-enhanced RAG. A further specialisation integrates explicit *knowledge graphs* (KGs) into the retrieval loop. Here, the model first pulls entityrelation triples via KG reasoning and then conditions generation on those triples. KGLM exemplifies the paradigm: it retrieves facts relevant to the current context and verbalises them as factual sentences, enabling the LLM to employ out-of-vocabulary entities and domain-specific relations (Logan IV et al., 2019). While LLMs augmented in this way achieve impressive domain precision, they remain vulnerable to hallucination when the KG is incomplete or ambiguous.

Therefore, we design SAKGC framework to supply hierarchically explained, redundancy-reduced KG evidence that mitigates hallucination and improves answer faithfulness.

3 Methodology

3.1 Task Formulation

Given a domain corpus $\mathcal{D} = \{d_i\}_{i=1}^{N}$ and an initial entity list $\mathcal{E}_0$ containing the core concepts of that domain, our goal is to automatically construct a *stereo* knowledge graph that couples surface facts with hierarchical explanations. Concretely,

$$\mathcal{G} = \big(\mathcal{G}_H, \mathcal{G}_V\big), \tag{1}$$

where the **horizontal graph** $\mathcal{G}_H$ stores triples extracted from $\mathcal{D}$ and the **vertical graph** $\mathcal{G}_V$ expands high-complexity entities in $\mathcal{G}_H$ into successively simpler explanatory triples. The construction proceeds iteratively: a horizontal loop grows $\mathcal{G}_H$ until no new entities remain, and a vertical loop then refines selected entities according to a complexity threshold τ. The process terminates once both loops converge, yielding a compact, hierarchically explained stereo KG.

3.2 SAKGC Framework

We propose a novel system called **Stereo Automatic Knowledge Enrichment** (SAKGC) to address this task. The framework of SAKGC is illustrated in Figure 1.

Horizontal expansion. Let the working entity queue be $\mathcal{E}$ (initially $\mathcal{E}_0$). For each unvisited entity $e \in \mathcal{E}$, the following LLM-driven pipeline is executed:

1. **Generator** $(\mathcal{D}, e, b) \rightarrow \mathcal{T}_{\text{gen}}$: To extract a comprehensive set of triples headed by entity e, we design a generator module that leverages large language models (LLMs) via few-shot prompting.

   We first retrieve a small set of reference triples involving e from Wikipedia. These examples are then formatted as demonstrations to guide the LLM in extracting additional triples from a given corpus $\mathcal{D}$. Specifically, the corpus is divided into sentence batches of size b to ensure compatibility with the LLMs context window. Each batch is paired with the prompt, instructing the model to identify all relation triples where e appears as the head entity.

   This process results in a set of generated triples $\mathcal{T}_{\text{gen}}$, which capture diverse relational knowledge about e grounded in textual evidence.

2. **Validator** $\mathcal{T}_{\text{gen}} \rightarrow \mathcal{T}_{\text{val}}$: To ensure the quality and factuality of the generated triples, we introduce a validator module that filters out invalid or illogical entries from $\mathcal{T}_{\text{gen}}$.

   Each triple is evaluated by prompting an LLM with a factuality judgment task framed from an engineer's perspective. The prompt explicitly asks the model to label a triple as "incorrect" only if it is certain the information is factually wrong. If the triple is factually accurate, or if there is insufficient context to make a definitive judgment, the model returns "correct." This conservative strategy minimizes false negatives while maintaining precision, resulting in a validated triple set $\mathcal{T}_{\text{val}}$.

3. **Merger** $\mathcal{T}_{\text{val}} \rightarrow \mathcal{T}_{\text{mrg}}$: To reduce redundancy and optimize downstream usage, we design a similarity-based merger that identifies semantically overlapping triples in $\mathcal{T}_{\text{val}}$. A pairwise similarity function is applied to assess semantic closeness between triples, and highly similar triples are merged into unified representations.

   This process helps preserve the core informational content while reducing storage cost and retrieval overhead in downstream tasks. Details of the merging algorithm are provided in Section 3.3, and its impact on retrieval-based knowledge graph augmentation is evaluated in Section 4.4.

4. **Pruner** $\mathcal{T}_{\text{mrg}} \rightarrow \mathcal{T}_{\text{prn}}$: To avoid redundant knowledge and maximize the utility of external triples, we implement a pruning module that identifies and removes triples likely already memorized by the LLM.

   A relatively strict protocol is used to assess whether a given triple reflects internal knowledge encoded within the LLM. This ensures that the retained triples contribute novel or complementary information during retrieval.

   The implementation details of this module are described in Section 3.4, and its effect on retrieval performance is empirically evaluated in Section 4.5.

5. **Iterator** $\mathcal{T}_{\text{prn}} \rightarrow \Delta\mathcal{E}$: To enable iterative expansion of the knowledge graph, we design an iterator module that determines which tail

entities in $\mathcal{T}_{\text{prn}}$ should be promoted as new head entities for the next round.

This decision is made by querying the LLM through an API, asking whether a given tail entity is sufficiently central or informative to justify further exploration. The selected entities $\Delta\mathcal{E}$ serve as the seeds for the subsequent iteration.

The final set $\mathcal{T}_{\text{prn}}$ is added to $\mathcal{G}_H$. The current entity e is marked as *visited*, and the new entities $\Delta\mathcal{E}$ are merged into the queue. This loop repeats until no new entities are generated or the maximum depth k_1 is reached.

Vertical expansion. Based on the horizontal graphs entity set $\mathcal{V}_H$, SAKGC constructs a hierarchical explanation graph via the mapping:

$$f_{\text{vert}} : (\mathcal{V}_H, \tau, k_2) \longrightarrow \mathcal{G}_V. \tag{2}$$

Entities with complexity $c(e) > \tau$ are queued for vertical expansion. For each such entity:

1. **RAG Explainer** $e \rightarrow d(e)$: Retrieves a concise definition text for e using retrieval-augmented generation.

2. **Converter** $(e, d(e)) \rightarrow \mathcal{S}_{\text{conv}}$: Converts the definition into a set of explanatory triples with e as the head in every triple.

3. **Vertical Pruner** $\mathcal{S}_{\text{conv}} \rightarrow \mathcal{S}_{\text{prn}}$: Removes any triple whose tail entity has complexity higher than e.

The filtered triples $\mathcal{S}_{\text{prn}}$ are added to $\mathcal{G}_V$, and eligible tail entities are added to the vertical queue if their complexity satisfies $\tau < c(t) < c(e)$. The process repeats until the queue is empty or a recursion depth k_2 is reached. Formal definitions of complexity, the RAG retrieval process, and triple conversion rules are provided in Section 3.5.

3.3 Merger Module

Given the validated triple multiset $\mathcal{T}_{\text{val}}(e) = \{t_k = (e, r_k, o_k)\}_{k=1}^{n}$, we first define the *similarity* between two triples as

$$s(t_i, t_j) \in [0, 1], \tag{3}$$

where larger values indicate stronger synonymy. This score is computed by prompting the backbone LLM with the pair $\langle t_i, t_j \rangle$ and reading its real-valued output. Evaluating all pairs produces an $n \times n$ symmetric matrix $\mathbf{S} = [S_{ij}]$ with $S_{ij} = s(t_i, t_j)$.

A threshold τ binarises the matrix into $A_{ij} = \mathbf{I}[S_{ij} \geq \tau]$. Starting from the full index set $R = \{1, \ldots, n\}$, we repeatedly remove the node with the largest degree $d(i) = \sum_{j \neq i} A_{ij}$ (ties broken by input order) until R contains no edges. The surviving indices $K \subseteq R$ define the deduplicated output

$$\mathcal{T}_{\text{mrg}}(e) = \{\, t_k \mid k \in K \}. \tag{4}$$

To alleviate the $O(n^2)$ oracle cost, triples are processed in mini-batches of size B; each batch needs B^2 similarity calls and there are n/B batches in total, giving an overall complexity of $O(nB)$. Because batches run in parallel threads, the wall-time is further reduced in practice.

3.4 Internal-Knowledge Pruning

In a triple (h, r, t) the relation r can be viewed as a logical condition that links two concepts h and t. We call r a *sufficient* condition for t with respect to h if the statement $(h, r) \Rightarrow t$ holds for the LLM, and a *necessary* condition if t together with r implies h, i.e. $(r, t) \Rightarrow h$. When both implications hold, r is a *necessary and sufficient* condition, meaning the three elements are mutually derivable. Under this theory we adopt the hypothesis that an LLM already possesses a fact whenever it can (i) recover the correct relation from the head-tail pair,

$$\hat{r} = \text{pred}(h, t), \qquad \hat{r} \approx r, \tag{5}$$

and (ii) derive at least one entity from the relation plus the other entity, $\hat{t} = \text{pred}(h, r)$ or $\hat{h} = \text{pred}(r, t)$, with the predicted element semantically equivalent to its ground-truth counterpart. Triples satisfying this two-step test form the internal set $\mathcal{T}_{\text{int}}(e)$ and are removed; the remainder $\mathcal{T}_{\text{prn}}(e)$ is retained. Prompt batching of size β cuts the number of LLM calls from $3|\mathcal{T}_{\text{mrg}}|$ to $3|\mathcal{T}_{\text{mrg}}|/\beta$, and parallel threads accelerate execution.

Consider the triple *(Obama, nationality, USA)*. If the LLM truly knows this fact, it correctly predicts *nationality* from the pair (Obama, USA); the relation is therefore at least *sufficient*. Furthermore, the model can use (Obama, nationality) to infer *USA*, satisfying the at least one entity clause, so the triple is discarded as internal. Now suppose the model lacks information about Obama. It still predicts the relation from (Obama, USA) but

fails to obtain *USA* from (Obama, nationality); the sufficient condition holds but the entity inference fails, so the triple is kept. Conversely, even if the model is familiar with both Obama and USA, it cannot invert (USA, nationality) to recover Obama because *nationality* is not a sufficient condition in that direction; only one entity inference succeeds, and the triple remains novel. Thus the criterion removes a substantial portion of already encoded knowledge while preserving facts that extend the LLMs capabilities.

3.5 Vertical Expansion of the Stereo KG

Input and fine-grained complexity. The vertical module consumes the horizontal entity set $\mathcal{V}_H$ and outputs a hierarchical explanation graph $\mathcal{G}_V$. For every entity e we obtain four ratings from one LLM prompt and combine them as

$$C(e) = D(e) + L(e) + I(e) + C_s(e) \quad (6)$$

Here $D(e)$ counts how many knowledge domains (biology, law, history, technology, art) are involved; $L(e)$ maps to primary, secondary, bachelor, master, or doctoral knowledge; $I(e)$ is public comprehension difficulty (very easy very hard); $C_s(e) \in \{0, 1, 2\}$ marks West-centric, neutral, or East-centric background so that culturally distant concepts are penalised when the target audience is known. Entities with $C(e) > C_{\text{thresh}}$ enter the expansion queue.

RAG-driven recursive expansion. Many graph entities are rare terms that lack a canonical encyclopedia entry. We therefore retrieve the k nearest neighbours $S_k(e)$ from an external corpus and assemble their definitions and triples into a context prompt C_e. A GPT-class generator uses C_e to synthesise a fresh definition $d(e)$ for e and then converts that definition into explanatory triples

$$\{(e, r_i, t_i)\}_{i=1}^{m} \quad (7)$$

each headed by the original entity $e$in effect, a set of tail entities that explain e. A triple is kept only when the tail complexity satisfies

$$C(t_i) < C(e) \quad (8)$$

so that more complex entities are never used to clarify a simpler one. Accepted tails with $C(t_i) > C_{\text{thresh}}$ are queued for the next recursion; the loop ends when the queue is empty or a depth limit k_2 is reached.

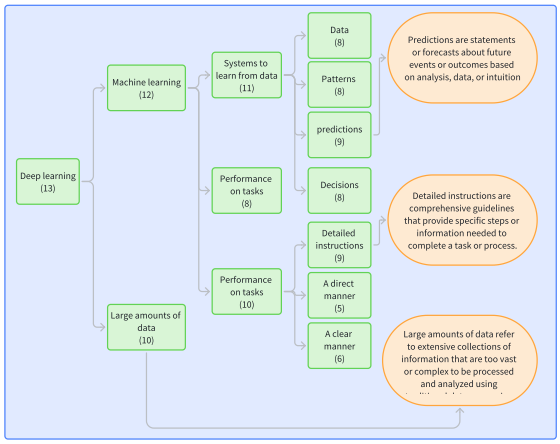

Figure 2: An example of recursive vertical expansion for a given entity (*deep learning*). Green boxes indicate the newly expanded entities, while orange boxes represent definitions for the final-layer entities. The number inside each green box denotes the entity's complexity. A definition is generated only if complexity exceeds 8. In this case, the expansion reaches 3 layers.

Illustrative example. For the entity *Deep learning* ($C = 13$), our RAG pipeline first retrieves an authoritative textual definition, then initiates a complexity-aware depth-first expansion (Fig. 2).

In layer 1, the system discovers two tail entities: *machine learning* ($C = 12$) and *large amounts of data* ($C = 10$). The former is further expanded, while the latter haltsits definition is parsed, but all resulting triples point to entities of comparable or higher complexity, violating the simplification constraint. Layer 2 expands *machine learning* into three entities: *systems to learn from data* ($C = 11$), *performance on tasks* ($C = 8$), and *explicit programming* ($C = 10$). The second entity is too simple to justify further definition, but the other two are expanded into new triples in layer 3. Layer 3 yields four leaf entities: *data* ($C = 8$), *patterns* ($C = 8$), *predictions* ($C = 9$), and *decisions* ($C = 8$). Only *predictions* and *decisions* exceed the complexity threshold and are assigned definitions. The rest are considered sufficiently simple to be handled by the LLMs background knowledge.

As the recursion depth is capped at 3, the expansion concludes at this layer. The entire process follows the rule $C(t) < C(h) - 1$: each concept is grounded through strictly simpler ones, ensuring meaningful abstraction, bounded graph size, and efficient retrieval. This structured explanation path also serves as an explicit chain-of-thought for the

Method	TC	TR	Prec.	Rec.	F1
Pive	147	0.608	0.3749	**0.9101**	0.4819
KG-LLM	223	0.611	0.2287	0.7929	0.3227
CoKGC	97	0.740	**0.5165**	0.8435	**0.5785**
SAKGC	**400**	**0.805**	0.1376	0.8876	0.2382

Table 1: Evaluation on Genwiki-5K. TC: Triple Count; TR: Triple Relevance. Rec.: Recall.

Method	TC	TR	Prec.	Rec.	F1
Pive	18,562	0.577	0.3749	**0.9101**	0.4819
KG-LLM	12,060	0.742	0.2287	0.7929	0.3227
CoKGC	12,691	0.584	**0.4704**	0.8777	**0.5559**
SAKGC	**41,377**	**0.956**	-	-	-

Table 2: Evaluation on Genwiki-500K. TC: Triple Count; TR: Triple Relevance. Rec.: Recall.

language model.

4 Experiments

To evaluate the effectiveness of SAKGC, we design a two-stage experimental setup. In the first stage, we extend a standard suite of text-based knowledge graph construction benchmarks to assess the quality of horizontally expanded graphs across datasets of increasing size. We also design separate experiments to independently verify the utility of the redundancy-merging and internal-knowledge pruning modules. In the second stage, we incorporate the vertically structured graph into a QA pipeline, prompting the language model to generate both answers and their corresponding reasoning traces. We evaluate the quality of the generated answers and reasoning chains to demonstrate the benefit of complexity-driven recursion. Results are presented in Sections 4.1–4.5. All evaluation metrics, baseline descriptions, and implementation details are provided in Appendix A.

4.1 Evaluation on Standard Benchmark

We evaluate the quality of horizontally expanded knowledge graphs on the Genwiki dataset at two corpus scales: 5K and 500K. For each setting, we report all proposed metrics except Triple Similarity (TS). The reason TS is excluded is that, under small-scale data settings, the number of generated triples is relatively low, making it unlikely to observe redundant triples with the same head entity.

As shown in Tables 1 and 2, our method produces substantially more triples than all baselines and achieves the highest relevance (TR), reflecting strong alignment with the source text. Although our system covers most ground-truth triples, its precision appears lower because the metric's denominator grows with the total number of generated triples. We exclude G-BERTScore on the 500K set, as it operates at the sentence level, while our model generates triples at the document level-making sentence-wise alignment prohibitively expensive at scale.

Method	TC	TR	TS
Pive	277,410	0.866	0.0004
KG-LLM	34,194	0.841	-
CoKGC	101,523	0.728	0.0004
Ours	**854,874**	**0.998**	**0.0002**

Table 3: Evaluation on the real-world dataset (no gold-standard available). TS: Triple Similarity.

4.2 Evaluation on Real-World Corpus

We further evaluate the quality of horizontally expanded graphs on a real-world dataset combining two heterogeneous corpora related to the automotive domain. Since no gold-standard triples are available, we exclude G-BERTScore and focus on the remaining three metrics: Triple Count (TC), Triple Relevance (TR), and Triple Similarity (TS). Results are shown in Table 3.

Our method produces the highest number of triples with the best relevance score and the lowest semantic redundancy (TS). The low TS score indicates that our deduplication module effectively eliminates near-duplicate triples, improving the quality of the resulting graph. KG-LLM often fails to generate triples in this setting, resulting in a very low overall triple count. Due to the small output size, we omit similarity evaluation for this method.

4.3 Effectiveness of Redundancy Reduction

To assess whether merging synonymous triples affects answer quality, we conduct a controlled experiment. From a physics corpus, we sample 50 triples and generate 10 semantically equivalent variants for each. For each group, we formulate two factual questionsone inferring the tail given the head and relation, the other inferring the head given the tail and relation. These questions are then posed under three prompting conditions: (1) no triples, (2) 10 unmerged triples, and (3) 1 merged triple. Accuracy is measured across eight language models.

As shown in Table 4, most models perform poorly when no triples are provided, but achieve over 95% accuracy with either unmerged or merged triples. This indicates that merging redun-

Model	Count	No Trip.	Unm.	Merged
deepseek-chat	100	0.08	0.98	0.93
glm-4	100	0.02	0.93	0.82
gpt-3.5-turbo	100	0.04	0.66	0.84
gpt-4	100	0.05	0.98	0.97
gpt-4o	100	0.10	0.97	0.97
gpt-4o-mini	100	0.07	0.98	0.98
gpt-4-turbo	100	0.07	0.97	0.96
QwenQwQ-32B	100	0.11	0.96	0.96

Table 4: Accuracy of different models with and without triple context.

Model	Count	No Trip,	Unm.	Merged
deepseek-chat	100	79.72	237.51	102.45
glm-4	100	77.09	236.37	99.02
gpt-3.5-turbo	100	67.23	230.01	89.97
gpt-4	100	67.23	230.01	89.97
gpt-4o	100	67.23	230.01	89.97
gpt-4o-mini	100	67.23	230.01	89.97
gpt-4-turbo	100	67.23	230.01	89.97
QwenQwQ-32B	100	73.67	233.31	95.23

Table 5: Avg. token usage per setting.

dant triples preserves essential semantics while reducing repetition.

We also compare token consumption across settings. As shown in Table 5, merged triples greatly reduce token usage compared to unmerged ones, lowering inference cost with minimal impact. Minor differences across models stem from varying tokenization schemes.

4.4 Pruning Internal Knowledge

To test whether factual triples already known to the LLM can be pruned, we construct 200 yes/no questions of the form *Is city A located in country B?*, sampled from Wikipedia with a 50% true/false split.

We evaluate GPT-3.5-turbo under two settings: (1) the question alone, and (2) the question with a supporting triple `(city, location, country)`. As shown in Table 6, both settings yield nearly identical accuracy, suggesting that including facts already internalized by the model provides little added value. This supports pruning as a way to reduce token usage without harming answer quality.

4.5 Evaluating Vertical Expansion

We evaluate the impact of hierarchical concept expansion through a QA task in the AI domain, which contains rich and diverse entities spanning multiple complexity levels. We collect 200 AI-related entities and generate one reasoning-based question per entity, each paired with a reference answer. Three knowledge formats are compared: **zero_shot** (entity only), **is_a** (entity + one-sentence definition), and **graph** (entity + definition + hierarchical triples).

Setting	Triple Included?	Accuracy
Direct Question	No	98.5%
With Triple	Yes	99.0%

Table 6: Accuracy of GPT-4o on location classification with and without explicit supporting triples.

Method	Accuracy	Rel. Ent.	Steps
zero_shot	0.79 (158/200)	3.36	2.07
is_a	0.80 (160/200)	3.27	2.06
graph	**0.81 (163/200)**	**3.65**	**2.29**

Table 7: Performance of different knowledge formats. Rel. Ent.: Relevant Entities. Steps: Reasoning Steps.

We use GPT-3.5-Turbo to generate both the questions and reference answers. For answer generation, we prompt THUDM/GLM-Z1-32B-0414, a weaker model chosen to better reveal the utility of external knowledgestronger models might answer correctly without any support, masking the effect of the knowledge input. In addition to answers, the model outputs reasoning traces in triple format.

We then evaluate: (a) **Accuracy** whether the answer matches the reference (strict for missing key content, tolerant for additions), (b) **Relevant Entities** number of distinct entities used in reasoning, (c) **Reasoning Steps** length of the reasoning chain. As shown in Table 7, accuracy improves with richer knowledge formats, with our graph-based method performing best. It also yields broader entity coverage and deeper reasoning, indicating that structured explanations help the model integrate more relevant concepts. Additionally, our method produces clearer, more traceable reasoning paths, enhancing interpretability for real-world use.

5 Conclusion

We introduced **SAKGC**, an iterative framework that constructs stereo knowledge graphs from raw domain text. It combines horizontal triple extraction with redundancy control, and vertical expansion via RAG to generate complexity-aware, hierarchical explanations. Experiments show that SAKGC yields significantly more relevant triples with lower redundancy, and improves answer accuracy and reasoning coherence in QA tasks. Our results suggest that SAKGC provides a scalable and interpretable solution for domain-specific KG construction.

Limitations

This study demonstrates how a large language model, coupled with retrieval-augmented generation, can construct a stereo KG while refusing redundant or already-internal facts. Although promising, the framework is not yet ready for seamless industrial deployment and several limitations remain.

Construction efficiency and computational cost. Each entity traverses four horizontal and up to two vertical modules, every step invoking the LLM. Even with parallel batches, the generator must scan the full document set, yielding $O(n^2)$ worst-case complexity. On very large corpora the resulting latency and API cost may be prohibitive; smarter indexing and caching are required.

Scope of the pruning modules. Redundancy merging and internal-knowledge pruning are tuned for the RAG-QA scenario. When used for direct KG-QA or with a different backbone model, they may delete helpful triples or miss memorised facts. An adaptive pruning strategy is needed for broader use.

Metric ambiguity. Redundancy, entity complexity and internal knowledge are fuzzy notions. Our operational definitions work in practice but do not cover all edge cases; sharper formalisations and learning-based estimators could improve reliability.

Scalability. Experiments were limited to a 15-million-sentence corpus. Performance at web scalebillions of sentencesremains unknown and may expose new bottlenecks in storage, retrieval or recursion depth, calling for distributed implementations and further evaluation.

References

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853.

Galen Andrew and Jianfeng Gao. 2007. Scalable training of L1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning*, pages 33–40.

Vahan Arsenyan, Spartak Bughdaryan, and 1 others. 2023. Large language models for biomedical knowledge graph construction: Information extraction from emr notes. In *Workshop on BNLP*.

Caio Viktor S Avila, Marco A Casanova, and 1 others. 2024. A framework for question answering on knowledge graphs using large language models. In *ICSC*.

Zhen Bi, Jing Chen, Yinuo Jiang, Feiyu Xiong, Wei Guo, Huajun Chen, and Ningyu Zhang. 2024. Codekgc: Code language model for generative knowledge graph construction. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 23(3):1–16.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, and 1 others. 2022. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR.

Lang Cao. 2023. Learn to refuse: Making large language models more controllable and reliable through knowledge scope limitation and refusal mechanism. *arXiv preprint arXiv:2311.01041*.

Salvatore Carta, Alessandro Giuliani, Marco Manolo Manca, Leonardo Piano, Alessandro Sebastian Podda, Livio Pompianu, and Sandro Gabriele Tiddia. 2024. A zero-shot strategy for knowledge graph engineering using gpt-3.5. *Procedia Computer Science*, 246:2235–2243.

Zhongwu Chen, Long Bai, and 1 others. 2024. A new pipeline for knowledge graph reasoning enhanced by large language models without fine-tuning. In *EMNLP*.

Christian Di Maio, Andrea Zugarini, and 1 others. 2024. Tomorrow brings greater knowledge: Large language models join dynamic temporal knowledge graphs. In *CoLLAs*.

Johannes Frey, Lars-Peter Meyer, and 1 others. 2024. Assessing the evolution of llm capabilities for knowledge graph engineering in 2023. In *ESWC*.

Hussam Ghanem and Christophe Cruz. 2024. Fine-tuning vs. prompting: Evaluating the knowledge graph construction with llms. In *ESWC*.

Xinyan Guan, Yanjiang Liu, and 1 others. 2024. Mitigating large language model hallucinations via autonomous knowledge graph-based retrofitting. In *AAAI*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.

Jiuzhou Han, Nigel Collier, Wray Buntine, and Ehsan Shareghi. 2023. Pive: Prompting with iterative verification improving graph-based generative capability of llms. *arXiv preprint arXiv:2305.12392*.

Zhijing Jin, Qipeng Guo, Xipeng Qiu, and Zheng Zhang. 2020. Genwiki: A dataset of 1.3 million content-sharing text and graphs for unsupervised graph-to-text generation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2398–2409.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2019a. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172*.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2019b. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172*.

Vamsi Krishna Kommineni, Birgitta König-Ries, and 1 others. 2024a. From human experts to machines: An llm supported approach to ontology and knowledge graph construction. *arXiv preprint arXiv:2403.08345*.

Vamsi Krishna Kommineni, Birgitta König-Ries, and 1 others. 2024b. Towards the automation of knowledge graph construction using large language models. In *International Workshop on NLP4KGC*.

Angeliki Lazaridou, Elena Gribovskaya, Wojciech Stokowiec, and Nikolai Grigorev. 2022. Internet-augmented language models through few-shot prompting for open-domain question answering. *arXiv preprint arXiv:2203.05115*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Huayang Li, Yixuan Su, Deng Cai, Yan Wang, and Lemao Liu. 2022. A survey on retrieval-augmented text generation. *arXiv preprint arXiv:2202.01110*.

Yading Li, Dandan Song, and 1 others. 2024. A framework of knowledge graph-enhanced large language model based on question decomposition and atomic retrieval. In *EMNLP*.

Robert L Logan IV, Nelson F Liu, Matthew E Peters, Matt Gardner, and Sameer Singh. 2019. Barack's wife hillary: Using knowledge-graphs for fact-aware language modeling. *arXiv preprint arXiv:1906.07241*.

Shirui Pan, Linhao Luo, and 1 others. 2024. Unifying large language models and knowledge graphs: A roadmap. *TKDE*.

Chaelim Park, Hayoung Lee, and 1 others. 2024. Leveraging medical knowledge graphs and large language models for enhanced mental disorder information extraction. *Future Internet*.

Mohammad Sadegh Rasooli and Joel R. Tetreault. 2015. Yara parser: A fast and accurate dependency parser. *Computing Research Repository*, arXiv:1503.06733. Version 2.

Swarnadeep Saha, Prateek Yadav, Lisa Bauer, and Mohit Bansal. 2021. Explagraphs: An explanation graph generation task for structured commonsense reasoning. *arXiv preprint arXiv:2104.07644*.

Alberto Santos, Ana R Colaço, Annelaura B Nielsen, Lili Niu, Maximilian Strauss, Philipp E Geyer, Fabian Coscia, Nicolai J Wewer Albrechtsen, Filip Mundt, Lars Juhl Jensen, and 1 others. 2022. A knowledge graph to interpret clinical proteomics data. *Nature biotechnology*, 40(5):692–702.

Kurt Shuster, Mojtaba Komeili, Leonard Adolphs, Stephen Roller, Arthur Szlam, and Jason Weston. 2022. Language models that seek for knowledge: Modular search & generation for dialogue and prompt completion. *arXiv preprint arXiv:2203.13224*.

Tianhan Xu, Yixun Gu, and 1 others. 2024. Knowledge graph construction for heart failure using large language models with prompt engineering. *Frontiers in Computational Neuroscience*.

Linhao Yang, Hui Chen, and 1 others. 2024. Give us the facts: Enhancing large language models with knowledge graphs for fact-aware language modeling. *TKDE*.

Bowen Zhang and Harold Soh. 2024. Extract, define, canonicalize: An llm-based framework for knowledge graph construction. In *EMNLP*.

Ningyu Zhang, Zhen Bi, Xiaozhuan Liang, Siyuan Cheng, Haosen Hong, Shumin Deng, Jiazhang Lian, Qiang Zhang, and Huajun Chen. 2022. Ontoprotein: Protein pretraining with gene ontology embedding. *arXiv preprint arXiv:2201.11147*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, and 1 others. 2023. Siren's song in the ai ocean: a survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219*.

Yun Zhu, Yaoke Wang, Haizhou Shi, and Siliang Tang. 2024. Efficient tuning and inference for large language models on textual graphs. *arXiv preprint arXiv:2401.15569*.

A Experimental Setup

Datasets. We evaluate SAKGC on three text-only corpora of increasing size: **5K**, **500K**, and

15M tokens. The 5K and 500K sets are constructed from Genwiki (Jin et al., 2020), a benchmark dataset for text-based knowledge graph construction, which provides sentence-level gold-standard triples. To focus on a specific domain, we extract only physics-related content and consolidate it into the final corpus. The 15M dataset serves as a realistic open-domain scenario, comprising two equal parts: automotive-related content from a technical e-book and discussions scraped from Reddit on car topics.

Evaluation Metrics. While prior work commonly adopts G-BERTScore for evaluating triple prediction quality against reference data, it does not capture several important aspects of the text-to-triple generation process. To address this, we extend the evaluation protocol with three complementary metricsTriple Count (TC), Triple Relevance (TR), and Triple Similarity (TS)that offer a more holistic assessment of system behavior.

(1) Triple Count (TC): Measures the total number of generated triples, reflecting the model's ability to extract structured facts.

(2) Triple Relevance (TR): Quantifies how well the generated triples align with the input text. For each triple, we check whether the head, relation, and tail strings appear in the source sentence via exact match. TR is computed as the proportion of matched elements across all triples.

(3) Triple Similarity (TS): Evaluates semantic redundancy among triples sharing the same head entity. We randomly sample 10,000 distinct triple pairs and query an LLM to determine whether they are semantically synonymous. TS is the percentage of pairs classified as synonymous.

(4) G-BERTScore (G-BS): We adopt G-BERTScore (Saha et al., 2021), an extension of BERTScore (Zhang et al., 2019), to compare predicted triples with gold-standard references. Triples are verbalized into natural language sentences, and alignment is computed via semantic similarity. We report precision, recall, and F1:

$$\text{Precision} = \frac{\sum_{(p,g)\in\text{Alignment}} \text{BERTScore}(p,g)}{|\text{Predicted Triples}|}, \tag{9}$$

$$\text{Recall} = \frac{\sum_{(p,g)\in\text{Alignment}} \text{BERTScore}(p,g)}{|\text{Ground-Truth Triples}|}, \tag{10}$$

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{11}$$

Here, $p \in$ Predicted Triples and $g \in$ Ground-Truth Triples. The alignment set consists of one-to-one mappings between predicted and reference triples, based on their BERTScore similarity.

Baselines. We compare SAKGC against three recent and representative triple extraction methods:

Pive (Han et al., 2023) introduces an auxiliary module that detects missing elements in initially generated triples. If a triple is found incomplete, the model iteratively regenerates missing parts, improving triple completeness.

KG-LLM-Prompting (Carta et al., 2024) constructs a structured pipeline using LLMs to extract entities and relations sequentially. It uses prompt chaining and task decomposition to facilitate triple generation.

CodeKGC (Bi et al., 2024) reformulates triple extraction as a code generation task. It prompts the LLM using code-style instructions rather than natural language, leveraging domain-specific few-shot examples to improve generation accuracy.

Implementation Details. For horizontal expansion, we set the number of iterations $k_1 = 3$, with a batch size of $b = 4$K tokens per iteration. The similarity threshold for merging semantically redundant triples is fixed at 0.7. For vertical expansion, we also use a maximum recursion depth of $k_2 = 3$, and only expand entities with complexity scores greater than 8. Unless otherwise specified, all stages of our method rely on the DeepSeek language model as the backend for definition generation, triple extraction, and reasoning.