KMatrix: A Flexible Heterogeneous Knowledge Enhancement Toolkit for Large Language Model

Shun Wu[1], **Di Wu**[1], **Kun Luo**[1,2], **XueYou Zhang**[1], **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
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences
[3]Shanghai Artificial Intelligence Laboratory
{shun.wu, jzhao, kliu}@nlpr.ia.ac.cn
{di.wu, xueyou.zhang}@ia.ac.cn, {luokun695}@gmail.com

Abstract

Knowledge-Enhanced Large Language Models (K-LLMs) system enhances Large Language Models (LLMs) abilities using external knowledge. Existing K-LLMs toolkits mainly focus on free-textual knowledge, lacking support for heterogeneous knowledge like tables and knowledge graphs, and fall short in comprehensive datasets, models, and user-friendly experience. To address this gap, we introduce KMatrix: a flexible heterogeneous knowledge enhancement toolkit for LLMs including verbalizing-retrieval and parsing-query methods. Our modularity and control-logic flow diagram design flexibly supports the entire lifecycle of various complex K-LLMs systems, including training, evaluation, and deployment. To assist K-LLMs system research, a series of related knowledge, datasets, and models are integrated into our toolkit, along with performance analyses of K-LLMs systems enhanced by different types of knowledge. Using our toolkit, developers can rapidly build, evaluate, and deploy their own K-LLMs systems. Our toolkit and resources are available at here.[1]

1 Introduction

Knowledge-Enhanced Large Language Models (K-LLMs) system uses external knowledge to enhance the capabilities of Large Language Models (LLMs) (Hu et al. (2023)), which alleviates the issues of hallucination and weak reasoning abilities for knowledge-intensive natural language processing tasks (Bang et al. (2023), Sasaki et al. (2024), Lewis et al. (2020)). Recently, K-LLMs have become a popular research topic and extensive works have been conducted from various dimensions such as knowledge, models, and enhancement methods (Gao et al. (2023)).

Early K-LLMs works primarily focused on free-textual knowledge enhancement (Karpukhin et al. (2020), Qu et al. (2020)), which led to the emergence of the Retrieval-Augmented Generation (RAG) research branch. Recent studies have explored methods for jointly enhancing LLMs with heterogeneous knowledge (like tables, knowledge graphs, etc) using unified retrieval (Oguz et al. (2020), Ma et al. (2022)) or selective query (Jiang et al. (2023), Li et al. (2023)). Meanwhile, increasing attention is being directed towards adaptive enhancement methods research (Wang et al. (2023b), Asai et al. (2023)), which autonomously control the interaction between generation and retrieval (Gao et al. (2023)) to achieve better performance. Moreover, with the development of K-LLMs, there is a need for an easy-to-use toolkit to flexibly implement K-LLMs works and compare different approaches under the same conditions. In recent years, many K-LLMs related toolkits (Chase (2022), Hoshi et al. (2023), Pietsch et al. (2019), Izsak et al. (2023)) have emerged, but they still have the following shortcomings: 1) Lacking support for joint enhancement with heterogeneous knowledge sources. The existing representative K-LLMs toolkits (Chase (2022), Hoshi et al. (2023), Jin et al. (2024)) predominantly focus on textual knowledge enhancement. 2) Lacking systematic support for various adaptive enhancement methods. Coze[2] and RALLE (Hoshi et al. (2023)) enabled the construction of naive K-LLMs (retrieval and generation) by selecting components, but they lacked support for building complex adaptive K-LLMs. FlashRAG (Jin et al. (2024)) implemented adaptive enhancement by simply integrating code of some existing K-LLMs works, lacking systematic integration of adaptive enhancement methods from different dimensions, like retrieval timing determination and retrieval source selection. 3) Not highly customizable or easily combinable, and lacking comprehensive support for training, evaluation, and deployment of K-LLMs systems. LangChain (Chase

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

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing
System Demonstrations, pages 280–290
November 12-16, 2024

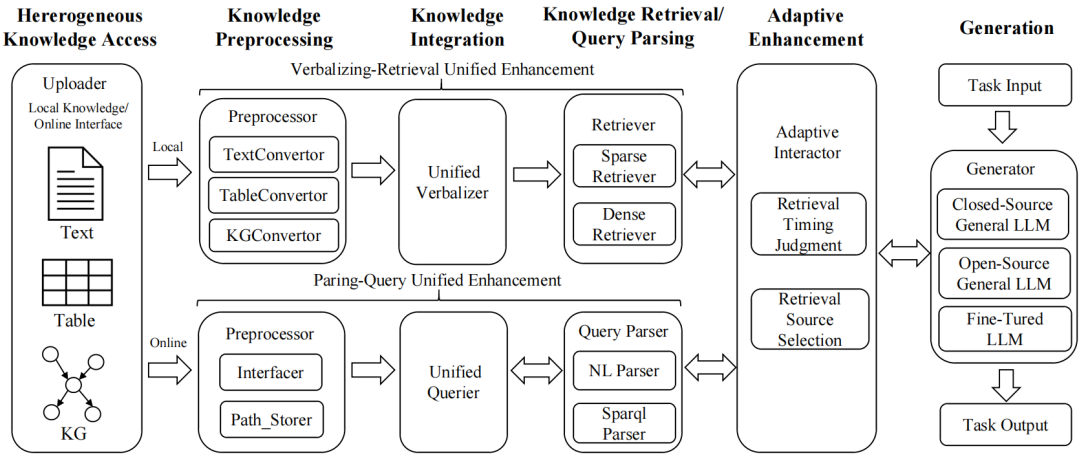

Figure 1: A overview framework of the KMatrix toolkit

(2022)) and Haystack (Pietsch et al. (2019)) are two fundamental K-LLMs toolkits which lacked integration of existing representative K-LLMs works, and did not provide sufficient flexibility for customization. FastRAG (Izsak et al. (2023)) and FlashRAG (Jin et al. (2024)) utilized customizable component design and integrate extensive existing datasets, knowledge, and models. However, they define component relations using hard-coding methods, which is not easily combinable. Comparison of existing representative K-LLMs toolkits is shown in Table 1.

To address the aforementioned shortcomings, we introduce KMatrix: a flexible heterogeneous knowledge enhancement toolkit for LLMs. Our toolkit uses both verbalizing-retrieval (Ma et al. (2022)) and parsing-query (Jiang et al. (2023)) methods to support unified enhancement of heterogeneous knowledge (like free-textual knowledge, tables, knowledge graphs, etc). And we systematically integrate adaptive enhancement methods from two aspects: retrieval timing judgment (Asai et al. (2023)) and knowledge source selection (Li et al. (2023)). To achieve high customizability and easy combinability, we deploy modular component definition and control-logic flow diagram design to flexibly construct components and their relations. In summary, our main contributions are:

1. We propose a K-LLMs toolkit that supports unified enhancement of heterogeneous knowledge to enhance the capabilities of LLMs.

2. KMatrix offers comprehensive adaptive enhancement methods including retrieval timing judgment and knowledge source selection.

3. We design modular component and control-logic flow diagram using graphical patterns, and integrate 22 training/evaluation datasets and 11 representative knowledge bases. This allows one-click support for training, evaluation, and deployment in K-LLMs system lifecycle.

4. Using our constructed toolkit, we implement representative K-LLMs works and provide comparative evaluation results on multiple datasets. Extensive experimental results show that KMatrix can effectively support flexible implementation, multi-dimensional evaluation, and improvement of K-LLMs system.

2 KMatrix Toolkit

As shown in Figure 1, our toolkit contains seven stages to complete knowledge-enhanced generation task. Knowledge Access, Knowledge Preprocessing, and Knowledge Integration are respectively used for the access, preprocessing, and unified fusion of heterogeneous knowledge. Knowledge Retrieval retrieves knowledge from a unified textual knowledge base and Query Parsing generates query statements for a unified querier. Adaptive Enhancement autonomously controls the interaction between generation and retrieval/query. Generation stage receives task inputs and generates outputs with knowledge enhancement. All stages are implemented based on our modular component definitions. Meanwhile, we design a control-logic flow diagram to combine components. Next, we will introduce the seven stages of KMatrix and present our modular design approach & toolkit usage.

2.1 Heterogeneous Knowledge Access

KMatrix designs a *Knowledge Uploader* component to support the access of heterogeneous knowledge, which contains textual knowledge (like Word,

Toolkit	Knowledge Type	Dataset/Model	Support Stage	Complex System	Customization	Usability
Haystack	Text	Few	Deployment	Good	Poor	Good
Langchain	Text	Few	Evaluation Deployment	Good	Poor	Fair
RALLE	Text	Moderate	Deployment	Poor	Fair	Fair
Coze	Text	Few	Deployment	Poor	Fair	Good
GraphRAG	Text	Moderate	Evaluation Deployment	Good	Fair	Fair
FastRAG	Text	Moderate	Evaluation Deployment	Good	Fair	Good
FlashRAG	Text	Rich	Evaluation Deployment	Good	Good	Fair
KMatrix	Text Table Knowledge Graph	Rich	Training Evaluation Deployment	Good	Good	Good

Table 1: Comparison of existing representative K-LLMs toolkits. Knowledge Type refers to knowledge types supported by toolkits. Dataset/Model refers to the number of specific datasets, knowledge and models accessed by toolkits. Support Stage refers to the stages of K-LLMs system construction supported by toolkits: Training, Evaluation, and Deployment, indicating support for system training, evaluation, and deployment, respectively. Complex System refers to toolkit capability support for the construction of complex K-LLMs systems. Customization refers to the flexibility of user-defined modules or systems. Usability refers to the ease of use of toolkits.

PDF, QA pairs, search engine results, and encyclopedias), table knowledge (like Excel and relational databases) and knowledge graph (in the form of triples). Meanwhile, KMatrix supports two types of knowledge access: local knowledge and online interface, representing local knowledge data and online knowledge query interfaces, respectively.

2.2 Knowledge Preprocessing & Integration

KMatrix implements the unified enhancement of heterogeneous knowledge using two methods: verbalizing-retrieval and parsing-query. Verbalizing-retrieval method converts different types of local knowledge (such as tables and knowledge graphs) into unified text fragments (Ma et al. (2021)), which will be retrieved by a *Retriever* uniformly. Parsing-query method integrates different types of knowledge interfaces into a *Unified Querier*, which receives queries generated by a *Query Parser* (Li et al. (2023)) and returns the query results. The flow diagram of the above two methods can be found in Appendix A.1.

For verbalizing-retrieval method, we design a *Knowledge Preprocessor* component containing three types of *Convertors* to implement format processing of local heterogeneous knowledge. We develop a *Unified Verbalizer* component to convert various types of local heterogeneous knowledge(such as text, tables and knowledge graphs) into unified text for local knowledge integration, which is trained based on the model framework in Ma et al. (2021).

For parsing-query method, we develop a *Knowledge Preprocessor* component containing *Interfacer* and *Path_storer* to support online heterogeneous knowledge interface design and standardization. We design a *Unified Querier* component to flexibly incorporate different types of knowledge query interfaces (like Wikipedia[3], Wikidata[4]) for online knowledge integration.

2.3 Knowledge Retrieval/Query Parsing

KMatrix retrieves knowledge from a unified textual knowledge base converted by a *Unified Verbalizer*, which is implemented by a *Retriever* component. For sparse retriever, we integrate BM25 and TF-IDF, using the rank-bm25[5] and scikit-learn[6] library. For dense retriever, we integrate three BERT-based retrieval models, including Contriever (Izacard et al. (2021)), DPR (Karpukhin et al. (2020)), and BGE (Xiao et al. (2023)), as well as a LLM-based retrieval model: E5-7b (Wang et al. (2023a)).

We also design a *Query Parser* component to implement parsing process, which receives query contents and generates query statements specifically tailored for the *Unified Querier* to obtain queried knowledge. KMatrix integrates two types of *Query Parser* components to support diverse query parsing tasks: 1) *NL Parser* : A natural language query generator based on ChatGPT[7], 2) *Sparql Parser*: A

[3] https://www.wikipedia.org/
[4] https://query.wikidata.org/
[5] https://pypi.org/project/rank-bm25/
[6] https://pypi.org/project/scikit-learn/
[7] https://openai.com/index/

SPARQL query generator built by Xu et al. (2023).

2.4 Adaptive Enhancement

Adaptive Enhancement autonomously controls the interaction between generation and retrieval/query. KMatrix integrates existing adaptive enhancement methods from two aspects: retrieval timing judgment and knowledge source selection.

Retrieval Timing Judgment: judging whether knowledge retrieval is necessary and how many times to retrieve knowledge. KMatrix achieves this goal by: 1) integrating the special tokens control method based on Self-RAG (Asai et al. (2023)), which uses LLM-generated special tokens to control retrieval timing. *For example, [Retrieval] represents continuing to retrieve, while [No Retrieval] represents stopping the retrieval process.* 2) integrating the self-consistency method (Wang et al. (2022)), which judges retrieval is needed when the consistency score of multiple responses to the question falls below a threshold.

Knowledge Source Selection: adaptively selecting which knowledge source to retrieve. We integrate two methods to achieve this target. 1) Knowledge sources are automatically selected by retrieving the unified textual knowledge base verbalized across multiple knowledge sources (Ma et al. (2021)). 2) We also integrate an active knowledge source selection method, which is inspired by COK (Li et al. (2023)). It deploys LLMs to select knowledge sources relevant to the question using demonstration learning based on correlation examples between questions and knowledge sources.

2.5 Generation

To meet the needs of different K-LLMs generation scenarios, KMatrix integrates: 1) a representative closed-source general *Generator*: ChatGPT, 2) two open-source general *Generators*: Baichuan-2-7b (Yang et al. (2023)) and Llama-2-7b (Touvron et al. (2023)), and 3) a retrieval instructions-enhanced *Generator*: SelfRAG (Asai et al. (2023)) for better adaptive enhancement.

2.6 Modular Design Approach & Toolkit Usage

Modular Design Approach: KMatrix deploys modular design approach to construct K-LLMs systems using two stages: modular component definition and control-logic flow diagram design.

Modular component definition: KMatrix component is an functional unit of K-LLMs system. We unify datasets, knowledge, and models involved in K-LLMs as components. To implement the processes in Figure 1, KMatrix defines 16 types of components, like *Retriever*, *Query Parser*, *Generator*, etc. And users can define their own components according to predefined formats.

Control-logic flow diagram design: We develop a control-logic flow diagram design method based on easy-flow[8] and Haystack (Pietsch et al. (2019)) framework to flexibly organize components for K-LLMs system construction. Flow diagram example can be found in Appendix A.2. For K-LLMs system with complex process (including multifarious arithmetic operations and logical judgments), we can use control flow diagram to design system process using Python programming. For K-LLMs system with concise process (like linear, branching, looping, and conditional structures), we can employ logic flow diagram to directly connect components with edges. By jointly using control and logic flow diagram, KMatrix flexibly supports common K-LLMs patterns using naive, iterative, and adaptive enhancement methods (Gao et al. (2023)).

Toolkit Usage: Users can select or customize components, and construct K-LLMs systems using control-logic flow diagram. Appendix A.3 shows K-LLMs training, evaluation and deployment flow diagram illustration. The K-LLMs system deployment interface with multiple knowledge bases and multiple queries is shown in figure 2. The left side of the interface displays system details, including system components, knowledge interfaces and query methods. The middle section contains the question and answer box. The right side shows the intermediate chains of system execution, illustrating multiple queries and corrections steps to generate the correct answer.

3 Experimental Settings

In this section, we evaluate the performance of K-LLMs constructed by KMatrix to demonstrate the entire lifecycle capabilities of our toolkit.

3.1 Knowledge and Datesets

KMatrix designs two ways of knowledge access: local knowledge and online interface. As shown in Table 2, for local knowledge, we integrate public Wikipedia (Chen et al. (2017), textual knowl-

introducing-chatgpt-and-whisper-apis/

[8] https://gitee.com/xiaoka2017/easy-flow

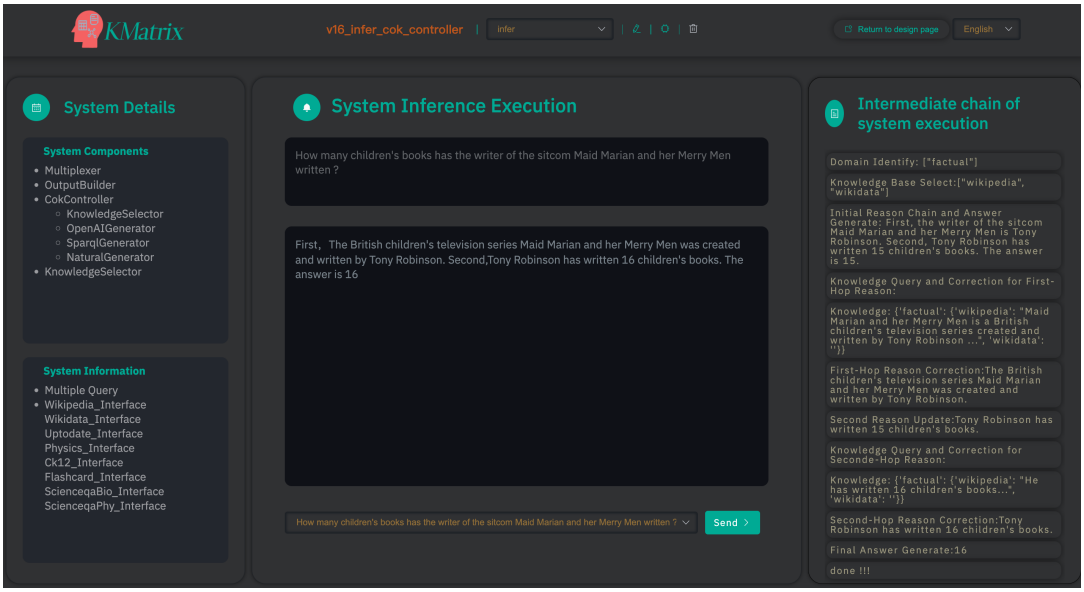

Figure 2: Deployment interface of KMatrix toolkit

edge), Wikidata (Vrandečić and Krötzsch (2014), knowledge graph) and Wikitable (Ma et al. (2022), table knowledge). For online interface, we integrate two general knowledge interfaces (Wikipedia and Wikidata query APIs) and six domain knowledge interfaces (APIs including Uptodate, CK12, etc). Details of online interfaces can be found in Appendix A.4.

Knowledge Access Way	**Knowledge Name**	**Knowledge Scale**
local knowledge	Wikipedia	21000k
	Wikidata	5790k
	Wikitable	6860k
online interface	Wikipedia	/
	Wikidata	/
	Uptodate	/
	Flashcard	33553
	BioScienceqa	1062
	CK12	/
	PhyScienceqa	780
	Physicsclassroom	/

Table 2: Knowledge components integrated by KMatrix

As shown in Table 3, KMatrix provides three classes of datasets to support evaluation of K-LLMs system. We provide RETRIEVE_EVAL to evaluate knowledge access performance of *Retriever* components, which contains eight retrieval datasets from the MTEB[9] benchmark. We provides ODQA_EVAL and ODQA_EVAL_Simplified to evaluate knowledge enhancement performance of K-LLMs system under two ways of knowledge access: local knowledge and online interface respectively. ODQA_EVAL contains six open domain question answering (OODA) datasets: 2Wikiqa (Ho et al. (2020)), HotpotQA (Yang et al. (2018)), NQ (Kwiatkowski et al. (2019)), PopQA (Mallen et al. (2022)), TriviaQA (Joshi et al. (2017)), and WebQA (Berant et al. (2013)). ODQA_EVAL_Simplified contains four simplified ODQA datasets similar to COK (Li et al. (2023)).

Dataset Class	Dataset Name	Dataset Scale
RETRIEVE_EVAL	MSMARCO	510k
	NFCorpus	3237
	NQ	3452
	Quora	15k
	ArguAna	1401
	FiQA2018	6648
	HotpotQA	97852
	SciFact	1109
ODQA_EVAL	2Wikiqa	12576
	Hotpotqa	7405
	NQ	3610
	Popqa	1399
	Triviaqa	7313
	Webqa	2032
ODQA_EVAL_Simplified	Hotpotqa	308
	Medmcqa	146
	MMLU_bio	454
	MMLU_phy	253

Table 3: Evaluation Datasets privided by KMatrix

3.2 Task Settings

To evaluate K-LLMs systems constructed by our toolkit, two task types are employed as follow: *Knowledge access performance evaluation*: we use RETRIEVE_EVAL dataset to evaluate three BERT-based Retrievers, including Contriever (Izacard et al. (2021)), DPR (Karpukhin et al. (2020)), and BGE (Xiao et al. (2023)), as well as a LLM-

[9] https://github.com/embeddings-benchmark/mteb

	ArguAna		FiQA2018		HotpotQA		MSMARCO	
	map@100	r@100	map@100	r@100	map@100	r@100	map@100	r@100
BERT	6.87%	34.48%	0.04%	0.71%	0.07%	0.5%	0.0%	0.02%
Contriever	24.59%	97.36%	27.38%	65.25%	55.27%	77.76%	21.90%	25.97%
DPR	21.33%	89.94%	11.75%	38.48%	31.25%	57.83%	16.00%	58.13%
BGE	28.4%	96.79%	37.55%	75.42%	48.58%	64.89%	36.28%	88.73%
E5-7b	30.50%	99.22%	41.80%	79.52%	39.04%	71.03%	21.99%	78.27%
	NFCorpus		NQ		Quora		SciFact	
	map@100	r@100	map@100	r@100	map@100	r@100	map@100	r@100
BERT	0.28%	3.22%	0.03%	0.30%	41.26%	67.85%	1.56%	14.26%
Contriever	15.33%	29.93%	43.23%	92.71%	83.06%	99.35%	62.88%	94.20%
DPR	6.79%	17.90%	22.29%	73.00%	78.47%	97.78%	29.95%	70.23%
BGE	18.00%	33.94%	44.66%	93.39%	86.15%	99.70%	69.04%	97.17%
E5-7b	11.42%	27.19%	10.28%	41.22%	85.57%	99.65%	70.40%	96.00%

Table 4: Comparative knowledge access performance of Retrievers

Methods	Knowledge	PopQA	TriviaqaQA	NQ	Hotpotqa	2Wikiqa	WebQA
Naive-GEN	Without	14.44%	35.00%	8.53%	11.45%	17.57%	17.03%
Naive-RAG	Wikipedia (Text)	27.51%	54.63%	33.77%	20.39%	22.07%	31.74%
	Wikipedia (Text) + Wikidata (KG)	42.82%	54.18%	33.68%	20.73%	23.19%	31.10%
	Wikipedia (Text) + Wikidata (KG) + Wikitable (Table)	42.89%	54.68%	34.13%	20.47%	23.43%	31.05%
Interleave	Wikipedia (Text)	25.80%	39.6%	24.96%	14.7%	18.03%	22.74%
	Wikipedia (Text) + Wikidata (KG)	41.03%	47.12%	25.01%	16.38%	18.26%	23.23%
	Wikipedia (Text) + Wikidata (KG) + Wikitable (Table)	41.17%	46.27%	25.43%	16.22%	22.1%	23.47%
Self-RAG	Wikipedia (Text)	41.95%	58.38%	29.28%	25.80%	29.34%	34.69%
	Wikipedia (Text) + Wikidata (KG)	61.37%	58.23%	28.92%	25.91%	29.99%	34.30%
	Wikipedia (Text) + Wikidata (KG) + Wikitable (Table)	61.37%	58.57%	29.25%	25.71%	30.12%	34.84%

Table 5: Single vs. multi-knowledge bases enhancement evaluation using local knowledge access way

Methods	Knowledge	Factual Domain	Medical Domain	Physics Domain	Biology Domain
		Hotpotqa	Medmcqa	MMLU_phy	MMLU_bio
COT	Without	37.99%	40.41%	45.85%	78.63%
COK-DE Selective Query	Four domains, eight knowledge interfaces (Text, KG, Table)	40.58%	46.58%	50.2%	78.63%
COK-DE Fixed Query		38.96%	44.52%	49.8%	77.97%

Table 6: Single vs. multi-knowledge bases enhancement evaluation using online interface knowledge access way

based Retriever: E5-7b (Wang et al. (2023a)) using MAP@100 and Recall@100 metrics.

Single vs. Multi-knowledge bases enhancement evaluation: 1) We use ODQA_EVAL dataset to evaluate K-LLMs systems performance using single vs. multi-knowledge bases enhancement under local knowledge access way. We compare four K-LLMs systems: Naive-GEN(answer generation without knowledge), Naive-RAG(naive K-LLMs), Interleave (Shao et al. (2023), iterative K-LLMs) and Self-RAG (Asai et al. (2023), adaptive K-LLMs). We employ the local knowledge shown in Table 2 as heterogeneous knowledge bases, and choose Contriever (Izacard et al. (2021)) as Retriever. The number of retrieval is uniformly set to 3. We use LLaMA2-7b (Touvron et al. (2023)) as Generator except Self-RAG, which uses a retrieval-instructed Generator. 2) We use ODQA_EVAL_Simplified dataset to evaluate K-LLMs systems performance using single vs. multi-knowledge bases enhancement under online interface knowledge access way. We compare two K-LLMs systems: COT ((Wei et al. (2022)), answer generation without knowledge) and COK-DE (K-LLMs system actively querying multiple knowledge interfaces, with main idea derived from COK (Li et al. (2023))). We employ the online interfaces shown in Table 2 as knowledge sources, which contains two general knowledge interfaces and six domain knowledge interfaces. We choose ChatGPT as Generator and adopt accuracy as metric for performance of K-LLMs systems.

4 Experimental Results

We report experimental results from two aspects:

Knowledge access performance evaluation: Table 4 shows the knowledge access performance of five Retrievers. Compared to BERT model, the three improved Retrievers based on BERT, namely Contriever, DPR, and BGE, have significant performance advantages. Among them, BGE has a significant advantage on most datasets. The E5-7b Retriever based on LLM achieves best performance on the vast majority of datasets, demonstrating the research potential of LLM-based Retrievers.

Single vs. Multi-knowledge bases enhancement evaluation: Table 5 shows single vs. multi-knowledge bases enhancement evaluation results using local knowledge access way. From the perspective of quantity of knowledge base types, compared to a single text knowledge, increasing the types of knowledge bases usually results in better performance. However, the joint enhancement performance of text, tables, and knowledge graphs is inferior to that of tables and knowledge graphs on a few datasets, which may be caused by noise of tables. The experimental results confirm the conclusion that joint enhancement using multi-knowledge bases can improve the performance of K-LLMs systems. From the perspective of enhancement methods, compared to Naive-GEN without knowledge enhancement, the three methods that use knowledge enhancement achieve significant performance improvements. Meanwhile, compared to Interactive (iterative K-LLMs), Naive-RAG has a performance advantage, and the reason may be that iterative retrieval is not suitable for factual question answer tasks. Self-RAG (adaptive K-LLMs) achieve best performance on most datasets, demonstrating enormous potential of adaptive K-LLMs research.

Table 6 shows single vs. multi-knowledge bases enhancement evaluation results using online interface access way. Compared to the COT method without knowledge enhancement, COK-DE with active knowledge query achieves performance improvements on most datasets, highlighting the importance of external knowledge enhancemant. Meanwhile, for the COK-DE method, we compare two experimental settings: selective query across multiple-domain knowledge interfaces vs. fixed query on single-domain knowledge interface. We find that allow LLMs to autonomously select knowledge can achieve better performance, which indicates that solutions of most tasks require integration of multi-domain knowledge.

5 Conclusions

We introduce KMatrix tootkit to facilitate the construction of adaptive heterogeneous K-LLMs system, which enables one-click support for training, evaluation, and deployment procedures. Meanwhile, we integrate a rich collection of representative K-LLMs knowledge, datasets, and methods, and provide performance analysis of heterogeneous knowledge enhancement, which can offer assistance for future works. Overall, KMatrix is particularly useful for K-LLMs practitioners without extensive expertise, and we hope KMatrix will contribute to the development of K-LLMs.

Limitations

KMatrix currently has some limitations, which we will gradually improve in the future. 1) Although we have integrated several representative Retriever components and achieved relatively good retrieval accuracy, the efficiency is low due to the excessively large knowledge base. We need to specifically optimize the performance of the retriever to improve retrieval speed. 2) We have found that the Wikitable knowledge integrated into our toolkit contains lots of noise, which directly affects the performance of knowledge enhancement. Next, we will conduct knowledge denoising. 3) Adaptive K-LLMs have become a hot research topic and a large number of new methods are being proposed. In the future, KMatrix will continue to integrate more adaptive K-LLMs methods.

Acknowledgements

This work was supported by the National Key R&D Program of China (No. 2022ZD0160503) and Beijing Natural Science Foundation (L243006). This work was also sponsored by CCF-BaiChuan-Ebtech Foundation Model Fund.

References

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

Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, et al. 2023. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *arXiv preprint arXiv:2302.04023*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1533–1544.

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*.

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*.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060*.

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*.

Linmei Hu, Zeyi Liu, Ziwang Zhao, Lei Hou, Liqiang Nie, and Juanzi Li. 2023. A survey of knowledge enhanced pre-trained language models. *IEEE Transactions on Knowledge and Data Engineering*.

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.

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

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*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

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*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Xingxuan Li, Ruochen Zhao, Yew Ken Chia, Bosheng Ding, Lidong Bing, Shafiq Joty, and Soujanya Poria. 2023. 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. 2021. Open domain question answering with a unified knowledge interface. *arXiv preprint arXiv:2110.08417*.

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*.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. *arXiv preprint arXiv:2212.10511*.

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*.

Miyu Sasaki, Natsumi Watanabe, and Tsukihito Komanaka. 2024. Enhancing contextual understanding of mistral llm with external knowledge bases.

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*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

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

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2023a. Improving text embeddings with large language models. *arXiv preprint arXiv:2401.00368*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Yile Wang, Peng Li, Maosong Sun, and Yang Liu. 2023b. Self-knowledge guided retrieval augmentation for large language models. *arXiv preprint arXiv:2310.05002*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

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

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*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

A Appendix

A.1 The Unified Enhancement of Heterogeneous Knowledge

A.1.1 Verbalizing-Retrieval Method

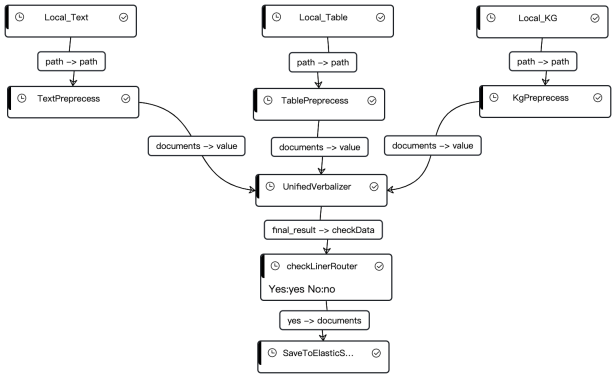

Figure 3: Verbalizing-Retrieval Method Flow Diagram. We develop a *Unified Verbalizer* component to convert various types of local heterogeneous knowledge (such as text, tables and knowledge graphs) into unified text fragments for local knowledge integration.

A.1.2 Parsing-Query Method

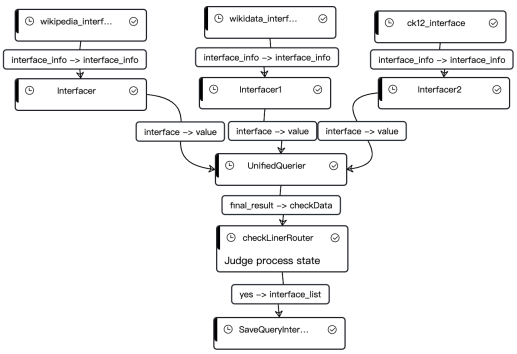

Figure 4: Parsing-Query Method Flow Diagram. We design a *Unified Querier* component to flexibly incorporate different types of knowledge query interfaces (like Wikipedia, Wikidata) for online knowledge integration.

A.2 Control-Logic Flow Diagram Example

A.2.1 Control Flow Diagram Example

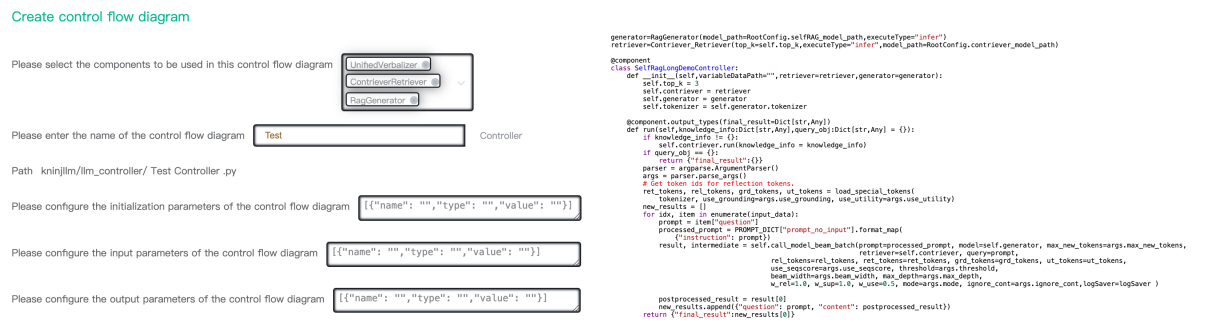

Figure 5: Control Flow Diagram Example. For K-LLMs system with complex process (including multifarious arithmetic operations and logical judgments), users can employ control flow diagram to design system process, which contains three steps: selecting components, configuring components parameters, as shown in left side, and designing system logics using Python programming, as shown in right side.

A.2.2 Logic Flow Diagram Example

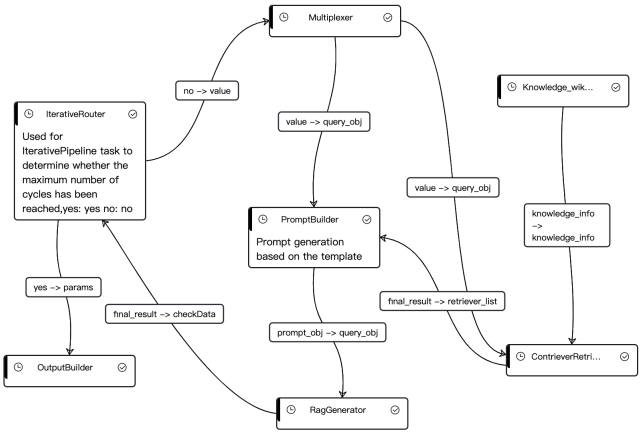

Figure 6: Logic Flow Diagram Example. For K-LLMs system with concise process (like linear, branching, looping, and conditional structures), Users can employ logic flow diagram to directly connect components with edges for K-LLMs system construction, which can achieve the transfer of data from task input to output on the flow diagram.

A.3 Toolkit Usage: Training, Evaluation, Deployment

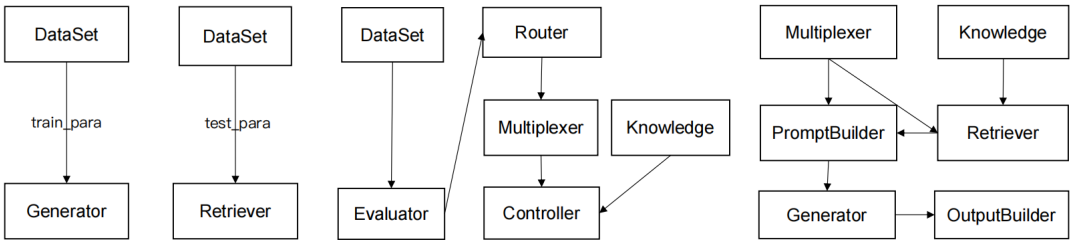

Figure 7: Toolkit Usage: Training, Evaluation, Deployment. For component training and evaluation, users can simply connect the Dataset component with the component to be trained/evaluated. For end-to-end evaluation of the K-LLMs system, users can employ the Evaluator component to connect Dataset component with K-LLMs system, and the Evaluator component will manage evaluation process. For K-LLMs system deployment, users can map the task inputs to the Multiplexer, and connect the task outputs to the OutputBuilder on the basis of original system flow diagram. After constructing system flow diagram, you can run it. Additonal details are available in our toolkit documentation.

A.4 Online Interfaces Integration

KMatrix integrates a total of eight knowledge query interfaces, which contain two general knowledge interfaces: Wikipedia[10] (textual knowledge interface) and Wikidata[11] (knowledge graph interface), as well as six domain textual knowledge interfaces: Uptodate[12], Flashcard[13], BioScienceQA[14], CK12[15], PhyScienceQA[16], and PhysicsClassroom[17].

A.5 The Screencast Video of KMatrix

The screencast video of our toolkit are available at here[18].

[10] https://www.wikipedia.org/
[11] https://query.wikidata.org/
[12] https://www.wolterskluwer.com/en/solutions/uptodate
[13] https://geekymedics.com/medicine-flashcard-collection/
[14] https://huggingface.co/datasets/veggiebird/biology-scienceqa
[15] https://www.ck12.org/book/ck-12-biology/
[16] https://huggingface.co/datasets/veggiebird/physics-scienceqa
[17] https://www.physicsclassroom.com/
[18] https://youtu.be/VL-zY2pphwI