Incremental LLM-Based Pipeline for Scalable Knowledge Graph Construction

Abstract

Constructing knowledge graphs (KGs) from large document collections is a time-consuming process that traditionally requires substantial input from domain experts. Key challenges include handling contradictory information across sources, detecting topic boundaries when documents span multiple domains, and scaling the process to large collections without proportionally increasing expert effort. This paper presents an automated pipeline that addresses these challenges by using large language models (LLMs) to drive each stage of KG construction. The system processes documents incrementally, performing domain detection, conflict identification, knowledge extraction, graph merging, and error correction in a loop. Extracted knowledge is represented in Rainbird's RBLang and validated against the Rainbird API, with a self-correction loop that resolves errors without human intervention. The incremental design keeps LLM context manageable and produces domain-specific graphs rather than a single undifferentiated KG. The system demonstrates that high-quality, validated KGs can be generated from substantial document collections in significantly less time than manual approaches, while acknowledging that LLM-based generation is non-deterministic and that human review remains important for production use.

1 Introduction

Creating knowledge graphs (KGs) from large collections of documents is a labour-intensive task that requires significant input from expert knowledge engineers. These specialists must first become familiar with the domain in question and identify how concepts interact across multiple sources. This process is further complicated by the inherent imperfections of real-world document collections: sources may contain repeated concepts expressed using different terminology, overlapping information, or even direct contradictions where different documents describe the same entity or relationship in mutually incompatible ways. A single document may span several loosely related topics that belong to entirely different domains, making it difficult to decide how the resulting KG should be partitioned and structured.

KGs have long been recognised as a fundamental structure for organising and reasoning over complex information [Hogan *et al.*, 2021; Ji *et al.*, 2021]. However, their construction remains costly and reliant on manual effort [Paulheim, 2016]. These challenges become particularly acute as the scale of document collections grows. Manual approaches do not scale well, and the cost of expert involvement grows linearly with the number of documents to be processed. Therefore, there is a clear need for automated approaches capable of handling the full complexity of real-world document collections: detecting the domains to which content belongs, identifying conflicts between sources, and structuring the extracted knowledge appropriately for each domain rather than merging everything into a single undifferentiated graph.

This paper presents a pipeline that addresses these challenges by using large language models (LLMs) to automate the end-to-end process of KG generation. The system processes each document in turn, extracting entities, relationships, and inference rules in an incremental way, and organises the extracted knowledge into domain-specific graphs expressed in Rainbird's [Rainbird AI, 2026b] RBLang knowledge representation language, an XML-based rule language designed for explainable decision automation over structured domain knowledge. Rainbird enables the modelling of complex business logic through explicit concepts, relationships, and inference rules, supporting transparent and auditable decision-making processes. Its focus on explainability and deterministic reasoning makes it well-suited for domains where trust, traceability, and consistency are critical. Additional pipeline stages handle domain detection, conflict identification, graph merging, validation against the Rainbird API, and autonomous error correction, forming a self-correcting generation loop. The Rainbird API and RBLang specification are publicly available, enabling full reproducibility of the proposed pipeline. Free community accounts are also available, providing open access to the Rainbird platform.

The remainder of this paper is structured as follows. Section 2 reviews related work on KG construction and information extraction using LLMs. Section 3 describes the proposed approach at a conceptual level. Section 4 provides a detailed explanation of the implementation. Section 5 concludes the paper and mentions directions for future work.

2 Related Work

The task of extracting structured knowledge from unstructured text has been widely studied, with approaches ranging from traditional information extraction techniques to modern neural methods. Early work focused on rule-based systems and statistical models for named entity recognition and relation extraction. More recently, transformer-based models and LLMs have significantly advanced the ability to extract entities, relationships, and semantic structures directly from raw text [Bian, 2025].

However, most of these approaches focus on generating descriptive knowledge (e.g. entities and relationships), whereas Rainbird KGs must also encode decision logic, including rules and conditions. This makes the extraction task fundamentally more complex than standard KG construction.

A representative example of these capabilities is Rainbird's own Generate from Docs (GFD) pipeline [Rainbird AI, 2026a], which applies LLM-based generation to KG construction but exhibits several limitations common to single-pass approaches. The current GFD pipeline consists of a three-step LLM process executed in a single-shot manner. First, the text from all input documents is extracted, aggregated and provided to an LLM to generate a KG. A second LLM then attempts to validate and correct any errors in the generated graph, and a final step enriches the graph by adding questions to the relationships.

While effective for small inputs, this approach presents several limitations. The number of documents is restricted (currently up to six), which prevents its application to large-scale collections. Furthermore, all input documents are merged into a single generation step, resulting in a single KG regardless of whether the documents span multiple domains or topics. This can lead to structurally incoherent graphs when unrelated or loosely related content is combined.

Another key limitation is the lack of explicit conflict handling. When multiple documents contain contradictory information, the system does not surface or track these inconsistencies. Instead, the resolution is implicitly left to the LLM during generation, making it unclear how conflicts are represented or resolved in the final graph. This lack of transparency is particularly problematic for decision-making systems, where conflicting knowledge may have significant downstream effects.

More recent research has explored modular and iterative pipelines for KG construction, which decompose the process into smaller steps such as extraction, validation, and refinement [Meher *et al.*, 2025; Zhang *et al.*, 2025]. These approaches aim to improve scalability, interpretability, and control compared to single-shot generation.

These limitations highlight the need for a more scalable and structured approach to KG construction. In particular, there is a need for systems that can process large document collections, separate knowledge into coherent domains, and detect and report conflicts across sources. Additionally, moving from a single-shot generation process to an iterative, incremental approach enables the construction of larger and more complex KGs, while avoiding the context limitations that often cause information loss in single-pass generation.

3 Proposed approach

The core idea of the proposed system is to treat KG construction as an incremental, document-by-document process mediated by an LLM. Rather than attempting to construct an entire KG in a single pass over all available documents, the pipeline processes one document at a time, updating a set of domain-specific graphs at each step. This incremental design has two important consequences: it keeps the amount of context presented to the LLM at any one time manageable, and it allows the system to detect and record conflicts as they arise, rather than discovering them only after all documents have been processed.

Each document passes through a sequence of specialised stages. Before any knowledge is extracted, the system determines which domain the document belongs to, either matching it to an existing domain or declaring a new one. This domain assignment governs which existing KG is consulted and updated during subsequent stages. Once the domain is known, the incoming document is compared against the current state of that domain's graph to identify potential conflicts, which are logged but do not halt the pipeline. The extraction stage then produces an incremental update to the graph, containing only the new concepts, relationships, instances, and rules contributed by the current document. This update is merged into the domain graph, validated against a formal knowledge representation environment, and if necessary corrected through an autonomous repair loop before the pipeline moves on to the next document.

4 Implementation

The system was implemented using Python and LangGraph [LangChain Inc., 2026]. Figure 1 shows the schematic of the implementation on how each node is connected to obtain the full iterative generation flow. The flow is composed of five different LLM calls with different roles each one of them, together with two validation nodes that interact with the Rainbird API. Each of the nodes are explained as follows:

- **Detect domain:** Before any knowledge is extracted, the detect_domain node determines where in the graph the document's knowledge belongs. Acting as a domain classification expert, it sends the document to the model alongside the list of already-known domains. The model analyses the document's content, terminology, and context to either match it to an existing domain or declare a new one, returning a structured JSON response with a domain ID, name, description, confidence level, and the key concepts found. Domains are automatically detected and created by the LLM itself, and the system currently does not support the manual definition or specifying of custom domains.
- **Detect conflict:** With the domain identified, the detect_conflicts node compares the incoming document against everything already known in that domain's graph. The model examines both sources for five categories of conflict: direct factual contradictions, semantic differences in how the same concept is defined, clashing business rules, structural inconsistencies in entity relationships, and overlapping subject matter treated

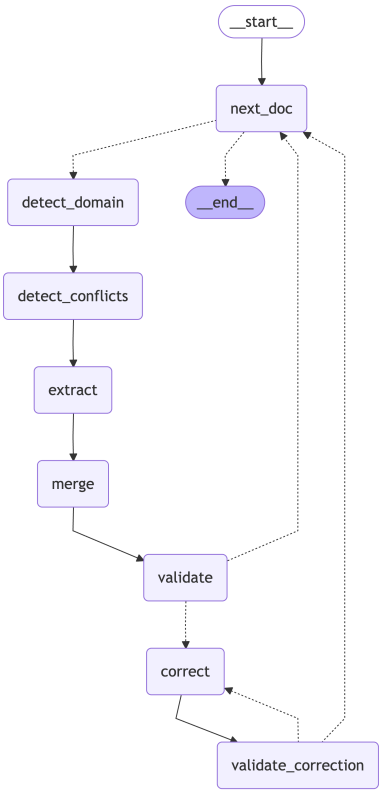

Figure 1: Incremental KG autogeneration workflow

differently across sources. The prompt calibrates the model's severity judgements conservatively, only flagging something as high severity when it represents a true logical contradiction, and marking as medium anything that could be resolved by adding rules or asserting priority. The resulting conflict report becomes an audit trail that helps knowledge engineers understand where their source documents disagree, without blocking the pipeline from continuing.

- **Extract:** The extract node is the intellectual core of the pipeline. It presents the model with both the document to process and the domain's existing KG and then asks it to output only the difference from the previous graph: the new concepts, relationships, instances, and rules that the document contributes, expressed in Rainbird's RBLang. This incremental approach is the key architectural decision that makes the system scalable: rather than regenerating the entire graph with each new document, the model produces only what needs to be added. The prompt embeds the full Rainbird RBLang specification as inline guidance, covering everything from concept types and relationship definitions to inference rules, conditional logic, date arithmetic, and certainty factors, giving the model everything it needs to produce valid, semantically rich knowledge in a single pass.
- **Merge:** Once the new information has been extracted, the merge node integrates it into the domain's cumulative graph. The model receives both the current graph and the new extraction and returns a single, complete merged RBLang document, deduplicating any entities or relationships that already exist, maintaining referential consistency throughout, and stripping any error comments left over from previous correction cycles. The prompt is kept concise, providing the two graphs and the merge objective without enumerating low-level rules, trusting the model's understanding of XML structure rather than over-specifying merge rules. After a successful merge, the updated domain graph is saved to disk, creating a persistent checkpoint at every step of the pipeline.
- **Validate:** The validate step sends the merged RBLang to the Rainbird Platform using its API. The Rainbird Platform attempts to save the graph and returns any validation errors encountered. A key limitation of this step is that the API only returns the first error it finds. As a result, a single validation pass is insufficient to guarantee correctness, and an iterative process is required to identify and resolve all issues in the graph.
- **Correct:** When the validation step fails, the correct node attempts to repair the graph autonomously. It sends the model the broken RBLang alongside the exact error messages returned by the Rainbird API, instructing it not only to fix the reported issues but to scan the rest of the graph for similar problems and resolve those proactively. The prompt embeds the full Rainbird RBLang specification, the same reference used during extraction, so the model can reason about schema rules while making corrections. A counter tracks how many attempts have been made, capping the correction loop at ten iterations to prevent infinite cycles in cases where errors prove intractable.
- **Validate correction:** This step re-applies the same validation mechanism as the initial validate node. After each correction attempt, it re-submits the updated graph to the Rainbird API to determine whether the fix succeeded. A passing result resets the correction counter and advances the pipeline to the next document, while a failing result sends the graph back to the *correct* node for another attempt, provided the attempt limit has not been reached.

This loop is executed for every document until all have been processed. At the end of the process, every complete and merged KG is stored in the Rainbird Platform, constructed from all input documents for each specific domain. Moreover, a comprehensive conflict report is generated, providing visibility into inconsistencies across sources.

This pipeline has been tested in proofs of concept with industry clients. It was applied to real-world document collections covering multiple domains. Due to non-disclosure agreements with these partners, we are unable to report detailed quantitative results. Nevertheless, the system was able to generate validated RBLang KGs, identifying conflicts across documents, and iteratively refining outputs through the correction loop without continuous domain expert intervention. While these results are qualitative and not based on a formal benchmark, they provide initial evidence of the practical applicability of this approach to iteratively generate KGs.

5 Conclusions and Future work

This paper presents a pipeline for automated KG generation from document collections, using LLMs to perform domain detection, conflict identification, incremental knowledge extraction, graph merging, and error correction. The system demonstrates that it is possible to construct large, extended KGs from a substantial number of documents in significantly less time than manual approaches would require, with the generated graphs persisted as validated RBLang graphs in the Rainbird Platform.

It is important to acknowledge the limitations of LLM-based generation. LLMs are non-deterministic and prone to hallucinations [Ji *et al.*, 2023], meaning that the generated graphs may contain errors, omissions, or fabricated relationships that are not present in the source documents. The graphs produced by this pipeline should therefore be treated as a starting point for expert review rather than a finished product, and human-in-the-loop validation is strongly recommended before any generated graph is used in a production setting. Based on this, several directions for future work have been identified:

- **Conflict resolution:** The conflict resolution mechanism could be substantially improved. The current system shows conflicts as a structured report but leaves their resolution to human knowledge engineers. Future work could investigate automated reconciliation strategies, either through LLM-based reasoning or through rule-based approaches that apply priority orderings or source credibility weights to resolve disagreements of known types.
- **Evaluation metrics:** There is currently no robust method for evaluating whether the generated graph fully and accurately captures the information present in the source documents. Developing evaluation metrics for KG completeness and correctness would allow the pipeline's output quality to be measured systematically. Iterative generation strategies that loop back over the source documents to check for missed information could be explored in this context.
- **Scalability:** The scalability of the approach to very large document collections could be improved. The current incremental design already avoids the need to regenerate the entire graph with each new document, but the context window presented to the LLM during extraction and merging grows as the domain graph accumulates knowledge. Chunking strategies, graph summarisation, or retrieval-augmented approaches that present only the most relevant portions of the existing graph to the LLM at each step could extend the system's effective capacity.
- **Limited domain detection:** The domain is currently identified once before processing each document, and the same domain is assumed for the entire document. This can be limiting, as a single document may contain information for multiple domains. Future work could focus on segmenting documents and detecting multiple domains within them, allowing the system to route different parts of the content to the appropriate domain-specific graphs. Additionally, domains are fully detected and generated by the LLM, and there is currently no option to manually define them.

Overall, this project demonstrates that LLMs can be effectively used to automate the construction of structured, extensible Rainbird KGs from unstructured documents, significantly reducing the time and effort required compared to manual approaches. As with any LLM-driven system, outputs should be treated as an informed first draft: a foundation that accelerates knowledge engineering rather than a substitute for it, and one that domain experts should review carefully before deployment.

References

[Bian, 2025] Haonan Bian. Llm-empowered knowledge graph construction: A survey. *arXiv preprint arXiv:2510.20345*, 2025.

[Hogan *et al.*, 2021] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, et al. Knowledge graphs. *ACM Computing Surveys (Csur)*, 54(4):1–37, 2021.

[Ji *et al.*, 2021] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S Yu. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE transactions on neural networks and learning systems*, 33(2):494–514, 2021.

[Ji *et al.*, 2023] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM computing surveys*, 55(12):1–38, 2023.

[LangChain Inc., 2026] LangChain Inc. Langgraph. https://github.com/langchain-ai/langgraph, 2026. Accessed: 2026-04-20.

[Meher *et al.*, 2025] Dipak Meher, Carlotta Domeniconi, and Guadalupe Correa-Cabrera. Core-kg: An llm-driven knowledge graph construction framework for human smuggling networks. *arXiv preprint arXiv:2506.21607*, 2025.

[Paulheim, 2016] Heiko Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web*, 8(3):489–508, 2016.

[Rainbird AI, 2026a] Rainbird AI. Generate from documentation. https://docs.rainbird.ai/rainbird/knowledge-modelling/modelling-features/building-with-co-author, 2026. Accessed: 2026-04-20.

[Rainbird AI, 2026b] Rainbird AI. Rainbird AI. https://www.rainbird.ai/, 2026. Accessed: 2026-04-20.

[Zhang *et al.*, 2025] Jian Zhang, Shihao Qi, Yuxuan Dong, Li Yuan, Tiesunlong Shen, Weiping Fu, Bifan Wei, Haiping Zhu, and Jun Liu. Gkg-llm: A unified framework for generalized knowledge graph construction. *Information Fusion*, page 103956, 2025.