

A Human-in-the-Loop Text-to-Triples Pipeline for Multi-Extractor Knowledge Graph Construction with Wikidata-grounded Extractors

Sonia HADJAB¹, Antoine PY¹, Carmelle MELI SONGUON¹ and Yoan CHABOT¹

¹Orange Research Belfort, France

Abstract

Building an enterprise knowledge graph from heterogeneous data sources remains challenging and requires significant effort in consistency and quality control, especially when knowledge must be extracted from large collections of unstructured documents. Although automatic triple extraction approaches have advanced significantly, relying on a single extractor may produce incomplete or inaccurate results. Moreover, the growing diversity of extraction tools and configurations, especially LLM-based extractors whose outputs are sensitive to prompt design, creates a need to compare and combine different extraction approaches. Existing platforms, however, provide limited support for combining multiple extractors, human validation of extracted triples, and their integration into a Wikidata-grounded knowledge graph within a single workflow. In this demo, we present TripleFlow, an open-source, web-based platform¹ for interactive knowledge graph construction through a modular end-to-end pipeline that combines multiple triple extractors, human-in-the-loop validation, and integration of validated triples into a target knowledge graph. TripleFlow’s key feature is its native support for Wikidata-grounded extractors, whose entities and predicates are identified using Wikidata QIDs and PIDs. These identifiers can be linked to Wikidata or to a private Wikibase instance. TripleFlow also offers an ergonomic interface for exploring the resulting knowledge graph. We demonstrate the platform and discuss its positioning with respect to existing knowledge graph construction tools.

Keywords

Knowledge Graph Construction, Triple Extraction, Human-in-the-Loop, Multi-Extractor Pipeline, Wikidata.

1. Introduction

Most large enterprises possess vast amounts of documents and data sources (technical reports, contracts, manuals, patents, etc.) that are often distributed across different locations and difficult to exploit automatically. The same piece of knowledge may be documented in multiple places, using different formulations, or contained in documents that are difficult to retrieve. Knowledge Graphs (KGs) offer a convenient way to structure and leverage this knowledge, but enriching an enterprise KG from unstructured text remains a challenging task, as it requires ensuring consistency, quality, and traceability [1]. Moreover, manually defining triples is a highly time-consuming process that requires the involvement of domain experts.

Several approaches have been proposed for automatic triple extraction from text and can broadly be grouped into two families. Supervised sequence-to-sequence models [2] and pipeline-based methods combining entity linking and relation classification [3], extended with semantic parsing techniques such as Abstract Meaning Representation (AMR), are deterministic and achieve high accuracy in the domains for which they are trained, but require annotated datasets and predefined relation schemas. In contrast, LLM-based extractors leverage recent advances in Large Language Models (LLMs) to offer high flexibility and require no training data, but are prone to hallucinations, and their outputs may vary across executions [4]. Among recent LLM-based approaches, FlexCIE [5] combines an LLM with a

¹<https://github.com/Orange-OpenSource/Tripleflow>

Wikidata’26: Wikidata Workshop at ISWC 2026

✉ sonia.hadjab@orange.com (S. HADJAB); antoine.py@orange.com (A. PY); carmelles.melisonuon@orange.com (C. M. SONGUON); yoan.chabot@orange.com (Y. CHABOT)

ORCID 0009-0003-4175-0270 (S. HADJAB); 0009-0009-5439-6753 (A. PY); 0009-0002-3305-7479 (C. M. SONGUON); 0000-0001-5639-1504 (Y. CHABOT)



knowledge graph completeness analysis tool to extract triples from text and align them with a target KG schema, but it remains subject to the variability inherent to LLM-based generation. Since each family of approaches presents its own strengths and weaknesses [6], running multiple extractors in parallel on the same document can be beneficial; it produces a larger set of candidate triples and allows users to compare extractors and benefit from their complementary results.

While such extraction approaches are increasingly integrated into KG construction tools, most existing platforms rely on a single extractor and therefore inherit its specific limitations. To the best of our knowledge, no existing tool combines, within a unified workflow, the execution and comparison of multiple extractors, human validation of extracted triples, and support for Wikidata-grounded triples [7].

In this paper, we present TripleFlow, an open-source web-based platform that addresses these limitations through a modular end-to-end pipeline. TripleFlow integrates four main components: (1) document parsing and segmentation into textual chunks, (2) parallel triple extraction through pluggable extractors, (3) human-in-the-loop validation with per-triple actions (accept, reject, review, and edit), and (4) interactive graph visualization and integration of validated triples into a target knowledge graph.

The extractors integrated in TripleFlow produce Wikidata-grounded triples whose entities and predicates are identified using QIDs and PIDs. This places TripleFlow in a closed information extraction (cIE) setting and improves interoperability with RDF-based KGs. The target knowledge base is configurable: identifiers can point to Wikidata or to a private Wikibase instance. Wikibase is the open-source software behind Wikidata that allows organizations to create and manage their own structured knowledge bases while keeping control over their data. A private instance is particularly useful when relevant entities and properties are organization-specific and therefore not available in Wikidata. Validated triples can then be published directly to this private knowledge base, facilitating their integration into enterprise knowledge graphs. In addition, TripleFlow provides an open extractor registry: any extraction service exposed through an HTTP endpoint can be registered in the platform without modifying the source code, making TripleFlow independent of any specific extraction approach.

The remainder of this paper is organized as follows: Section 2 reviews related work. Section 3 describes the architecture of TripleFlow. Section 4 presents a demonstration of the platform. Section 5 discusses future work, and Section 6 concludes the paper.

2. Related Work

In this section, we focus on platforms that orchestrate the knowledge graph construction workflow, from document processing to human validation. We review the representative systems below and position TripleFlow with respect to them.

Programmatic extraction pipelines. iText2KG [8] uses zero-shot LLMs to incrementally build knowledge graphs: it extracts entities and relations document by document, checking whether a concept already exists in the graph to avoid duplicates and semantic inconsistencies. EDC [9] follows a three-step approach (open extraction, schema definition, and schema canonicalization) allowing it to build a coherent schema even when none is defined initially. However, both systems rely on a single built-in LLM-based extractor, offer no graphical interface for human validation, and do not natively align entities and predicates with Wikidata.

Interactive human-in-the-loop tools. CleanGraph [10] is a web-based tool for refining knowledge graphs through a human-in-the-loop approach: users can visualize graphs, perform CRUD operations, and use machine learning plugins to detect errors or suggest new relations. In contrast, it operates on already constructed graphs and does not cover the upstream extraction from text. SSKG Hub [11] is the closest system to TripleFlow. It transforms sustainability standards into auditable knowledge graphs through an LLM-centered pipeline with expert review and provenance tracking. Nevertheless, it relies on a single extractor and is specifically designed for sustainability standards documents, whereas TripleFlow is domain-independent.

Document processing frameworks. Docs2KG [12] extracts multimodal information (text, images, tables) from heterogeneous documents using a dual-path strategy combining layout analysis and markdown parsing, producing a unified knowledge graph with source references. It follows a Human-LLM collaborative approach, allowing users to manually refine the generated graph through a web interface, and supports adding extractors, although this requires modifying the source code. However, it relies on a single extraction process, without support for running multiple extractors in parallel, and does not align entities and predicates with Wikidata.

Unlike the systems described above, TripleFlow combines multi-extractor orchestration, human-in-the-loop validation, and support for Wikidata-grounded triples. Table 1 compares seven criteria: parallel extraction (multiple extractors on the same input), human validation (review and correction of triples), multimodal extraction (text, images, or tables), pluggable extractors (integration without modifying the core system), domain independence (applicability across domains), interactive Web UI (graphical user interaction), and reviewer roles (distinct roles or permissions during validation).

Table 1
Comparison of TripleFlow with existing KG construction tools.

System	Parallel extraction	Human validation	Multimodal extraction	Pluggable extractors	Domain-independent	Interactive Web UI	Reviewers roles
iText2KG	$\times$	$\times$	$\times$	$\times$	$\checkmark$	$\times$	$\times$
EDC	$\times$	$\times$	$\times$	$\times$	$\checkmark$	$\times$	$\times$
CleanGraph	$\times$	$\checkmark$	$\times$	$\times$	$\checkmark$	$\checkmark$	$\times$
Docs2KG	$\times$	$\checkmark$	$\checkmark$	$\times$	$\checkmark$	$\checkmark$	$\times$
SSKG Hub	$\times$	$\checkmark$	$\times$	$\times$	$\times$	$\checkmark$	$\checkmark$
TripleFlow	$\checkmark$	$\checkmark$	$\times$	$\checkmark$	$\checkmark$	$\checkmark$	$\times$

3. System Architecture

TripleFlow is implemented as a web-based platform for knowledge graph construction from textual documents. Figure 1 shows its overall architecture, which is organized into four main modules: document parsing and chunking, triple extraction, validation, and graph visualization and export.

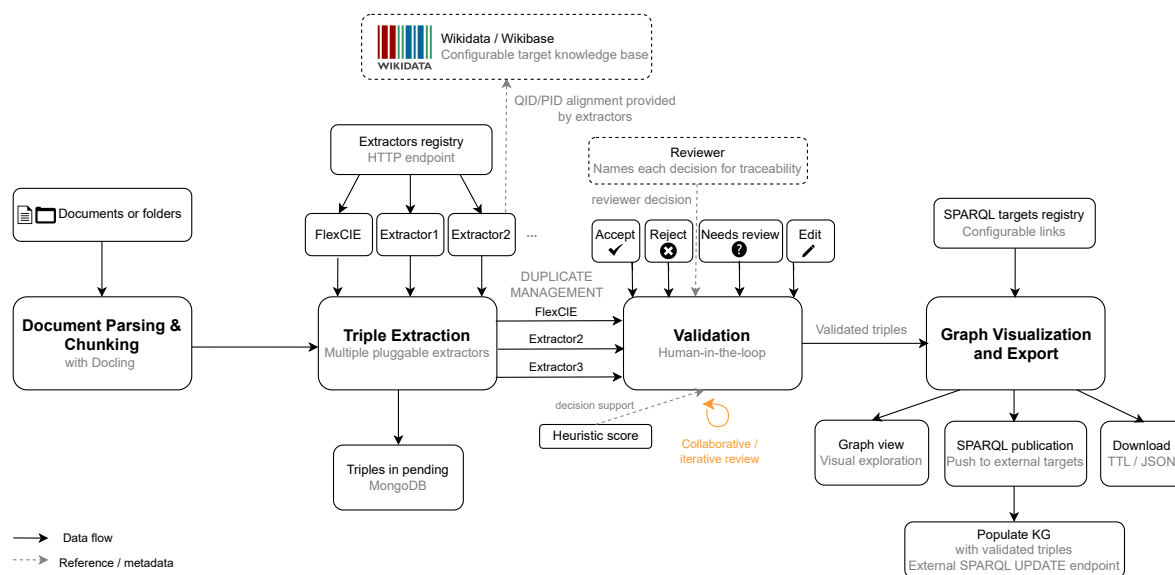


Figure 1: Overall architecture of TripleFlow.

3.1. Document Parsing and Chunking

Documents in .txt or .pdf format are uploaded through the extraction workspace, either individually or as a folder. Alternatively, users can paste text directly into the workspace. PDF documents are parsed with Docling¹, which identifies their textual structure, while visual and code content are excluded from the extraction pipeline. A chunking step then splits long paragraphs into coherent segments of a configurable size (512 tokens by default) with a configurable overlap (50 tokens by default). Each chunk retains positional metadata for provenance tracking: a chunk identifier, the start and end offsets of the chunk within the document, and the page number for PDF files. This metadata links every extracted triple to its exact location in the source text, which is later used in the validation interface. This module is optional in the workflow. When an extractor already integrates its own parsing and chunking, the module can be skipped to reduce processing time.

3.2. Triple Extraction

Once the chunks are built, the extraction engine sends each chunk to one or more extractors selected by the user. The extractors run in parallel on the same text, and a timer records the processing time of each one. Their results are then displayed in a clear and readable format, allowing the user to visually compare the outputs produced by each extractor.

Extractors are added only through the extractor registry (Section 3.5). Entity and predicate identification, candidate generation or retrieval, disambiguation, and alignment are performed by the extractors rather than by TripleFlow. Each extractor returns triples with subject, predicate, and object labels and, when available, their corresponding Wikidata identifiers (QIDs for entities and PIDs for predicates). Since different extractors may return different output structures, the extractor registry uses a configurable output mapping to indicate where these labels and identifiers can be found in each extractor's response. TripleFlow then normalizes these outputs into a common triple representation so that results from different extractors can be displayed, compared, and validated within the same interface. The unified triples are stored in MongoDB with a pending status.

An alignment coverage score is assigned to each extracted triple. It is a simple heuristic that counts how many of the three parts of a triple (subject, predicate, object) have a resolved identifier, and maps this count to one of four fixed levels, from 0.20 (none resolved) to 0.85 (all three resolved). A higher score therefore indicates greater identifier coverage and does not represent the likelihood that the triple itself is correct.

When several extractors return the same triple, TripleFlow groups these occurrences into a single row and records which extractors produced it. Two triples are considered the same when each part matches: by identifier when one is present, and by label (case-insensitive) otherwise. A resolved component, therefore, remains distinct from an unresolved one carrying the same label. This grouping avoids duplicate results and highlights agreement across extractors.

3.3. Validation

Extracted triples are presented to the reviewer in a tabular interface. Each row displays the triple (subject [QID], predicate [PID], object [QID]), its alignment coverage score, its source extractor(s), its current status, and a set of per-triple actions:

- **Accept:** the triple moves to validated and becomes eligible for integration into the target knowledge graph;
- **Reject:** the triple moves to rejected and is excluded from publication;
- **Review:** the triple moves to needs review for editing;
- **Edit:** the reviewer corrects the subject, predicate, or object if incorrect, then assigns the triple to any of the other statuses.

¹<https://github.com/docling-project/docling>

A sidebar lists all uploaded files (or manually entered text, labeled by default as `manual-input`), which can be renamed for clarity. The uploaded content is preserved so that another reviewer can verify or continue the work of a previous one. To support this collaborative setting, a side panel displays the source text of the document, and the entities and the predicate of the selected triple are highlighted within it, giving the reviewer the context from which each triple was extracted. Every action is recorded with the reviewer's name and a timestamp to ensure traceability. Triples can also be filtered by status or by extractor.

3.4. Graph Visualization and Export

Triples can be explored through an interactive graph view, which provides a visual representation of the extracted knowledge and can be filtered by status or by source file. TripleFlow offers two ways to output validated triples. First, they can be published directly to a target knowledge graph through either a configurable SPARQL endpoint or a custom HTTP API. For SPARQL endpoints, the user defines a parameterized query template adapted to the target graph, which TripleFlow fills with the validated triples before sending the resulting SPARQL UPDATE query. For custom HTTP APIs, each triple is sent through a configurable request. Second, validated triples can be exported in Turtle (TTL) or JSON format for manual integration into other RDF-based knowledge graphs.

3.5. Extractor Registry

The extractor registry allows new extraction services to be integrated into TripleFlow without writing code or redeploying the application. Any service exposed through an HTTP endpoint can be registered directly from the interface: the user describes the call (URL, headers, request body, and a configurable timeout) as well as how to interpret the response. For the latter, two modes are available. If the service follows the standard TripleFlow output contract, no configuration is needed. Otherwise, a custom mapping based on dot-paths tells the backend where to read each component of the triple in an arbitrary response. The registry also handles the grounded identifiers returned by the extractors: these identifiers become clickable links to Wikidata or to a private Wikibase instance, connecting the extracted triples to the organization's knowledge base. Finally, a built-in test lets users validate the configuration before saving: the user provides a sample text, the backend calls the endpoint, applies the mapping, and shows the resulting triples. This makes it easy to debug.

4. Demonstration

We demonstrate TripleFlow on a text document from the telecommunications domain (Figure 2), containing sentences such as:

“Orange S.A., formerly France Télécom S.A. (stylized as `france telecom`) is a French multinational telecommunications corporation. It has 266 million customers worldwide and employs 89,000 people in France and 59,000 people elsewhere. In 2015, the group had a revenue of €40 billion. The company's head office is located in Issy-les-Moulineaux.”

The user first selects two internally developed extractors, FlexCIE and KEFT (1), and provides the input text (2). Optional parsing and chunking steps can be configured before extraction (3). Both extractors process the same input in parallel, and their outputs and processing times are displayed separately (4), allowing the user to directly compare their results.

From the text above, the two extractors produce several triples, with Wikidata identifiers when available. For example, both extractors identify the following triple:

- Orange S.A. [Q1431486] — headquarters location [P159] — Issy-les-Moulineaux [Q135265]

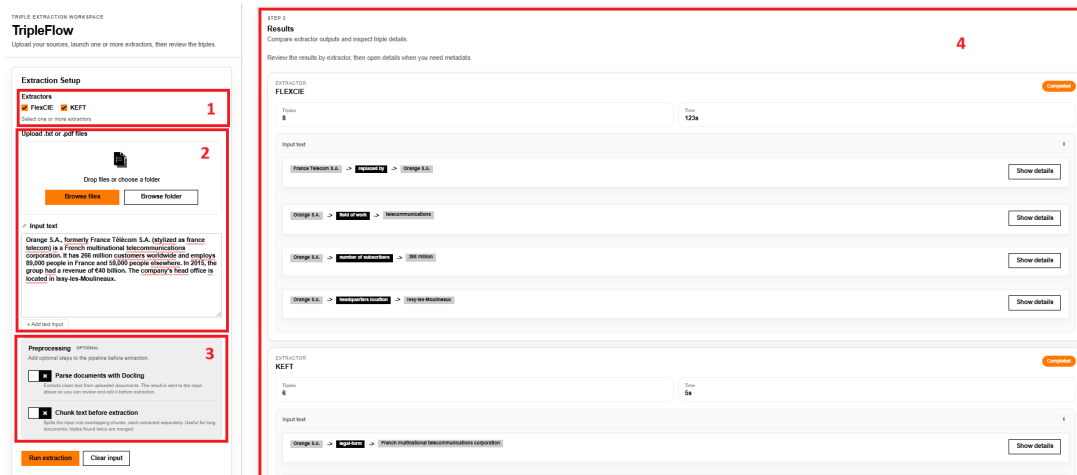


Figure 2: Extraction workspace - pipeline overview and extracted triples.

The extracted triples are then sent for validation (Figure 3). They are listed in a table (1), where each row shows the triple with its entity and predicate identifiers, alignment coverage score, source extractor(s), status, and available review actions. The left panel (2) lists all processed documents and allows the reviewer to filter the table to show only the triples of a given document. The right panel (3) highlights the entities and the predicate of the selected triple in the source text, giving the reviewer the context from which it was extracted. Validated triples can finally be explored as an interactive graph, published to a target knowledge graph, or exported in TTL and JSON (4).

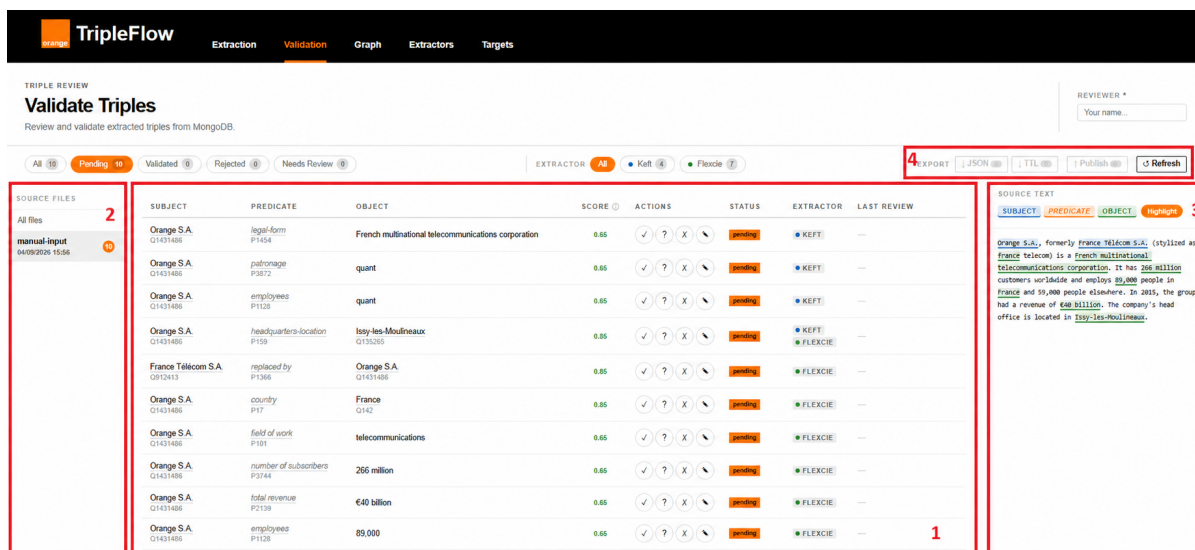


Figure 3: Validation interface with per-triple actions and source text highlighting.

5. Future Work

Each candidate triple is currently associated with a heuristic alignment coverage score based on the presence of resolved identifiers. A natural next step would be to complement this score with a more robust confidence measure that combines signals such as extractor agreement, the reliability of each extractor, and the consistency of a triple across extractors, in order to better estimate the reliability of the extracted triples and guide the reviewer during validation.

We also want to expand the range of available extractors. Currently, an extractor must be exposed through an HTTP endpoint to be registered; extractors that do not provide a web API, such as local libraries, must first be wrapped in an HTTP service. The next step is to remove this requirement.

Furthermore, comparison across extractors currently relies on visual inspection by the reviewer. Incorporating quantitative metrics into the interface, such as agreement between extractors, validation rates, and processing time, would allow for a better comparison of the tools and their configurations, which is particularly useful for LLM-based extractors, whose results depend heavily on the prompt used.

Another direction for future work is to explore the integration of TripleFlow with the Knowledge Integration Framework (KIF) [13], a Wikidata-based framework designed to integrate heterogeneous knowledge sources through a common representation. Such an integration could extend TripleFlow’s interoperability with knowledge bases beyond Wikidata, such as DBpedia [14] or PubChem [15], while maintaining a common knowledge representation layer.

Finally, the validation process does not include roles or conflict resolution: when two reviewers make different decisions on the same triple, the most recent decision overrides the previous one. Future versions will introduce review roles and explicit conflict detection, so that disagreements are clearly flagged and resolved.

6. Conclusion

Building an enterprise knowledge graph from text is not a fully automatic process. Extractors can generate candidate triples, but human validation remains necessary before their integration into the graph. TripleFlow addresses this need by combining multiple extractors, human-in-the-loop validation, and support for Wikidata-grounded triples within a single workflow. Validated triples can then be published to a target knowledge graph or exported for later integration. This demonstration shows how TripleFlow provides a practical workflow for enterprise knowledge graph construction.

Acknowledgments

This work was supported by the French National Research Agency (Agence Nationale de la Recherche - ANR) under the ECLADATTA project (ExtraCtion of LATent knowledge in Documents by conjointly Analyzing Texts and Tables, <https://ecladatta.github.io/>), grant number ANR-22-CE23-0020 (2023–2026).

Declaration on Generative AI

During the preparation of this work, the authors used DeepL for text translation from French to English, Grammarly and ChatGPT for grammar and spelling checking, paraphrasing, and improving writing style. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the publication’s content.

References

- [1] L. Jarnac, Y. Chabot, M. Couceiro, Uncertainty Management in the Construction of Knowledge Graphs: a Survey, *Transactions on Graph Data and Knowledge* 3 (2025) 1–48. doi:10.4230/TGDK.3.1.3.
- [2] P.-L. Huguet Cabot, R. Navigli, REBEL: Relation Extraction By End-to-end Language generation, in: *Findings of the Association for Computational Linguistics: EMNLP 2021*, Association for Computational Linguistics, 2021, pp. 2370–2381. doi:10.18653/v1/2021.findings-emnlp.204.
- [3] A. Shimorina, J. Heinecke, F. Herledan, Knowledge Extraction From Texts Based on Wikidata, in: *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), Industry Track*, Association for Computational Linguistics, 2022, pp. 297–304. doi:10.18653/v1/2022.naacl-industry.33.

- [4] D. Xu, W. Chen, W. Peng, C. Zhang, T. Xu, X. Zhao, X. Wu, Y. Zheng, Y. Wang, E. Chen, Large Language Models for Generative Information Extraction: A Survey, *Frontiers of Computer Science* (2024). doi:10.1007/s11704-024-40555-y.
- [5] C. Meli Songuon, Y. Chabot, L. Jarnac, V.-P. Huynh, A Guided and Flexible LLM-based Approach for Knowledge Extraction from Text, in: *Proceedings of the 13th Knowledge Capture Conference 2025 (K-CAP)*, Association for Computing Machinery, 2025, p. 223–226. URL: <https://doi.org/10.1145/3731443.3771373>. doi:10.1145/3731443.3771373.
- [6] L. Zhong, J. Wu, Q. Li, H. Peng, X. Wu, A Comprehensive Survey on Automatic Knowledge Graph Construction, *ACM Computing Surveys* 56 (2023) 1–62. doi:10.1145/3618295.
- [7] D. Vrandečić, M. Krötzsch, Wikidata: a free collaborative knowledgebase, *Commun. ACM* (2014). doi:10.1145/2629489.
- [8] Y. Lairgi, L. Moncla, R. Cazabet, K. Benabdeslem, P. Cléau, iText2KG: Incremental Knowledge Graphs Construction Using Large Language Models, in: *Web Information Systems Engineering – WISE 2024: 25th International Conference, Doha, Qatar, December 2–5, 2024, Proceedings, Part IV*, 2024, p. 214–229. doi:10.1007/978-981-96-0573-6_16.
- [9] B. Zhang, H. Soh, Extract, Define, Canonicalize: An LLM-based Framework for Knowledge Graph Construction, in: *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Miami, Florida, USA, 2024, pp. 9820–9836. doi:10.18653/v1/2024.emnlp-main.548.
- [10] T. Bikaun, M. Stewart, W. Liu, CleanGraph: Human-in-the-loop knowledge graph refinement and completion, 2024. arXiv:2405.03932.
- [11] C. He, X. Zhou, X. Yu, L. Zhang, Y. Zhang, Y. Wu, L. Xiao, L. Li, D. Wang, H. Xu, X. Wang, W. Liu, C. Miao, SSKG Hub: An expert-guided platform for LLM-empowered sustainability standards knowledge graphs, 2026. arXiv:2603.00669.
- [12] Q. Sun, S. Li, Y. Luo, J. Li, W. Zhang, K. Niu, X. Kong, W. Liu, Docs2KG: Unified knowledge graph construction from heterogeneous documents assisted by large language models, 2024. arXiv:2406.02962.
- [13] G. Lima, J. M. B. Rodrigues, M. Machado, E. Soares, S. R. Fiorini, R. Thiago, L. G. Azevedo, V. T. da Silva, R. Cerqueira, Kif: A wikidata-based framework for integrating heterogeneous knowledge sources, arXiv preprint arXiv:2403.10304 (2024).
- [14] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, Z. Ives, DBpedia: A Nucleus for a Web of Open Data, in: *The Semantic Web*, Springer Berlin Heidelberg, 2007, pp. 722–735.
- [15] S. Kim, J. Chen, T. Cheng, A. Gindulyte, J. He, S. He, Q. Li, B. A. Shoemaker, P. A. Thiessen, B. Yu, L. Zaslavsky, J. Zhang, E. E. Bolton, PubChem 2023 update, *Nucleic Acids Research* 51 (2023) D1373–D1380. doi:10.1093/nar/gkac956.