

Designing and Evaluating Knowledge Graph Pipelines with KGpipe Explorer

Marvin Hofer^{1,2,*}, Oliver Welz^{1,2,†} and Erhard Rahm^{1,2,†}

¹*Institute of Computer Science, Leipzig University, Germany, <https://cs.uni-leipzig.de>*

²*Center for Scalable Data Analytics and Artificial Intelligence (ScaDS.AI), Dresden/Leipzig, Germany, <https://scads.ai>*

Abstract

Knowledge graph (KG) pipelines integrate heterogeneous data sources, diverse processing tasks, and multiple implementation strategies. Despite their importance, existing approaches provide limited support for systematically designing, comparing, and understanding such pipelines.

We present KGpipe Explorer, an interactive demonstrator for constructing, analyzing, and comparing KG integration pipelines. Users visually compose typed directed acyclic graphs, export executable configurations, inspect precomputed benchmark results and stage-specific metadata, and rank pipelines under customizable quality objectives. The system supports responsive exploration of how pipeline choices affect KG quality and reveals trade-offs between alternative task and pipeline configurations. **Demo URL:** <https://vehnem.github.io/kgpipe-explorer/>

Keywords

Knowledge Graph, Data Integration, Benchmarking

1. Introduction

Knowledge graphs are commonly constructed through multi-task pipelines involving extraction, transformation, linking, and fusion of heterogeneous data sources [1]. These pipelines vary significantly in structure and implementation, and even small design differences can lead to substantial variations in the resulting knowledge graphs [2, 3].

However, current KG construction workflows are often implemented as opaque scripts or loosely structured workflows, making it difficult to (i) design pipelines in a modular and reusable manner, (ii) compare alternative pipelines under consistent evaluation criteria, and (iii) understand how individual pipeline components influence final outcomes [4]. Recent work has also highlighted the challenges of systematically evaluating and comparing alternative KG construction pipelines [5].

To address these challenges, we present KGpipe Explorer, an interactive demonstrator built on top of the KGpipe framework [6]. The system supports the design-analysis loop of pipeline exploration by enabling users to: (i) visually construct and validate pipelines as typed directed acyclic graphs, (ii) export pipelines as executable configurations, (iii) analyze precomputed execution results and intermediate artifacts, and (iv) compare and rank pipelines under customizable evaluation criteria.

Exported configurations are run externally with KGpipe, while prepared metrics and artifact metadata are integrated into the Explorer backend for analysis.

KGpipe Explorer complements the KGpipe execution framework with an interactive environment for pipeline design, metadata exploration, result comparison, and configurable multi-objective ranking. Execution remains external because KGpipe tasks may depend on containers, mounted datasets, and experiment-specific environments. PipeKG is KGpipe’s metadata knowledge graph, describing available pipeline tasks, their implementations, and evaluation metrics. KGpipe Explorer uses this metadata for pipeline design and exposes it for interactive exploration.

ISWC 2026 Companion Volume, October 25–29, 2026, Bari, Italy

*Corresponding author.

† These authors contributed equally.

✉ hofer@informatik.uni-leipzig.de (M. Hofer); oliver.welz@uni-leipzig.de (O. Welz); rahm@uni-leipzig.de (E. Rahm)

ORCID: 0000-0003-4667-5743 (M. Hofer); 0009-0005-4637-968X (O. Welz); 0000-0002-2665-1114 (E. Rahm)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

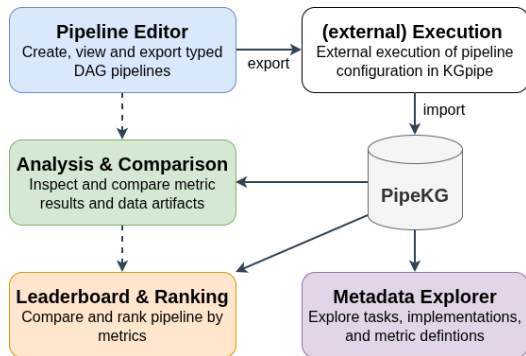
The demonstrator has the following contributions:

- a PipeKG-backed editor for constructing type-compatible pipelines and exporting executable configurations;
- live SPARQL-based metadata exploration and graph visualization;
- interfaces for comparing benchmark metrics and stage-specific artifacts;
- configurable multi-objective ranking with alternative groupings, weights, aggregations, and visualizations; and
- integrated tours and guided demonstration workflows.

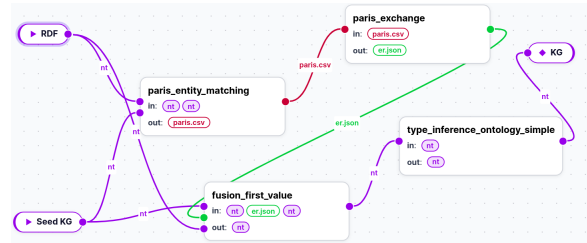
2. System Overview

KGpipe Explorer follows a three-stage workflow. First, users construct and export user-defined, type-compatible pipelines as directed acyclic graphs (DAGs) in the visual interface. Second, the exported configurations are executed externally using KGpipe, where the specified tasks produce the corresponding pipeline outputs. Third, prepared execution results, including evaluation metrics and stage-specific artifact metadata, are loaded into KGpipe Explorer for interactive exploration and comparison. The deployed demonstrator illustrates this third stage using prepared KGI-Bench runs.

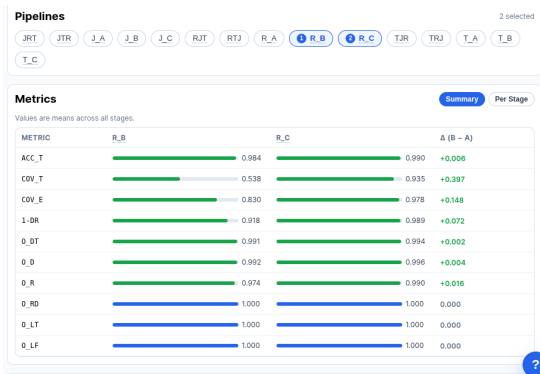
The interface comprises four sub-pages, complemented by a learn page with concise background material. The Pipeline Editor supports visual composition and validation of pipelines. The Results View enables comparison of execution outputs and inspection of intermediate artifacts. The Leaderboard ranks pipelines under customizable evaluation criteria, while the Metadata Explorer provides live SPARQL-based exploration and visualization of PipeKG metadata.



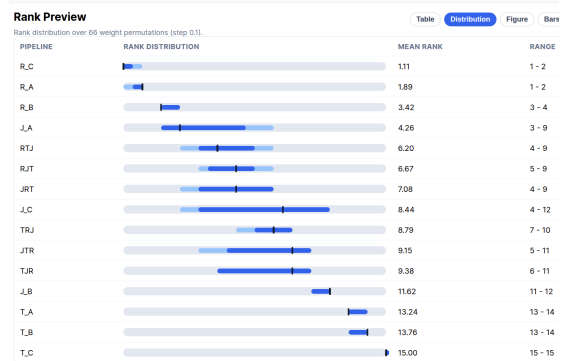
(a) Page architecture and workflow of KGpipe Explorer



(b) Pipeline Editor interface showing a typed DAG with alternative extraction branches and validated task connections.



(c) Results View comparing two pipelines across multiple evaluation metrics; differences between metric values are highlighted in the summary view.



(d) Rank distributions of pipelines across the evaluated metric-weight configurations. Bars indicate the observed distribution, while markers and labels report the mean rank and rank interval.

As illustrated in Figures 1a and 1b, KGpipe Explorer separates pipeline design, external execution, and interactive analysis. The Pipeline Editor represents tasks as nodes with typed input and output ports. Figure 1b shows part of the Movie KG running example described in Section 4, where matching results for the source data and seed KG are combined in the final fusion step.

Together, these components provide a unified environment for exploring KG pipeline designs and analyzing how design choices affect the resulting knowledge graphs, while execution remains delegated to external infrastructure.

3. Architecture & Design

This section describes the architecture of KGpipe Explorer and the underlying design principles that guide its development. The system supports the pipeline design-analysis loop while maintaining a clear separation between pipeline design, execution, and analysis.

3.1. System Architecture

Figure 1a illustrates the architecture of KGpipe Explorer. The system consists of three main layers: (i) pipeline design, (ii) external execution, and (iii) result exploration and analysis. The PipeKG metadata graph introduced in Section 1 is populated and updated during KGpipe execution and registers available tasks, implementations, and evaluation metrics.

The pipeline design layer is implemented through a React Flow canvas containing task nodes and explicit source and sink data nodes. Its task palette is populated from PipeKG and supports task search, specification inspection, parameter editing, and loading of example pipelines. Connections are validated client-side according to the input and output formats declared by the tasks. Valid graphs can be exported as YAML or JSON `pipeline.conf` files together with a corresponding KGpipe CLI preview.

Pipeline execution is intentionally decoupled from the system and performed externally using the KGpipe framework. This separation allows pipelines to be executed in heterogeneous environments and at different scales without affecting the responsiveness of the interface. It also enables the reuse of existing execution infrastructure and supports reproducible experiments.

The result exploration and analysis layer consists of the Results View, Leaderboard, and Metadata Explorer (PipeKG). These components operate on precomputed execution results that are imported into the system. The Results View presents benchmark metrics and stage-specific artifact metadata imported into the backend. In the current demonstrator, benchmark runs and artifact listings are provided through reproducible fixtures, while the interface and import model are designed to accommodate result bundles generated by external KGpipe executions. The Leaderboard supports configurable aggregation and ranking of pipelines under user-defined evaluation criteria. The Metadata Explorer provides live SPARQL CONSTRUCT access to PipeKG and visualizes returned subgraphs using Cytoscape. Users can start from predefined queries, discover and filter entity types, search for entities, insert selected URIs into queries, inspect outgoing triples, and follow deep links to task-specific details. User-defined query examples can be stored for later reuse.

KGpipe Explorer is implemented as a React and TypeScript web application, using React Flow for pipeline editing, Cytoscape for metadata-graph visualization, and driver.js for interactive tours. A FastAPI backend exposes task and metadata services, loads benchmark fixtures, and optionally connects to a Virtuoso-hosted PipeKG endpoint. Together, these layers form a modular architecture that separates concerns between construction, execution, and analysis, while providing a unified interface for pipeline exploration.

3.2. Data Flow and Execution Model

The system follows a structured data flow that reflects the separation between design and execution. First, users construct or modify pipelines in the Pipeline Builder. The resulting pipeline specifications

are exported as configuration files, which define the sequence of tasks, their parameters, and their interconnections.

Exported configurations are executed outside KGpipe Explorer using the KGpipe framework or experiment-specific tooling. The Explorer does not execute pipelines or automatically ingest arbitrary result bundles. Instead, measured outputs are integrated into the backend through benchmark metric files and artifact metadata. For the demonstrator, prepared benchmark fixtures are loaded at backend startup, providing reproducible and responsive result exploration without requiring a live execution environment.

Within the interface, users can then explore results at different levels of granularity. They can inspect stage-specific artifact metadata, compare pipelines based on evaluation metrics, and analyze ranking behavior under different weighting schemes. This workflow supports iterative experimentation, where users refine pipeline designs based on observed results.

3.3. Design Decisions and Trade-offs

KGpipe Explorer makes four principal design trade-offs. External execution preserves interface responsiveness and supports heterogeneous execution environments, but requires results to be generated before analysis. Typed DAGs prevent incompatible task connections while retaining structural flexibility. Precomputed results enable fast exploration at the cost of on-demand execution. PipeKG supports provenance and meta-level analysis, but introduces additional metadata-management overhead.

4. Use Cases and Demonstration Scenario

KGpipe Explorer supports several use cases centered on the systematic design, analysis, and comparison of knowledge graph construction pipelines. We demonstrate these capabilities using the KG integration benchmark (KGI-Bench) [7], which provides multiple data sources and alternative pipeline configurations. As a running example, we consider the construction of a movie KG from source RDF and an existing seed graph. The example pipeline processes these inputs through matching and fusion tasks, followed by type inference, to produce the resulting KG. Alternative pipeline configurations vary individual processing choices for the same integration problem, allowing their effects on intermediate artifacts and final KG quality to be compared.

The on-site demonstration follows this running example through three connected interactions. First, the visitor loads the example RDF pipeline, inspects its source and seed-KG inputs and its matching and fusion stages, replaces a fusion task, reconnects the graph, and exports the resulting `pipeline.conf`. Second, the visitor compares two prepared pipeline variants for the same Movie KG integration problem, inspects their stage-wise metrics and artifact metadata, and opens the Data View. Third, the visitor constructs a ranking configuration by selecting pipeline variants, creating accuracy and coverage groups, assigning metrics, and inspecting the resulting rank distribution. The Metadata Explorer can then be used to inspect the task definitions and PipeKG entities underlying the pipeline components. The deployed demonstrator additionally provides an end-to-end guide that follows this Movie KG scenario across the Pipeline Editor, Results View, Leaderboard, and Metadata Explorer.

This scenario illustrates the following representative use cases.

Pipeline Prototyping and Design Exploration. Users can compose alternative pipeline structures from typed components, validate their compatibility, and export executable configurations. This supports rapid experimentation with different data sources, extraction methods, matching techniques, and fusion strategies.

Comparative Evaluation and Pipeline Selection. Users can compare precomputed pipeline results under consistent quality and resource metrics. The Leaderboard supports customizable weighting schemes, making trade-offs explicit and helping users select pipelines for specific requirements.

Understanding Pipeline Behavior. Stage-specific metrics, artifact metadata, and PipeKG metadata help users relate differences in final results to individual tasks and implementations.

Educational Use. The visual representation of pipeline structures, intermediate results, and evaluation outcomes makes the system suitable for teaching and demonstrating KG construction workflows.

Integrated page tours and the end-to-end guide help visitors reproduce the running workflow across the Pipeline Editor, Results View, Leaderboard, and Metadata Explorer.

5. Results & Leaderboard

KGpipe Explorer supports systematic analysis of precomputed pipeline executions through stage-specific artifacts, quality metrics, and configurable ranking mechanisms. The evaluation model and aggregation of pipeline-quality metrics follow our previous work [7].

The Results View enables detailed inspection of pipeline outputs at different levels of granularity. In addition to final knowledge graphs, the system presents stage-specific artifact metadata and illustrative artifact listings at each pipeline stage, such as extracted entities, linking results, and fused triples. This allows users to analyze how individual processing steps contribute to the overall outcome.

Pipelines can be compared side-by-side using a range of evaluation metrics. The system supports core quality dimensions, including accuracy, coverage, and consistency, as well as auxiliary metrics such as runtime, resource usage, and task-specific evaluation scores. By combining structural information about pipelines with their execution results, KGpipe Explorer enables users to identify differences between pipelines and relate these differences to specific design choices. For the running Movie KG example, alternative pipeline configurations can be compared using coverage, correctness, and consistency metrics. Stage-specific scores and artifacts, such as matching results and fused triples, provide additional context for understanding where differences between pipeline outcomes arise.

As shown in Figure 1c, users can select pipelines and compare their evaluation scores side by side, making differences in individual metrics immediately visible. Furthermore, users can inspect stage-specific artifact metadata and illustrative listings, such as matching-score files, through the Data Artifacts and Data View interfaces.

A key feature of the system is the ability to analyze pipelines across multiple, potentially conflicting quality objectives. Rather than assuming a single optimal metric, KGpipe Explorer treats evaluation as a multi-dimensional problem. Users can explore trade-offs between quality dimensions, for example between maximizing coverage and maintaining high precision, or between result quality and computational cost.

The Leaderboard implements a configurable two-level ranking model. Users organize metrics into weighted groups and subgroups, select relevant pipeline stages, choose aggregation functions across stages and metrics, and interactively inspect the resulting ranks. Rankings can be inspected through tabular, distributional, bar-chart, and heatmap views, and configurations can be shared through compact URL-encoded state.

To highlight the impact of weighting choices, the system supports analyzing ranking distributions across multiple configurations. For example, by enumerating different weight combinations for accuracy, coverage, and consistency, users can observe how stable or sensitive pipeline rankings are under varying evaluation criteria. Figure 1d illustrates this by showing the distribution of ranks for each pipeline across a range of weight configurations, including mean rank and observed rank intervals.

The resulting distributions show that no pipeline is universally optimal: preferred pipelines depend on the selected quality objectives and weights. By exposing this sensitivity, the leaderboard supports informed pipeline selection under application-specific requirements.

6. Conclusion

KGpipe Explorer demonstrates how knowledge graph construction pipelines can be treated as first-class, composable, and analyzable objects. By decoupling pipeline design from execution and focusing on interactive exploration of results, the system provides a novel approach to understanding and optimizing KG construction workflows.

The demonstrator highlights the impact of pipeline design choices and supports principled comparison and selection of pipelines, contributing to more transparent and reproducible KG construction processes.

Acknowledgments

The authors acknowledge the financial support by the Federal Ministry of Education and Research of Germany and by the Sächsische Staatsministerium für Wissenschaft, Kultur und Tourismus in the program Center of Excellence for AI-research "Center for Scalable Data Analytics and Artificial Intelligence Dresden/Leipzig", project identification number: ScaDS.AI.

Declaration of use of Generative AI

During the preparation of this work, the authors used OpenAI ChatGPT for grammar and spelling checks, paraphrasing and rewording, and improving the writing style of the manuscript. Cursor was used to assist with the implementation of parts of the frontend based on the authors' specifications, existing KGpipe functionality, and outputs from the KGI-Bench experiments. All AI-assisted text and code were reviewed, edited, tested, and revised by the authors as appropriate. The authors take full responsibility for the implementation and the content of the publication.

References

- [1] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. D. Melo, C. Gutierrez, S. Kirrane, J. E. L. Gayo, R. Navigli, S. Neumaier, et al., Knowledge graphs, *ACM Computing Surveys (Csur)* 54 (2021) 1–37.
- [2] A. Zaveri, A. Rula, A. Maurino, R. Pietrobon, J. Lehmann, S. Auer, Quality assessment for linked data: A survey: A systematic literature review and conceptual framework, *Semantic web* 7 (2015) 63–93.
- [3] H. Paulheim, Knowledge graph refinement: A survey of approaches and evaluation methods, *Semantic Web* 8 (2017) 489–508. URL: <https://doi.org/10.3233/SW-160218>. doi:10.3233/SW-160218.
- [4] M. Hofer, D. Obraczka, A. Saeedi, H. Köpcke, E. Rahm, Construction of Knowledge Graphs: Current State and Challenges, *Information* 15 (2024) 509. URL: <https://www.mdpi.com/2078-2489/15/8/509>. doi:10.3390/info15080509.
- [5] G. Tamašauskaitė, P. Groth, Defining a knowledge graph development process through a systematic review, *ACM Transactions on Software Engineering and Methodology* 32 (2023) 1–40.
- [6] M. Hofer, E. Rahm, Kgpipe: Generation of pipelines for data integration into knowledge graphs, in: *15th International Workshop on Quality in Databases (QDB'26), VLDB 2026, 2026*. URL: https://vldb.org/2026/Workshops/VLDB-Workshops-2026/QDB/QDB26_7.pdf.
- [7] M. Hofer, E. Rahm, Evaluation of pipelines for data integration into knowledge graphs, *CoRR abs/2605.22304* (2026). URL: <https://doi.org/10.48550/arXiv.2605.22304>. doi:10.48550/ARXIV.2605.22304. arXiv:2605.22304.