MobileKGQA: On-Device KGQA System on Dynamic Mobile Environments

Junyong Ahn

Seoul National University
Data Science & Artificial Intelligence Laboratory

DSAIL
Data Science & AI Laboratory

Motivation of the Study

- **Knowledge graph question answering (KGQA) system** can provide significant benefits when integrated into LLMs by improving reasoning capabilities and reducing hallucinations.

Motivation of the Study

- **Knowledge graph question answering (KGQA) system** can provide significant benefits when integrated into LLMs on mobile devices by improving reasoning capabilities and reducing hallucinations.

- However, current systems are not suitable for mobile deployment.
  - **heavy computational resources**
  - **struggle to handle distribution shifts** caused by evolving user data

Motivation of the Study

- **Knowledge graph question answering (KGQA) system** can provide significant benefits when integrated into LLMs on mobile devices by improving reasoning capabilities and reducing hallucinations.

- However, current systems are not suitable for mobile deployment.
  - **heavy computational resources**
  - **struggle to handle distribution shifts** caused by evolving user data

- **proposed framework: MobileKGQA**
  First on-device KGQA framework enabling efficient computation and distribution shift adaptation !

Limitations of Previous Methods

category	model	computation		adaptability
		latency	tuning parameters	
w/o LLM	RnG-KBQA (Ye et al., 2022)	**low**	443M	×
	DecAF (Yu et al., 2023)	**low**	848M	×
w/ LLM (finetuned)	RoG (Luo et al., 2024c)	**low**	7B	×
	ChatKBQA (Luo et al., 2024a)	**low**	7B	×
w/ LLM (frozen)	KB-BINDER (Li et al., 2023a)	high	-	○
	ToG (Sun et al., 2024)	high	-	○
	MobileKGQA	**low**	**0.136M**	○

Limitations of Previous Methods

category	model	computation		adaptability
		latency	tuning parameters	
w/o LLM	RnG-KBQA (Ye et al., 2022)	**low**	443M	×
	DecAF (Yu et al., 2023)	**low**	848M	×
w/ LLM (finetuned)	RoG (Luo et al., 2024c)	**low**	7B	×
	ChatKBQA (Luo et al., 2024a)	**low**	7B	×
w/ LLM (frozen)	KB-BINDER (Li et al., 2023a)	high	-	○
	ToG (Sun et al., 2024)	high	-	○
	MobileKGQA	**low**	**0.136M**	○

- Recent on-device LLMs (e.g., MobileLLM[1]) typically have sub-billion parameters.
- In contrast, KGQA systems require significantly larger models.

1. Liu, Zechun, et al. "Mobilellm: Optimizing sub-billion parameter language models for on-device use cases." *Forty-first International Conference on Machine Learning*. 2024.

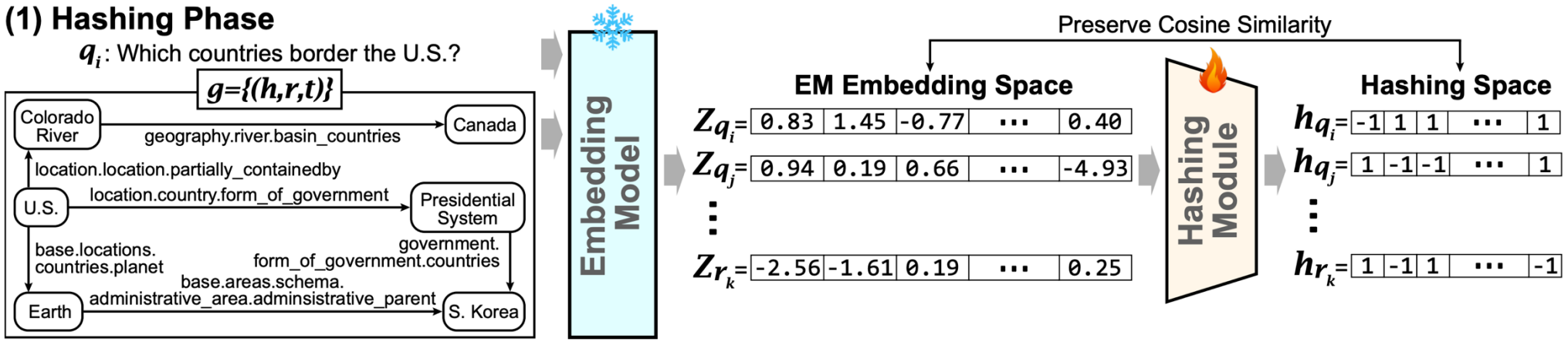
Limitations of Previous Methods

category	model	computation		adaptability
		latency	tuning parameters	
w/o LLM	RnG-KBQA (Ye et al., 2022)	low	443M	×
	DecAF (Yu et al., 2023)	low	848M	×
w/ LLM (finetuned)	RoG (Luo et al., 2024c)	low	7B	×
	ChatKBQA (Luo et al., 2024a)	low	7B	×
w/ LLM (frozen)	KB-BINDER (Li et al., 2023a)	high	-	○
	ToG (Sun et al., 2024)	high	-	○
	MobileKGQA	low	0.136M	○

- Recent on-device LLMs (e.g., MobileLLM[1]) typically have sub-billion parameters.
- In contrast, KGQA systems require significantly larger models.
- Training-free approaches suffer from high latency due to extensive inference.

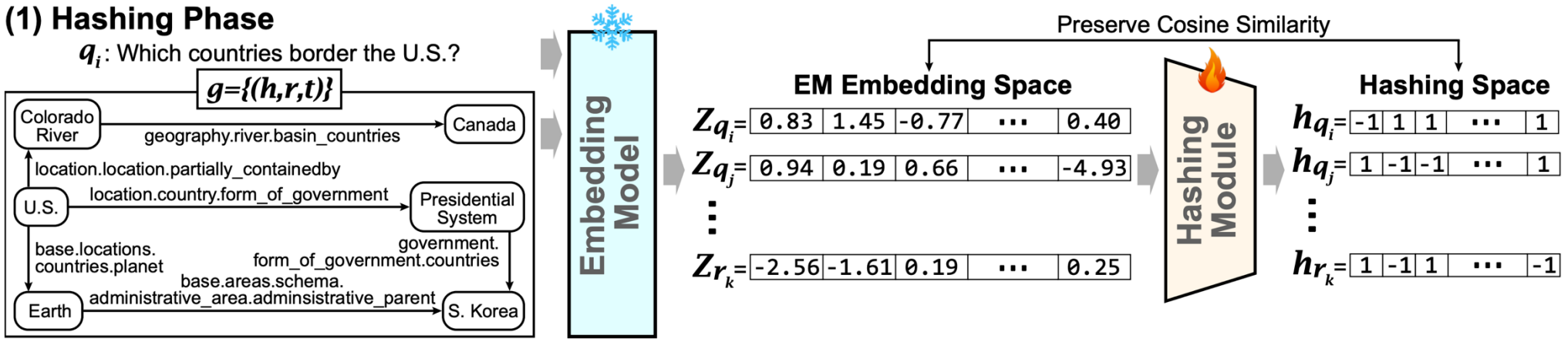
1. Liu, Zechun, et al. "Mobilellm: Optimizing sub-billion parameter language models for on-device use cases." *Forty-first International Conference on Machine Learning*. 2024.

Proposed Solution 1 – efficient KGQA system based on hash codes



- We propose a method that transforms the information of nodes and relations in a graph database into hash codes for storage.

Proposed Solution 1 – efficient KGQA system based on hash codes



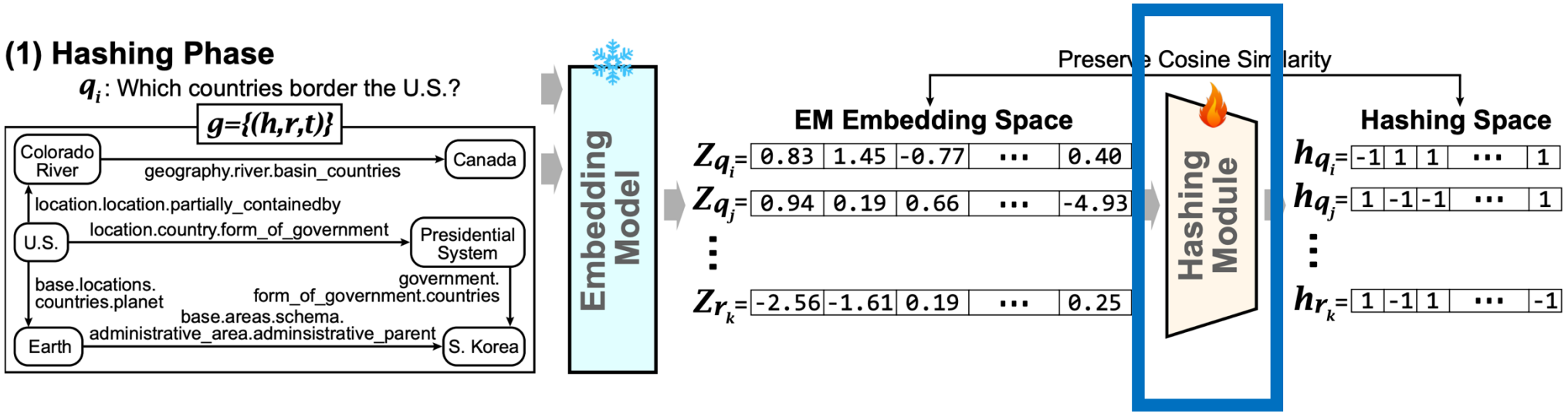
- We propose a method that transforms the information of nodes and relations in a graph database into hash codes for storage.

- This design provides two key advantages for KGQA systems:
  - Enables significantly faster search and information retrieval
  - Requires minimal additional storage due to binarization

1) https://microsoft.github.io/graphrag/

Proposed Solution 1 – efficient KGQA system based on hash codes

How should we train the hashing module ?



- We propose a method that transforms the information of nodes and relations in a graph database into hash codes for storage.

- This design provides two key advantages for KGQA systems:
  - Enables significantly faster search and information retrieval
  - Requires minimal additional storage due to binarization

1) https://microsoft.github.io/graphrag/

Proposed Solution 1 – efficient KGQA system based on hash codes

- **Derivation of loss function**
  - **If hashing is defined as a composition of a dimensionality-reduction mapping ϕ and a binarization mapping ψ, the objective is to maximize the mutual information $I(z, h)$ between the original embedding z and the hash code h, thereby minimizing information loss.**

$$\text{Maximize; } I(z, h); \text{ subject to; } h = \psi(\phi(z))$$

1) https://microsoft.github.io/graphrag/

Proposed Solution 1 – efficient KGQA system based on hash codes

- **Derivation of loss function**
  - **If hashing is defined as a composition of a dimensionality-reduction mapping ϕ and a binarization mapping ψ, the objective is to maximize the mutual information $I(z,h)$ between the original embedding z and the hash code h, thereby minimizing information loss.**

$$\text{Maximize; } I(z,h); \text{ subject to; } h = \psi(\phi(z))$$

Theorem 1 (Conditions for Mutual Information Maximization). *Suppose Assumption 1 holds, and both ϕ and ψ preserve the cosine similarity between any pair of embeddings in Z. Then, the mutual information $I(z,h)$ is maximized.*

1) https://microsoft.github.io/graphrag/

Proposed Solution 1 – efficient KGQA system based on hash codes

- **Derivation of loss function**
  - **If hashing is defined as a composition of a dimensionality-reduction mapping ϕ and a binarization mapping ψ, the objective is to maximize the mutual information $I(z, h)$ between the original embedding z and the hash code h, thereby minimizing information loss.**

$$\text{Maximize; } I(z, h); \text{ subject to; } h = \psi(\phi(z))$$

Theorem 1 (Conditions for Mutual Information Maximization). *Suppose Assumption 1 holds, and both ϕ and ψ preserve the cosine similarity between any pair of embeddings in Z. Then, the mutual information $I(z, h)$ is maximized.*

dimension reduction binarization

$$L_{\text{hash}} = \ell_\phi(Z_a, Z_i, Z_j) + \alpha \ell_\psi(d_a, d_i, d_j)$$

$$\ell_{\text{f}}(a, i, j) = \left\{ \log \frac{\text{sim}(f(a), f(i))}{\text{sim}(f(a), f(j))} - \log \frac{\text{sim}(a, i)}{\text{sim}(a, j)} \right\}^2$$

log-ratio loss

1) https://microsoft.github.io/graphrag/

Proposed Solution 2 – stepwise annotation generation

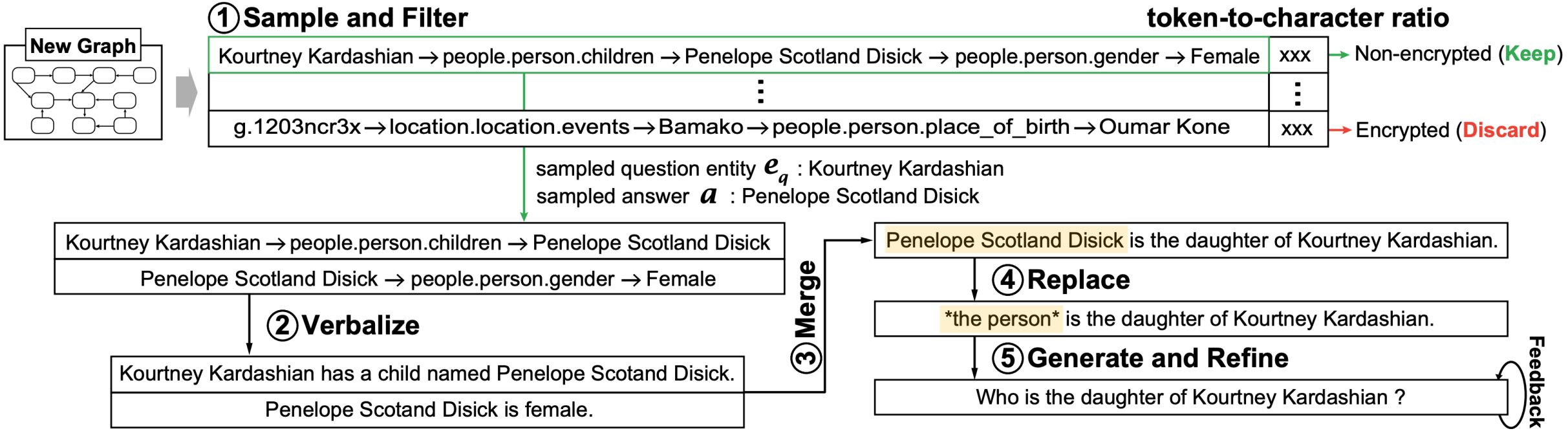


Figure 2: Adaptation Pipeline: stepwise generation of annotations. (Details in Sec. 4.3)

1) https://microsoft.github.io/graphrag/

Experiment

- **Q1) Does the proposed hash code–based KGQA system effectively reduce computational cost and improve the performance?**

(a) Computational cost for training

Model	P(M)	PFLOPs	T(h)	VRAM(GB)	Disk(GB)
NuTrea*	1.2	10.32	3.3	8.6	**0.51**
UnifiedSKG	2850	$\geqslant 10^3$	OOM	$\geqslant 32$	1.4
RoG	6738	$\geqslant 10^3$	OOM	$\geqslant 32$	2.9
GNN-RAG*	0.8	2.4	1.85	4.9	0.54
GNN-RAG†	1.1	8.3	2.10	10.4	0.54
GNN-RAG‡	1.7	25.1	2.22	26.3	0.54
SubgraphRAG†	4.2	5.3	1.74	**1.8**	18.12
MobileKGQA	**0.1**	**0.6**	**1.63**	3.5	0.6

Table 3: Resource usage and KGQA performance of MobileKGQA vs. GNN-RAG under 7W and 15W power modes on the **NVIDIA Jetson platform** (*: Whether the best configuration determined on server can be applied, †: CPU–GPU unified memory, ‡: reasoning-module performance, MobileKGQA utilized 256-bit hash codes. **Red** indicates **MobileKGQA** outperforms, **blue** otherwise.)

Mode	Model	Best* Config	Training Time (h)	RAM† (GB)	CPU (% / °C)		GPU (% / °C)		Energy (Wh)	Throttle	RM‡	
					Usage	Temp.	Usage	Temp.			Hit	F1
7W	GNN-RAG	impossible	5.9	7.3	25.7	49.3	55.7	49.7	28.7	No	61.7	54.3
	MobileKGQA	possible	2.1	5.0	23.5	48.9	70.1	50.1	11.8	No	74.2	62.9
	comparison	**Ours better**	**64.4%↓**	**31.5%↓**	**8.6%↓**	**0.8%↓**	**25.9%↑**	**0.8%↑**	**58.9%↓**	Tie	**20.3%↑**	**15.8%↑**
15W	GNN-RAG	impossible	3.6	7.3	19.9	50.9	50.4	50.0	22.5	Yes	61.7	54.3
	MobileKGQA	possible	1.1	5.0	18.4	49.6	55.5	51.4	6.84	No	74.2	62.9
	comparison	**Ours better**	**69.4%↓**	**31.5%↓**	**7.5%↓**	**2.6%↓**	**10.1%↑**	**2.8%↑**	**69.6%↓**	**Ours better**	**20.3%↑**	**15.8%↑**

Experiment

- **Q2) Does the proposed annotation generation method enable effective handling of distribution shift?**

Table 4: Hit score of various KGQA models on graph datasets for WebQSP and CWQ datasets across different domains based on Gemma 2 (2B, 4bit) model. (**red**: best results)

Dataset	WebQSP							CWQ						
Model	S(D1) → T(D2)			S(D1+D2) → T(D3)				S(D1) → T(D2)			S(D1+D2) → T(D3)			
	S_1	S_1(PA)	T_2	S_{12}	S_{12}(PA)	T_3	total	S_1	S_1(PA)	T_2	S_{12}	S_{12}(PA)	T_3	total
ToG	31.3		30.9	31.1		**32.1**	31.6	29.3		32.1	30.9		29.9	30.6
ReaRev	76.2		30.0	58.1		20.4	41.4	41.0		15.7	26.4		14.7	22.4
NuTrea	69.6	N/A	25.4	52.3	N/A	12.5	34.7	37.3	N/A	9.62	21.3	N/A	23.5	22.1
GNN-RAG	**79.2**		36.4	62.5		27.4	47.0	**57.9**		31.6	42.7		32.1	39.1
SubgraphRAG	22.9		15.6	20.0		14.3	17.5	30.1		10.3	18.7		5.1	14.0
MobileKGQA	79.1	78.9	**43.2**	**64.9**	63.4	30.4	**48.8**	57.8	60.2	**34.0**	**45.1**	45.6	**32.5**	**41.0**

1) https://microsoft.github.io/graphrag/