KMatrix-2: A Comprehensive Heterogeneous Knowledge Collaborative Enhancement Toolkit for Large Language Model

Shun Wu[1], **Di Wu**[1], **Wangtao Sun**[1,2], **Ziyang Huang**[1,2], **Xiaowei Yuan**[1,2,4], **Kun Luo**[1,2], **XueYou Zhang**[1], **Shizhu He**[1,2*] **Jun Zhao**[1,2], **Kang Liu**[1,2,3*]
[1]The Key Laboratory of Cognition and Decision Intelligence for Complex Systems
Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences
[3]Shanghai Artificial Intelligence Laboratory [4]Beijing Academy of Artificial Intelligence
{shun.wu, jzhao, kliu}@nlpr.ia.ac.cn {di.wu, xueyou.zhang}@ia.ac.cn

Abstract

The paper presents *KMatrix-2*, an open-source toolkit that supports comprehensive heterogeneous knowledge collaborative enhancement for Large Language Models (LLMs). As the successor of KMatrix (Wu et al. 2024), our toolkit offers powerful modular components and typical enhancement patterns for convenient construction of mainstream knowledge-enhanced LLMs systems. Besides, it provides unified knowledge integration and joint knowledge retrieval methods to achieve more comprehensive heterogeneous knowledge collaborative enhancement. Compared with KMatrix which mainly focuses on descriptive knowledge, this work additionally considers procedural knowledge. Moreover, systematic inter-context and context-memory knowledge conflict resolution methods are offered for better knowledge integration. Some key research questions in heterogeneous knowledge-enhanced Large Language Models systems are analyzed, and our toolkit's capability in building such systems is validated. Our code and data resources are available here.[1]

1 Introduction

Knowledge-Enhanced Large Language Models (K-LLMs) systems are gaining increasing attention in recent years, which combine Large Language Models (LLMs) with external knowledge to enhance reasoning capabilities (Gao et al. 2023). Nowadays, K-LLMs systems have become a mainstream paradigm for mitigating hallucination issues and supporting domain-specific applications (Ji et al. 2023, Huang et al. 2023, Emelin et al. 2022).

There are two types of knowledge that critically contributes to the cognition of human beings: declarative knowledge ("knowing what") and procedural knowledge ("knowing how") according to metacognitive theories (Jacobs and Paris 1987). Existing K-LLMs studies mainly focus on declarative knowledge enhancement, such as textual knowledge (Karpukhin et al. 2020, Qu et al. 2020), tables and knowledge graphs (Oguz et al. 2020, Ma et al. 2022, Jiang et al. 2023a). Wang et al. (2024) and Sun et al. (2024) attempted to analyze the enhancement performance of procedural knowledge (rules) on LLMs. Meanwhile, knowledge conflict resolution is becoming an important sub-direction in K-LLMs research (Xu et al. 2024), which resolves the discrepancies among external symbolic knowledge and LLMs' parametric knowledge for better knowledge integration. Recently, several K-LLMs toolkits (Chase 2022, Pietsch et al. 2019, Izsak et al. 2023) have been developed, but they still have the following limitations: 1) Lacking support for collaborative enhancement with comprehensive declarative and procedural knowledge. The representative K-LLMs toolkits (Chase 2022, Hoshi et al. 2023, Wu et al. 2024) predominantly support declarative knowledge (like text, tables, knowledge graphs, etc) enhancement, but ignore procedural knowledge (such as rules) enhancement. 2) Not highly flexible or easy-to-use. RALLE (Hoshi et al. 2023) and Coze[2] enabled the construction of naive K-LLMs systems (one-time knowledge retrieval & generation) by simply selecting components, which is easy-to-use but lacks the flexibility to support complex K-LLMs systems construction (such as adaptive knowledge retrieval & generation). FlashRAG (Jin et al. 2024) utilized customizable components and defined component relations through hard-coding method to flexibly support the construction of complex K-LLMs systems. KMatrix (Wu et al. 2024) further improves the composability of component relations by using control-logic flow diagram. However, these toolkits require users' involvement in the entire process of component definition and relation design, which is not easy-to-use. 3) Lacking knowledge conflict resolution. Existing toolkits (Izsak et al. 2023,

*Corresponding author
[1]https://github.com/NLPerWS/KMatrix-2
[2]https://www.coze.com/store/plugin

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 591–601
November 4-9, 2025

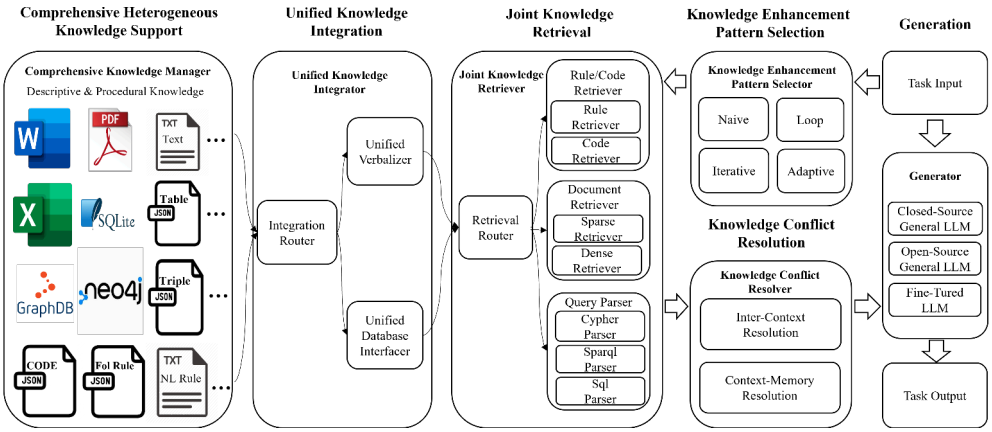


Figure 1: A overview framework of the KMatrix-2 toolkit

Wu et al. 2024, Edge et al. 2024) provide insufficient support for conflict resolution among external knowledge and LLMs' parametric knowledge.

To fill these gaps, we develop KMatrix-2: a comprehensive heterogeneous knowledge collaborative enhancement toolkit for large language model. As shown in Figure 1, KMatrix-2 contains six main stages, and it specifically provides Unified Knowledge Integration and Joint Knowledge Retrieval stages to achieve descriptive and procedural heterogeneous knowledge collaborative enhancement. Meanwhile, a rich of modular components (like Retriever, Generator, etc) and several typical enhancement patterns (like Loop, Adaptive, etc) are encapsulated, and can be combined to conveniently construct mainstream heterogeneous K-LLMs systems, which balances flexibility and ease of use. Moreover, we integrate inter-context and context-memory conflict resolution methods (Hong et al. 2023, Zhou et al. 2023) in KMatrix-2 to mitigate two kinds of conflicts: 1) conflict between external knowledge, and 2) conflict between external knowledge and LLMs' parametric knowledge, respectively. Our main contributions are:

1. The paper proposes a comprehensive heterogeneous knowledge collaborative enhancement toolkit (KMatrix-2) for LLMs. Compared with previous toolkits, which mainly focused on descriptive knowledge, KMatrix-2 specifically considers the enhancement on procedural knowledge.

2. KMatrix-2 offers a rich selection of modular components and several typical enhancement patterns to support convenient construction of mainstream heterogeneous K-LLMs systems.

3. We offer systematic knowledge conflict resolution methods for better knowledge integration.

4. Some key research questions in heterogeneous K-LLMs systems are analyzed, and our toolkit's capability in building such systems is validated.

2 KMatrix-2 Toolkit

As shown in Figure 1, our toolkit contains six stages to complete knowledge-enhanced generation task. Specifically, *Comprehensive Heterogeneous Knowledge Support* is used for the access of comprehensive knowledge. *Unified Knowledge Integration* supports the fusion of comprehensive knowledge. *Joint Knowledge Retrieval* supports the query of the integrated knowledge. *Knowledge Conflict Resolution* is used to mitigate conflicts among external knowledge and LLMs' parametric knowledge, and *Knowledge Enhancement Pattern Selection & Generation* is used for knowledge enhancement process control and answer generation. KMatrix-2 offers a rich selection of modular components and several typical enhancement patterns to implement the above six stages.

2.1 Comprehensive Heterogeneous Knowledge Support

We develop a *Comprehensive Knowledge Manager* component to support heterogeneous knowledge access and management. As shown in Table 1, KMatrix-2 supports the access of rich descriptive and procedural knowledge. It supports three forms of descriptive knowledge: text, table, and knowledge graph, and each knowledge form supports multiple formats. For example, textual knowledge includes formats like WORD and PDF. Tables includes formats such as EXCEL and SQLite. Knowledge graphs include graph databases like GraphDB and Neo4j. Additionally, KMatrix-2 supports two types of representative procedural knowledge: rule and code knowledge (Li et al. 2024). In our toolkit, rule knowledge could be represented as natural languages or formal languages (such as first-order logic). And code knowledge could be represented

as 14 mainstream languages, like Python, Java, etc.

Table 1: Comprehensive Knowledge Support

Knowledge Type	Knowledge Form	Format
Declarative knowledge	Text	WORD, PDF, TXT, Markdown, etc
	Table	EXCEL, Table JSON, SQLite, etc
	Knowledge Graph	Triple JSON, Neo4j, GraphDB, etc
Procedural knowledge	Rule	NL Rule TXT,etc
		FOL Rule JSON, etc
	Code	Formal Code JSON, etc

2.2 Unified Knowledge Integration

All descriptive and procedural knowledge can be divided into two formats: files (like WORD, EXCEL, Rule JSON, etc) and databases (like SQLite, GraphDB, Neo4j, etc). We achieve unified knowledge integration through *Unified Knowledge Integrator*. Specifically, we develop a *Integration Router* component to automatically distribute knowledge in format of files and databases (according to the tag of knowledge format), and build corresponding knowledge integrators for them.

For knowledge in format of files, we build a *Unified Verbalizer* component to achieve knowledge integration. The Unified Verbalizer converts different files into unified text fragments. Following KMatrix's Unified Verbalizer (Wu et al. 2024), we further develop a Verbalizer model for rules and code knowledge using template-based method. (e.g., {*rule body*: an animal lives in a cold area, *rule head*: it probably has thick fat or think fur} => If an animal lives in a cold area, then it probably has thick fat or think fur.)

For knowledge in format of databases, we build a *Unified Database Interfacer* component to achieve knowledge integration. The Unified Database Interfacer standardizes and integrates query interfaces of different types of databases. We develop a Unified Database Interfacer that supports SQLite, GraphDB, Neo4j, etc.

2.3 Joint Knowledge Retrieval

The knowledge outputted by the *Unified Knowledge Integrator* includes three types: a) text fragments from declarative knowledge, b) text fragments from procedural knowledge, and c) unified database interfacer. To achieve joint knowledge retrieval, we develop corresponding *Retriever* components for each type of knowledge and build a *Retrieval Router* component to automatically distribute knowledge to their respective Retrievers.

For text fragments from declarative knowledge, we develop the *Document Retriever* component to perform declarative knowledge retrieval, which includes sparse retrievers: BM25[3], and dense retrievers: Contriever (Izacard et al. 2021), DPR (Karpukhin et al. 2020), BGE (Xiao et al. 2023), Qwen3 (Zhang et al. 2025).

For text fragments from procedural knowledge, considering the inherent semantic gaps between procedural knowledge (code & rule) and declarative knowledge (Li et al. 2024), we build dedicated *Code and Rule Retriever* components with self-induction method, which utilizes LLMs to induce potential rules or code from natural language queries. And these induced rules or code are used for query augmentation to alleviate semantic gaps and improve retrieval effectiveness. A simplified instruction for self-induction method are provided in Appendix A.1.

For unified database interfacer, we develop corresponding *Query Parser* components for the query interfaces of SQLite, GraphDB, and Neo4j databases, which receive input contents and generate query statements specifically tailored for the query interfaces to obtain knowledge. Specifically, KMatrix-2 includes: 1) SQLite-oriented Sql Parser: a seq2seq SQL query generator based on Li et al. (2023a), 2) GraphDB-oriented Sparql Parser: a LLM-based SPARQL query generator inspired by Xu et al. (2023), and 3) Neo4j-oriented Cypher Parser: a Cypher query generator by transforming SPARQL into Cypher using a unified intermediate representation (Xu et al. 2023).

2.4 Knowledge Conflict Resolution

We build inter-context and context-memory conflict resolution methods to mitigate: 1) conflict between external knowledge, and 2) conflict between external knowledge and LLMs' parametric knowledge, respectively. And they are implemented by *Knowledge Conflict Resolver* component.

For inter-context (IC) conflict resolution, In addition to the conventional ICL (Dong et al. 2022) approach, we follow Hong et al. (2023), and use instruction-based discrimination to eliminate erroneous or conflicting information in external knowledge. A instruction for discrimination could be as follows: *Given these passages, some passages may*

[3] https://pypi.org/project/rank-bm25/

have been perturbed with wrong information. Find the perturbed passages if there are any.

For context-memory (CM) conflict resolution, KMatrix-2 supports two strategies: 1) faithful to context: aiming to align with context and focus on context prioritization (Xu et al. 2024). In specific, we follow Zhou et al. (2023), and employ counterfactual demonstration to realize this target. 2) factuality improvement: aiming for an integrated response leveraging both context and parametric knowledge towards a more truthful solution (Xu et al. 2024). In specific, we develop two methods based on COIECD (Yuan et al. 2024) and CAD (Shi et al. 2024), respectively. COIECD employs adaptive decoding with a contextual information-entropy constraint, while CAD utilizes context-aware decoding to implement this strategy.

2.5 Knowledge Enhancement Pattern Selection & Generation

KMatrix-2 designs four typical enhancement patterns to support the convenient construction of mainstream heterogeneous K-LLM systems, including *Naive, Loop, Iterative, and Adaptive*. Diagrams of these enhancement patterns are provided in Appendix A.3.

1) *Naive enhancement pattern*: this pattern supports the construction of naive K-LLMs system, which executes one-time retrieval and generation, and the representative works of naive K-LLMs system contain: DenseX (Chen et al. 2023), UPRISE (Cheng et al. 2023), etc.

2) *Loop enhancement pattern*: this pattern supports the construction of loop K-LLMs system, which executes query decomposition to obtain N subqueries firstly, and performs a N-step retrieval loop for these subqueries. Finally, it completes generation based on the retrieved knowledge. The representative works of loop K-LLMs system contain: COK (Li et al. 2023b), KnowledGPT (Wang et al. 2023), etc.

3) *Iterative enhancement pattern*: this pattern supports the construction of iterative K-LLMs system, which executes iterative retrievals and generations, and the representative works of iterative K-LLMs system contain: Interleave (Shao et al. 2023), ITRG (Feng et al. 2024), etc.

4) *Adaptive enhancement pattern*: this pattern supports the construction of adaptive K-LLMs system, which executes adaptive retrievals and generations on demand, and the representative works of adaptive K-LLMs system contain: SelfRAG (Asai et al. 2023), FLARE (Jiang et al. 2023b), etc.

To meet the needs of different generation scenarios, KMatrix-2 further integrates multiple *Generators* components on base of KMatrix (Wu et al. 2024). Our toolkit further includes: 1) a closed-source general Generator: GPT-4, 2) three open-source general Generators: Baichuan-2-13b (Yang et al. 2023), DeepSeek-R1 (Guo et al. 2025), Qwen2.5-14B-Instruct (Hui et al. 2024).

2.6 Toolkit Design & Usage

We design a concise user interface (UI) to hide K-LLMs design details for ease of use.[4] A rich of modular components and several typical enhancement patterns are encapsulated, and can be combined to conveniently construct mainstream K-LLMs systems. User interface (UI) of a loop K-LLMs system deployment is shown in Figure 2, and the usage instructions of our toolkit are provided in Appendix A.2.

3 Experimental Settings

3.1 Datesets and Knowledge

Evaluation datasets are shown in Table 6 (Appendix A.4). To evaluate the performance of knowledge retrieval, this paper selects COIR (Li et al. 2024) and RuleBench (Sun et al. 2024). In specific, COIR is used for code knowledge retrieval evaluation, which contains 14 main code languages. RuleBench is used for rule knowledge retrieval evaluation, which contains natural language and first-order logic rules. To evaluate the semantic parsing performance of Query Parser, this paper selects WWQ (Xu et al. 2023) and spider_realistic (Deng et al. 2020) to assess performance of Sparql and Sql Parsers, respectively. To evaluate descriptive and procedural knowledge enhancement performance of K-LLMs system, this paper selects: 1) 6 open domain question answering (QA) datasets from KMatrix, which is used to evaluate descriptive and procedural knowledge enhancement performance on factual QA task, and 2) RuleBench, which contains rule-following QA datasets from 5 domains and is used to evaluate descriptive and procedural knowledge enhancement performance on inferential QA task. And this paper selects ConflictBank (Su et al. 2024) and ConflictQA (Xie et al. 2024) to assess performance of IC and CM knowledge conflict resolution methods, respectively.

Heterogeneous knowledge used in evaluation is shown in Table 7 (Appendix A.4). For descriptive

[4] https://github.com/NLPerWS/KMatrix-2

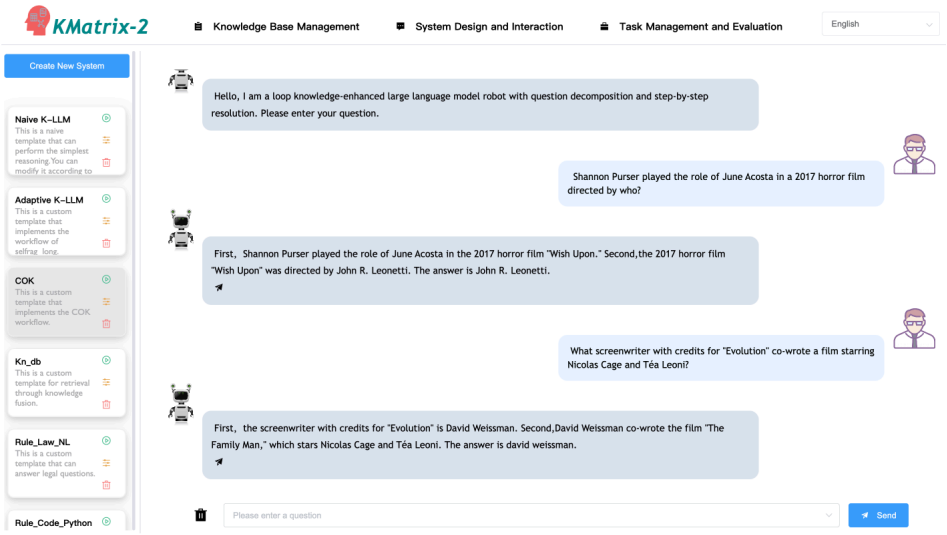


Figure 2: Deployment UI of KMatrix-2: toolkit usage instructions are provided in Appendix A.2.

knowledge, we integrate Wikipedia (Chen et al. 2017), Wikidata (Vrandečić and Krötzsch 2014) and Wikitable (Ma et al. 2022). For procedural knowledge, we integrate the natural language and first-order logic rules in RuleBench.

3.2 Task Settings

Some key research questions (RQs) in heterogeneous K-LLMs systems are analyzed, and our toolkit's capability in building such systems is validated.

RQ1: Can Self-Induction method bridge the semantic gap in procedural knowledge retrieval to boost performance? To validate this, we evaluate four retrieval models on two types of procedural knowledge: code and rules. The models include: 1) two representative dense retrievers: BGE (Xiao et al. 2023) and Qwen3 (Zhang et al. 2025), 2) their enhanced versions with our Self-Induction method: Self-Induction+BGE and Self-Induction+Qwen3.

RQ2: Do factual question answering (QA) and inferential QA tasks rely on different types of knowledge? To validate this, we evaluate descriptive and procedural knowledge enhancement performance of K-LLMs system on factual and inferential QA tasks. We use Naive K-LLMs (one-time retrieval & generation) as the evaluated system. Baichuan-2-13B (Yang et al. 2023) is used as the Generator. BGE (Xiao et al. 2023) is used as the procedural knowledge Retriever. Contriever (Izacard et al. 2021) is used as the descriptive knowledge Retriever. The descriptive and procedural knowledge in Table 7 (Appendix A.4) is used as the knowledge bases.

RQ3: Can knowledge conflict resolution methods further improve the performance of K-LLM systems? To validate this, we evaluate: 1) the CM conflict resolution method: COIECD (Yuan et al. 2024) and CAD (Shi et al. 2024), 2) the IC conflict resolution method: ICL (Dong et al. 2022) and Discriminator (Hong et al. 2023).

4 Experimental Analyses

We report experimental results and analyses as follows:

RQ1: The answer is YES. Table 2 summarizes the evaluation results of procedural knowledge retrieval. Specifically, Tables 2a and 2b detail the performance evaluations of the four retrievers for rule and code knowledge retrieval, respectively. Compared to the two representative dense retrievers (BGE and Qwen3), their enhanced versions with our Self-Induction method demonstrate consistent performance improvements across most datasets, *revealing our method's potential in mitigating semantic gaps for procedural knowledge retrieval.*

Considering that KMatrix has evaluated the retrieval performance of declarative knowledge, we further report the semantic parsing performance of the two integrated Query Parsers in Table 3. They exhibit strong semantic parsing capabilities.

RQ2: Factual and Inferential QA tasks rely on different types of knowledge. Table 4 summarizes the evaluation results. Specifically, Table 4a and Table 4b show descriptive & procedural knowledge enhancement performance on Factual and Inferential QA tasks, respectively. On Factual QA tasks, Compared to answer generation without knowledge, the performance of K-LLMs system with descriptive knowledge enhancement shows a significant improvement. The collabora-

Table 2: Procedural knowledge retrieval performance evaluation

(a) Rule knowledge retrieval performance evaluation (Recall@10)

	Clutrr	Deer	Law	Ulogic	TheoremQA	Clutrr-Fol	Ulogic-Fol	Law-Fol
BGE	4.77%	100%	97.59%	95.06%	67.45%	4.96%	90.72%	92.77%
Self-Induction+BGE	15.08%	100%	98.19%	98.67%	80.36%	13.55%	98.80%	97.59%
Qwen3	3.34%	100%	96.39%	96.63%	81.27%	3.15%	92.41%	93.98%
Self-Induction+Qwen3	13.84%	100%	98.80%	98.80%	84.00%	13.07%	96.99%	98.19%

(b) Code knowledge retrieval performance evaluation (Recall@10)

	Codetrans-Dl	Stackoverflow	Apps	Codefeedback	Codetrans-Contest	Text2sql	Cosqa
BGE	59.44%	87.51%	41.71%	48.81%	69.25%	79.36%	41.20%
Self-Induction+BGE	57.84%	87.70%	48.12%	50.36%	70.83%	92.13%	44.02%
Qwen3	77.08%	94.19%	90.86%	93.47%	93.06%	95.58%	60.95%
Self-Induction+Qwen3	79.41%	94.52%	93.19%	94.61%	93.65%	98.18%	58.51%

Table 3: Semantic parsing performance of Query Parser (EM)

Sparql Parser Evaluation		Sql Parser Evaluation	
Model	EM	Model	EM
WikiSP	75.55%	RESDSQL(t5-3b)	72.64%

Table 4: Descriptive & procedural knowledge enhancement performance evaluation

(a) Descriptive & procedural knowledge enhancement performance evaluation on Factual QA tasks (Acc)

Knowledge	Popqa	Webqa	NQ	Triviaqa	Hotpotqa	2Wikiqa
Without	23.45%	29.53%	13.24%	42.86%	16.65%	23.75%
Declarative	62.26%	37.60%	36.62%	61.70%	27.90%	26.45%
Declarative+Procedural	62.54%	36.96%	35.84%	60.74%	27.20%	26.35%

(b) Descriptive & procedural knowledge enhancement performance evaluation on Inferential QA tasks (Acc)

Knowledge	Clutrr	Deer	Law	Ulogic	Clutrr-Fol	Ulogic-Fol	Law-Fol
Without	25.19%	85.71%	27.71%	11.81%	24.81%	11.81%	27.11%
Declarative	27.96%	95.24%	22.89%	48.80%	28.34%	48.55%	23.49%
Declarative+Procedural	31.97%	95.24%	40.96%	58.43%	32.06%	58.55%	40.96%

tive enhancement performance of descriptive and procedural knowledge does not differ significantly from that of descriptive knowledge enhancement alone, *indicating that Factual QA tasks primarily rely on descriptive knowledge, with a weak dependence on procedural knowledge.* On Inferential QA tasks, Compared to answer generation without knowledge, the K-LLMs system with descriptive knowledge enhancement demonstrates a marked performance improvement, and the collaborative enhancement of descriptive and procedural knowledge further boosts performance, *highlighting that inferential QA tasks require both descriptive and procedural knowledge.*

RQ3: knowledge conflict resolution further improves the performance of K-LLM systems. Table 5 displays the evaluation results. Compared to method without knowledge, the K-LLMs systems with knowledge enhancement demonstrate a significant performance improvement. Our developed CM and IC knowledge conflict resolution methods both yield additional substantial gains, *demonstrating their capability in mitigating hallucinations and improving factual accuracy of LLMs.*

Table 5: Knowledge conflict resolution evaluation

	Method	Acc
CM Conflict Resolution	w/o knowledge	10.00%
	w/ knowledge	56.65%
	+COIECD	63.98%
	+CAD	60.08%
IC Conflict Resolution	w/o knowledge	1.00%
	w/ knowledge	21.00%
	+ICL	34.90%
	+Discriminator	54.00%

5 Conclusions

The paper presents KMatrix-2, an open-source toolkit that supports comprehensive heterogeneous knowledge enhancement for LLMs. Unified knowledge integration and joint knowledge retrieval methods are provided to achieve descriptive and procedural knowledge collaborative enhancement. Meanwhile, systematic knowledge conflict resolution methods are offered for better knowledge integration. We develop a rich selection of components and several typical enhancement patterns to support convenient construction of mainstream K-LLMs systems. Some key research questions are analyzed and comparative performance results demonstrate the capabilities of KMatrix-2.

Limitations

KMatrix-2 currently has some limitations, which we will gradually improve in the future. 1) Although our toolkit supports the construction of Adaptive K-LLMs systems, it lacks comprehensive and systematic support for the currently popular Agentic RAG. This is an area we need to strengthen in our follow-up work. 2) Regarding the productization of our toolkit, we need to do more work.

Acknowledgements

This work was supported by Beijing Natural Science Foundation (L243006) and the National Natural Science Foundation of China (No.62376270).

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection.

Harrison Chase. 2022. Langchain, october 2022. *URL https://github.com/langchain-ai/langchain*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*.

Tong Chen, Hongwei Wang, Sihao Chen, Wenhao Yu, Kaixin Ma, Xinran Zhao, Hongming Zhang, and Dong Yu. 2023. Dense x retrieval: What retrieval granularity should we use? *arXiv preprint arXiv:2312.06648*.

Daixuan Cheng, Shaohan Huang, Junyu Bi, Yuefeng Zhan, Jianfeng Liu, Yujing Wang, Hao Sun, Furu Wei, Denvy Deng, and Qi Zhang. 2023. Uprise: Universal prompt retrieval for improving zero-shot evaluation. *arXiv preprint arXiv:2303.08518*.

Xiang Deng, Ahmed Hassan Awadallah, Christopher Meek, Oleksandr Polozov, Huan Sun, and Matthew Richardson. 2020. Structure-grounded pretraining for text-to-sql. *arXiv preprint arXiv:2010.12773*.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, et al. 2022. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Denis Emelin, Daniele Bonadiman, Sawsan Alqahtani, Yi Zhang, and Saab Mansour. 2022. Injecting domain knowledge in language models for task-oriented dialogue systems. *arXiv preprint arXiv:2212.08120*.

Zhangyin Feng, Xiaocheng Feng, Dezhi Zhao, Maojin Yang, and Bing Qin. 2024. Retrieval-generation synergy augmented large language models. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 11661–11665. IEEE.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Giwon Hong, Jeonghwan Kim, Junmo Kang, Sung-Hyon Myaeng, and Joyce Jiyoung Whang. 2023. Why so gullible? enhancing the robustness of retrieval-augmented models against counterfactual noise. *arXiv preprint arXiv:2305.01579*.

Yasuto Hoshi, Daisuke Miyashita, Youyang Ng, Kento Tatsuno, Yasuhiro Morioka, Osamu Torii, and Jun Deguchi. 2023. Ralle: A framework for developing and evaluating retrieval-augmented large language models. *arXiv preprint arXiv:2308.10633*.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.

Peter Izsak, Moshe Berchansky, Daniel Fleischer, and Ronen Laperdon. 2023. fastrag: Efficient retrieval augmentation and generation framework.

Janis E Jacobs and Scott G Paris. 1987. Children's metacognition about reading: Issues in definition, measurement, and instruction. *Educational psychologist*, 22(3-4):255–278.

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

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. 2023a. Structgpt: A general framework for large language model to reason over structured data. *arXiv preprint arXiv:2305.09645*.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023b. Active retrieval augmented generation. *arXiv preprint arXiv:2305.06983*.

Jiajie Jin, Yutao Zhu, Xinyu Yang, Chenghao Zhang, and Zhicheng Dou. 2024. Flashrag: A modular toolkit for efficient retrieval-augmented generation research. *arXiv preprint arXiv:2405.13576*.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. 2023a. Resdsql: Decoupling schema linking and skeleton parsing for text-to-sql. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13067–13075.

Xiangyang Li, Kuicai Dong, Yi Quan Lee, Wei Xia, Yichun Yin, Hao Zhang, Yong Liu, Yasheng Wang, and Ruiming Tang. 2024. Coir: A comprehensive benchmark for code information retrieval models. *arXiv preprint arXiv:2407.02883*.

Xingxuan Li, Ruochen Zhao, Yew Ken Chia, Bosheng Ding, Lidong Bing, Shafiq Joty, and Soujanya Poria. 2023b. Chain of knowledge: A framework for grounding large language models with structured knowledge bases. *arXiv preprint arXiv:2305.13269*, 3.

Kaixin Ma, Hao Cheng, Xiaodong Liu, Eric Nyberg, and Jianfeng Gao. 2022. Open-domain question answering via chain of reasoning over heterogeneous knowledge. *arXiv preprint arXiv:2210.12338*.

Barlas Oguz, Xilun Chen, Vladimir Karpukhin, Stan Peshterliev, Dmytro Okhonko, Michael Schlichtkrull, Sonal Gupta, Yashar Mehdad, and Scott Yih. 2020. Unik-qa: Unified representations of structured and unstructured knowledge for open-domain question answering. *arXiv preprint arXiv:2012.14610*.

Malte Pietsch, Timo Möller, Bogdan Kostic, Julian Risch, Massimiliano Pippi, Mayank Jobanputra, Sara Zanzottera, Silvano Cerza, Vladimir Blagojevic, Thomas Stadelmann, et al. 2019. Haystack: the end-to-end nlp framework for pragmatic builders. and denny zhou. 2022b. chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2020. Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2010.08191*.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. *arXiv preprint arXiv:2305.15294*.

Weijia Shi, Xiaochuang Han, Mike Lewis, Yulia Tsvetkov, Luke Zettlemoyer, and Wen-tau Yih. 2024. Trusting your evidence: Hallucinate less with context-aware decoding. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 783–791.

Zhaochen Su, Jun Zhang, Xiaoye Qu, Tong Zhu, Yanshu Li, Jiashuo Sun, Juntao Li, Min Zhang, and Yu Cheng. 2024. Conflictbank: A benchmark for evaluating the influence of knowledge conflicts in llm. *arXiv preprint arXiv:2408.12076*.

Wangtao Sun, Chenxiang Zhang, XueYou Zhang, Xuanqing Yu, Ziyang Huang, Pei Chen, Haotian Xu, Shizhu He, Jun Zhao, and Kang Liu. 2024. Beyond instruction following: Evaluating inferential rule following of large language models. *arXiv preprint arXiv:2407.08440*.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10).

Siyuan Wang, Zhongyu Wei, Yejin Choi, and Xiang Ren. 2024. Can llms reason with rules? logic scaffolding for stress-testing and improving llms. *arXiv preprint arXiv:2402.11442*.

Xintao Wang, Qianwen Yang, Yongting Qiu, Jiaqing Liang, Qianyu He, Zhouhong Gu, Yanghua Xiao, and Wei Wang. 2023. Knowledgpt: Enhancing large language models with retrieval and storage access on knowledge bases. *arXiv preprint arXiv:2308.11761*.

Shun Wu, Di Wu, Kun Luo, XueYou Zhang, Jun Zhao, and Kang Liu. 2024. Kmatrix: A flexible heterogeneous knowledge enhancement toolkit for large language model. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 280–290.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighof. 2023. C-pack: Packaged resources to advance general chinese embedding. *arXiv preprint arXiv:2309.07597*.

Jian Xie, Kai Zhang, Jiangjie Chen, Renze Lou, and Yu Su. 2024. Adaptive chameleon or stubborn sloth: Revealing the behavior of large language models in knowledge conflicts. In *The Twelfth International Conference on Learning Representations*.

Rongwu Xu, Zehan Qi, Zhijiang Guo, Cunxiang Wang, Hongru Wang, Yue Zhang, and Wei Xu. 2024. Knowledge conflicts for llms: A survey. *arXiv preprint arXiv:2403.08319*.

Silei Xu, Shicheng Liu, Theo Culhane, Elizaveta Pertseva, Meng-Hsi Wu, Sina J Semnani, and Monica S Lam. 2023. Fine-tuned llms know more, hallucinate less with few-shot sequence-to-sequence semantic parsing over wikidata. *arXiv preprint arXiv:2305.14202*.

Aiyuan Yang, Bin Xiao, Bingning Wang, Borong Zhang, Ce Bian, Chao Yin, Chenxu Lv, Da Pan, Dian Wang, Dong Yan, et al. 2023. Baichuan 2: Open large-scale language models. *arXiv preprint arXiv:2309.10305*.

Xiaowei Yuan, Zhao Yang, Yequan Wang, Shengping Liu, Jun Zhao, and Kang Liu. 2024. Discerning and resolving knowledge conflicts through adaptive decoding with contextual information-entropy constraint. *arXiv preprint arXiv:2402.11893*.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.

Wenxuan Zhou, Sheng Zhang, Hoifung Poon, and Muhao Chen. 2023. Context-faithful prompting for large language models. *arXiv preprint arXiv:2303.11315*.

A Appendix

A.1 A simplified instruction for self-induction method

You are given a Context and a Question. Please write the inferential rule that may help answer the question. A demonstration example is as follows:
Context: Artist Alice is commissioned for a large mural titled "Sunset Dreams" in the city center, involving vibrant colors and abstract shapes to represent the diversity of the community.
Question: Does Alice need to create "Sunset Dreams"?
Rule: Commissioned(Person X, Artwork Y) => NeedToCreate(Person X, Artwork Y)

A.2 The Usage Instructions of KMatrix-2

KMatrix-2 consists of three sections: Knowledge Base Management, System Design and Interaction, and Task Management and Evaluation. *The screencast video* of our toolkit are available here.[5]

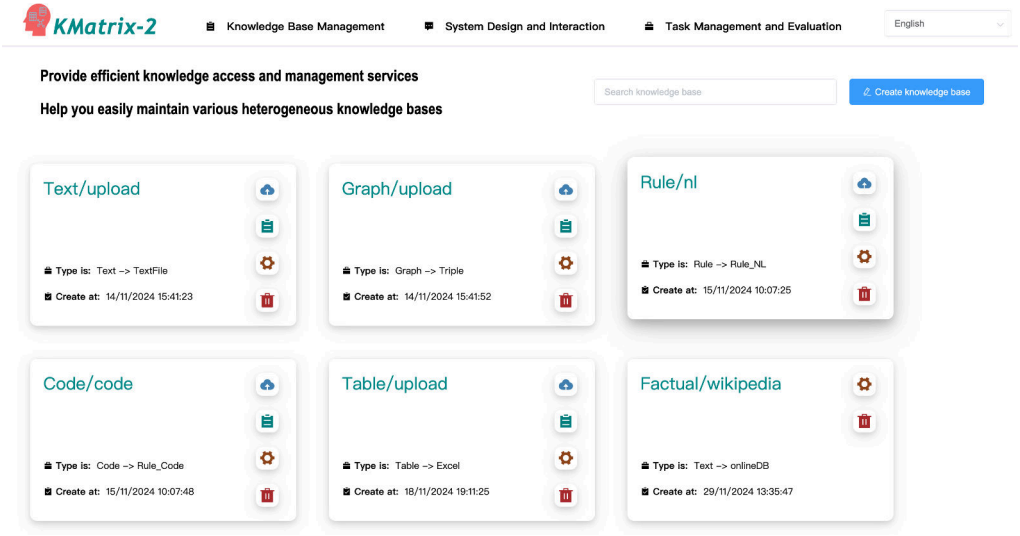
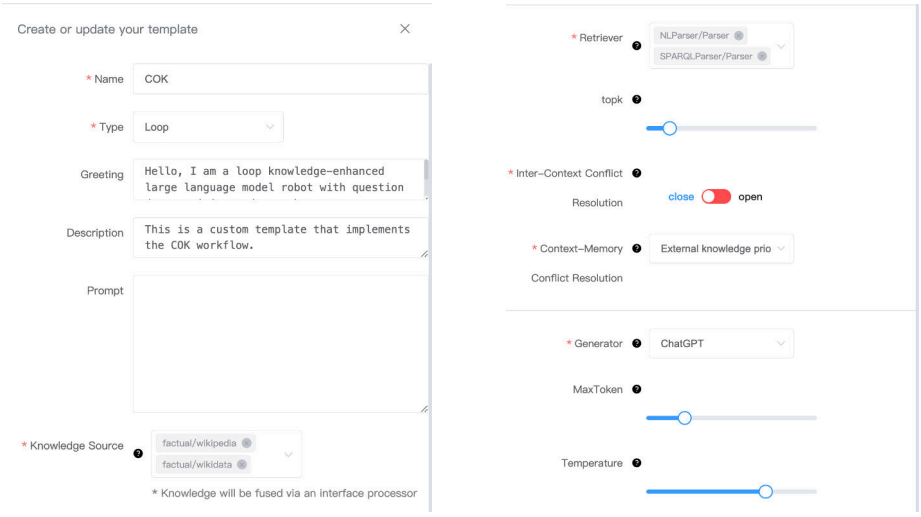


Figure 3: Knowledge Base Management interface: providing efficient heterogeneous knowledge access and management services. We can easily create descriptive and procedural knowledge bases, including Text, Table, Knowledge Graph, Rule and Code. And upload corresponding files (like PDF, WORD, Rule JSON, etc) or databases (like SQLite, Neo4j, GraphDB, etc). All knowledge bases we create will be uniformly managed, and users can easily search and modify the information in the knowledge bases.



[5] https://youtu.be/E7hk-jrM2CY

Figure 4: System Design and Interaction: supporting the construction, deployment, and user interaction of the K-LLMs system. We can construct K-LLMs system by: 1) selecting system enhancement pattern (like Adaptive and Iterative), 2) selecting knowledge source, 3) selecting Retriever and configuring parameters, 4) selecting knowledge conflict resolution strategy, 5) selecting Generator and configuring parameters. After that, we can deploy the K-LLMs system and interact with it. The interface of System Deployment and Interaction is shown in Figure 2.

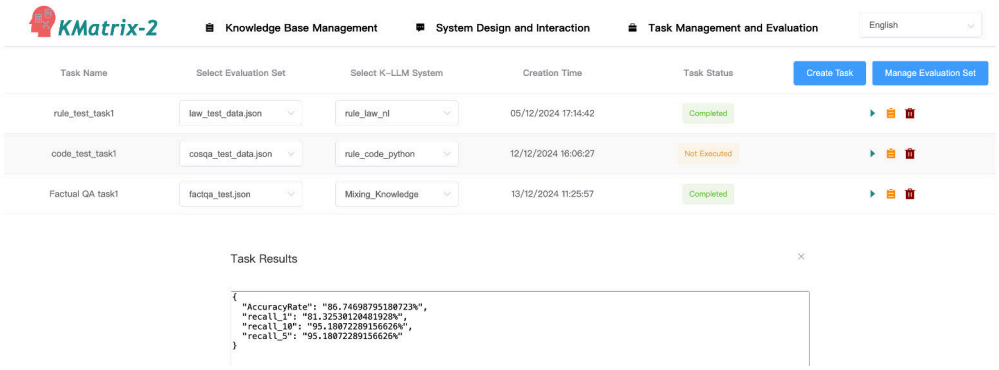


Figure 5: Task Management and Evaluation interface: support for constructing, managing and executing evaluation tasks for the K-LLMs system. We can create evaluation task, and select evaluation dataset & K-LLMs system we create. After that, we can run the evaluation task, and the evaluation results will be presented.

A.3 The Enhancement Patterns of KMatrix-2

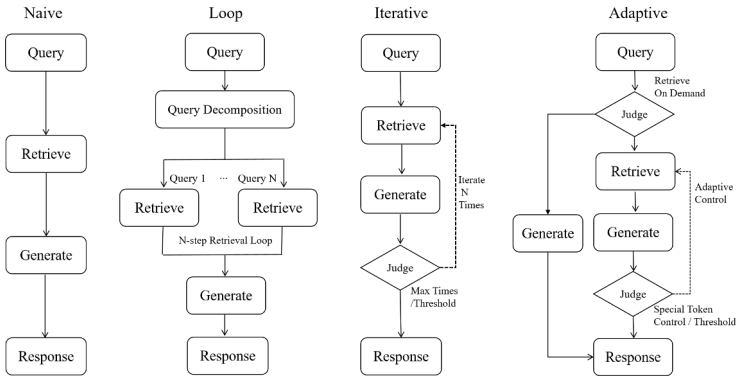


Figure 6: The Enhancement Patterns of KMatrix-2

A.4 Heterogeneous knowledge and datasets used in evaluation

Table 6: Evaluation datasets used by KMatrix-2

Dataset Type	Dataset Name	Dataset Scale
Code Retrieval	COIR	223358
Rule Retrieval	RuleBench	4527
Sparql Parsing	WWQ	454
Sql Parsing	spider_realistic	508
Factual QA	2Wikiqa	12576
	Hotpotqa	7405
	NQ	3610
	Popqa	1399
	Triviaqa	7313
	Webqa	2032
Inferential QA	RuleBench	4527
IC Conflict Resolution	ConflictBank (subset)	1000
CM Conflict Resolution	ConflictQA	8027

Table 7: Heterogeneous knowledge used in evaluation

Knowledge Type	Knowledge Name	Knowledge Scale
Descriptive Knowledge	Wikipedia	21000k
	Wikidata	5790k
	Wikitable	6860k
Procedural knowledge	NL Rule (RuleBench)	2462
	FOL Rule (RuleBench)	2065