NLP&AI 연구실 세미나 (04/14, Thu)

Knowledge Graph 이모저모

김진성

Natural Language Processing & Artificial Intelligence

고려대학교 KOREA UNIVERSITY

Knowledge Graph-Guided Retrieval Augmented Generation

Xiangrong Zhu[♣] **Yuexiang Xie**[♡] **Yi Liu**[♣] **Yaliang Li**[♡] **Wei Hu**[♣]
[♣] State Key Laboratory for Novel Software Technology, Nanjing University, China
[♡] Alibaba Group
{xrzhu, yiliu07}.nju@gmail.com, whu@nju.edu.cn
{yuexiang.xyx, yaliang.li}@alibaba-inc.com

NAACL 2025 (Main)

SetKE: Knowledge Editing for Knowledge Elements Overlap

Yifan Wei[1], **Xiaoyan Yu**[2†], **Ran Song**[3], **Hao Peng**[1] and **Angsheng Li**[1†]
[1]State Key Laboratory of CCSE, School of Computer Science and Engineering, Beihang University
[2]School of Computer Science and Technology, Beijing Institute of Technology
[3]Kunming University of Science and Technology
{weiyifan, angsheng}@buaa.edu.cn, xiaoyan.yu@bit.edu.cn, song_ransr@163.com

IJCAI 2025

Motivation

*** Conventional RAG의 정보 종합 관점의 한계**

- 기존 Semantic RAG는 (실제로는 서로 연결된) 근거를 효과적으로 종합하지 못 함.

→ 즉, 대부분 retrieved chunks 간의 relations 를 모름. (LM에게 그냥 알아서 맡김)

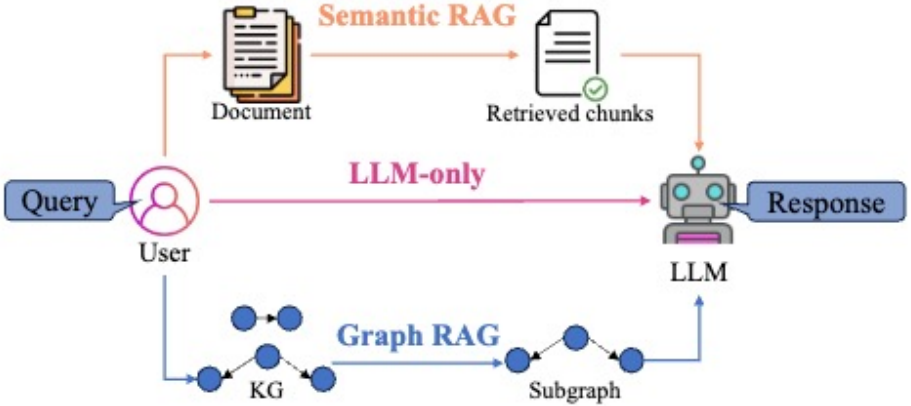


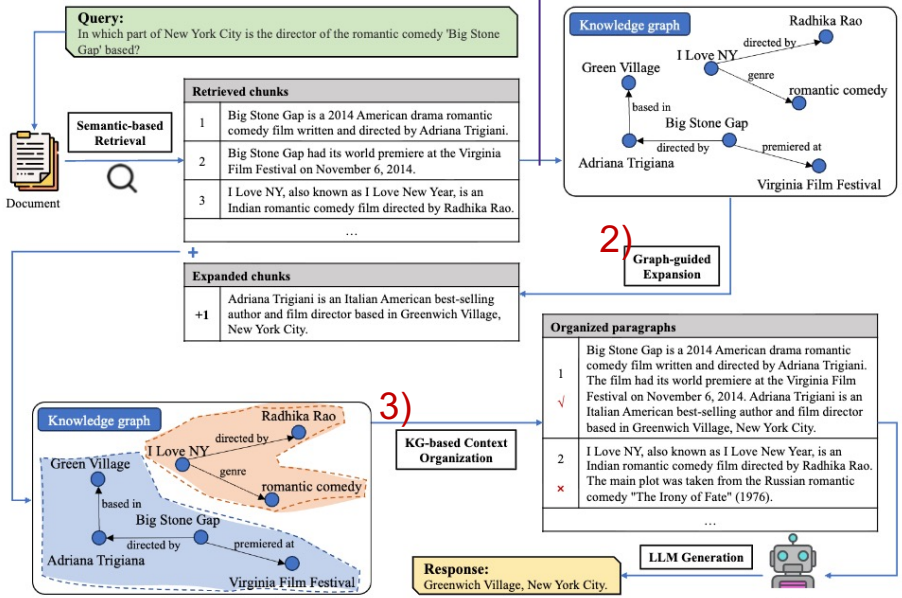
Figure 1: A comparison among LLM-only, Semantic RAG, and Graph RAG paradigms.

Method

* KG2RAG 프레임워크: 3 steps-based "retrieve → expand via KG → organize into coherent context"

1. Pre-processing: Offline KG 구축 (검색 pool)
   : Raw 데이터셋(HotpotQA) → 각 chunks 에서 triplet 추출 (s, r, o)
   → chunk index와 mapping 해놓기 (s,r,o,c)

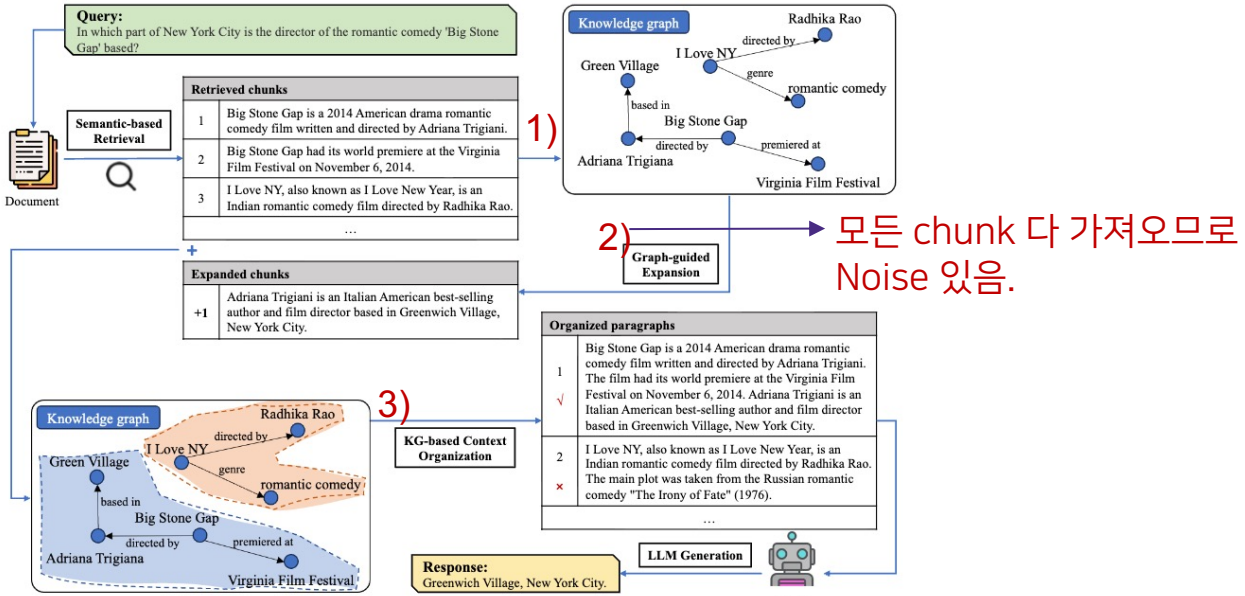
1) Llama-3-8B로 구축



Method

* KG2RAG 프레임워크: 3 steps-based "retrieve → expand via KG → organize into coherent context"

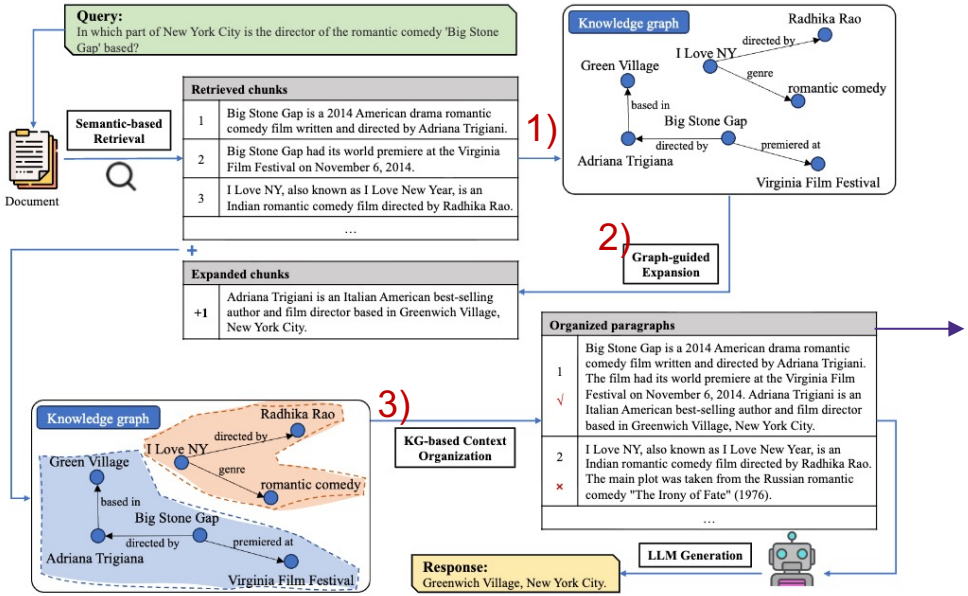
2. Retrieved chunks를 seed chunks 삼아서, KG 를 따라 entity/relation이 이어지는 다른 chunks 까지 graph-guided expansion
   → when to stop? hyperparam. *m*-hop (탐색거리 내의 모든 edges 추가 획득)



Method

* KG2RAG 프레임워크: 3 steps-based "retrieve → expand via KG → organize into coherent context"

3. Relevance 높은 chunks 만 남겨서, coherent paragraph 형태로 organization
   → filtering을 위한 relevance 는 query와 chunks 간의 유사도 및 리랭킹하여 측정
   → 단순 merge 안하고, MST(Maximum Spanning Tree) 기반 핵심 지식 경로만 남김



Context Organization:
그래프 경로(DFS)를 따라, 논리적 chunks 나열
→ entities 간 relation 중심의 rerank
→ 표현 자체를 새로 쓰는 것은 아님.

Experiment

* Main Results
 - response quality: 최종 응답 품질
 - retrieval quality: LLM 입력 직전 chunks가 얼마나 reference facts를 잘 반영하는가

Methods	Hotpot-Dist			Hotpot-Full			Shuffle-Hotpot-Dist			Shuffle-Hotpot-Full		
	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall
LLM-only	0.237	0.259	0.234	0.237	0.259	0.234	0.158	0.175	0.158	0.158	0.175	0.158
Semantic RAG	0.617	0.646	0.643	0.528	0.558	0.535	0.508	0.533	0.524	0.422	0.449	0.433
+ Rerank	0.652	0.685	0.665	0.587	0.613	0.603	0.532	0.560	0.546	0.447	0.476	0.456
Hybrid RAG	0.653	0.676	0.655	0.551	0.582	0.558	0.520	0.548	0.534	0.443	0.473	0.446
LightRAG	0.293	0.288	0.480	0.261	0.259	0.364	0.285	0.284	0.404	0.202	0.199	0.293
GraphRAG	0.400	0.408	0.491	0.169	0.157	0.429	0.351	0.365	0.401	0.163	0.155	0.362
KG^{2}RAG	**0.663**	**0.690**	**0.683**	**0.631**	**0.665**	**0.643**	**0.545**	**0.572**	**0.566**	**0.507**	**0.539**	**0.512**

Table 1: Comparisons in terms of response quality between KG^{2}RAG and baselines.

Methods	Hotpot-Dist			Hotpot-Full			Shuffle-Hotpot-Dist			Shuffle-Hotpot-Full		
	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall
Semantic RAG	0.343	0.206	0.894	0.300	0.178	0.790	0.321	0.201	0.837	0.268	0.167	0.708
+ Rerank	0.357	0.224	**0.932**	0.306	0.197	0.833	0339	0.213	**0.886**	0.286	0.179	0.754
Hybrid RAG	0.354	0.222	0.921	0.302	0.189	0.795	0.334	0.210	0.837	0.279	0.174	0.739
LightRAG	0.234	0.150	0.638	0.132	0.083	0.340	0.227	0.148	0.535	0.116	0.073	0.295
GraphRAG	0.255	0.167	0.594	0.180	0.113	0.470	0.210	0.138	0.482	0.199	0.126	0.510
KG^{2}RAG	**0.436**	**0.301**	0.908	**0.310**	**0.203**	**0.838**	**0.405**	**0.279**	0.840	**0.305**	**0.193**	**0.790**

Table 2: Comparisons in terms of retrieval quality between KG^{2}RAG and baselines.

* shuffle setting: LLM 사전 지식 배제를 위한 변형 자체 데이터
→ 문서 내의 entity를 동일 category 내 다른 entity로 무작위 교체한 setting.

Experiment

* Ablation
 - Coherent Organization: retrieval 품질에서 유의미, response 에서 글쎄
 - KG-based expansion 효과: response 품질에서 유의미, retrieval 에서 글쎄

	Response Quality			Retrieval Quality			
	F1	Precision	Recall	F1	Precision	Recall	#Avg.
KG^{2}RAG	0.663	0.690	0.683	0.436	0.301	0.908	8.11
w/o organization	0.660	0.678	0.679	0.259	0.153	0.963	16.76
w/o expansion	0.626	0.653	0.645	0.473	0.341	0.842	4.41

Table 3: Experimental results of an ablation study conducted on HotpotQA in the distractor setting.

Limitation

*** Wikipedia-based Context**

- HotpotQA, TriviaQA, Musique → Wikipedia 기반 Benchmarks

→ 기본적으로 Wikpedia 자체가 Entity-centric이므로,
비정형적인 web-crawled data, 도메인에서도 강건할지는 의문점으로 남음.

Motivation

*** Knowledge Element Overlap (KEO) 문제**

- 문제 의식: "기존 knowledge editing (KE)는 **overlapping knowledge**를 잘 다루지 못 한다."

- conventional KE는 통상 단일 **triplet** (s, r, o) 내 o → o* 수정하는 task

  But, 현실에서는 같은 (s, r)에 대해 여러 object 가 동시에 참인 상황 발생 가능.

  t1: (Roman Empire, continent, **Europe**)
  t2: (Roman Empire, continent, **Asia**)

  → 즉, 단일 object editing이 아닌, object 집합 전체를 일관되게 수정하는 문제로 간주해야 한다.
  (s, r, o) → (s, r, o*) 가 아니라, (s, r, O) → (s, r, O*) 로 문제 정의를 다시 해야한다.

Motivation

* Knowledge Element Overlap (KEO) 문제
 - 좌(a): 기존 KE에서 다루던 일반 현상
  우(b): KEO 현상이 일어날 때 뉴런 활성화 시각화

  → 기존에는 편집할 때 뉴런 단위에서 충돌 안 나는게 당연했다.

  → 근데 overlapped knowledge 에서는 편집할 때 문제 생긴다.
  (무슨 문제? knowledge overwriting:
  e.g., 로마제국 위치를 Africa로 바꿀려고 하면, Europe 뿐만 아니라, Asia 까지 Africa 로 덮어씌워져버린다.)

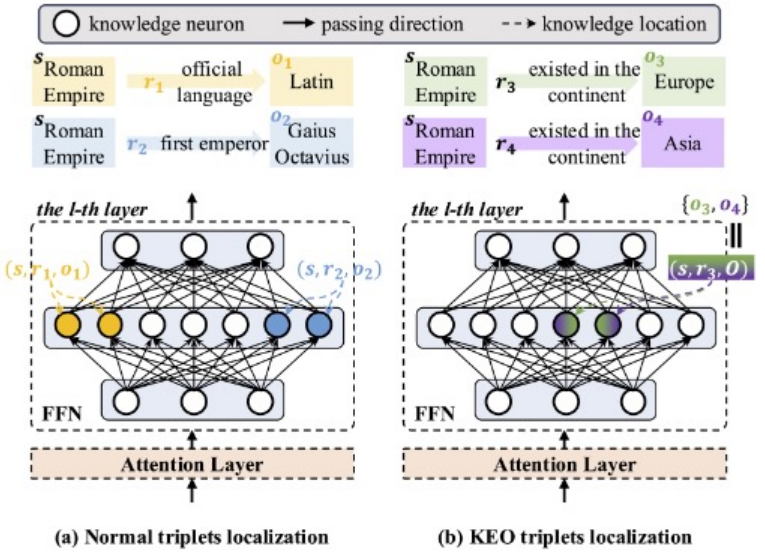


Figure 1: A demonstration of normal triplets and KEO triplets, along with a toy example showing how they are localized within Transformer-based LLMs, where normal triplets are mapped to distinct neurons, and KEO triplets are mapped to overlapping neurons.

Probing

* KEO 진짜 일어나나?
 - 기존 유명한 KE 벤치마크인 “Counterfactual” 데이터셋에서 KEO 케이스만 골라서 테스트
  → 기존 KE 방법론들 성능 폭락

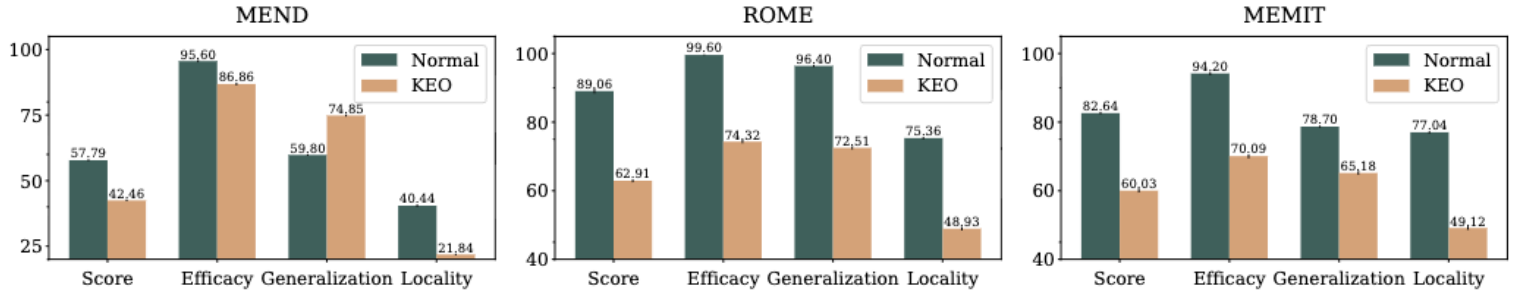


Figure 2: Comparison of editing performance on Normal and KEO type for MEND, ROME, and MEMIT.

Method

* Knowledge Set Editing (KSE)
  - object 집합 전체를 일관되게 수정하여, set 단위 일관성을 유지하는 문제로 치환 (s, r, O) → (s, r, O*)

  t1: (Roman Empire, continent, **Europe**)
  t2: (Roman Empire, continent, **Asia**)

  → e.g.) {**Europe**, Asia}를 {**Africa**, Asia}로.

Method: Benchmarking

*** EDITSET:** KSE 전용 벤치마크 만들어봄

- Wikidata에서 발견된 710개의 KEO relations 중에서,
  기존 연구와 겹치는 31개 주요 relation subset을 기반으로 40,904개 instances로 만든 것.

- main 및 ablation 실험에는, 10,000개, 7,500개 instances 씩만 사용 (비용 절감?)

Method

*** SetKE 프레임워크: Set Knowledge Editor**

A. 수정 전 모델:
Query: “로마 제국은 어느 대륙에 위치하는가?”

문제 1) y^1, y^2 에서 knowledge overwriting 의 위험성
문제 2) y^3 에서 오답을 뱉고 있는 상태임.

B. Bipartite Matching (이분 매칭) 기술 적용
: 모델의 예측값 집합과 원하는 목표값(정답) 집합을 연결
* How? **Hungarian Algorithm**
: 행렬 내 예측값(y)과 목표값(y^) 사이의 **매칭 비용** 계산
→ 최적의 쌍 찾기 (비용이 **가장 낮은**)
* 목적: 어떤 y가 어떤 y^로 변해야 모델의 전체적인 지식 구조를 가장 적게 해치면서 정확하게 수정될지를 결정
* 정답 세트를 '작업자', 예측 세트를 '일'로 간주
→ 전체 비용이 최소가 되도록 최적으로 배정

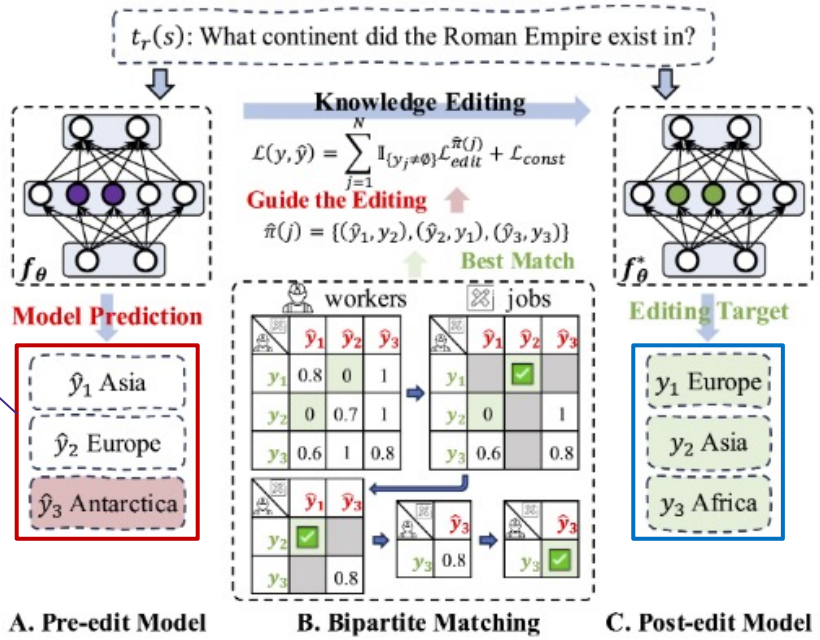


Figure 3: Simplified illustration of the SetKE framework.

Method

*** SetKE 프레임워크: Set Knowledge Editor** - B. Bipartite Matching (이분 매칭)

1. 현재 상황:
   (Roman Empire, continent, O) → (Roman Empire, continent, O*)
   O: {Asia, Europe, Antarctica}
   O*: {Asia, Europe, Africa}

   → y^j (y^1: Asia, y^2: Europe, y^3: Antarctica)
   y_i (y1: Asia, y2: Europe, y3: Africa)

정답(y) \ 예측($\hat{y}$)	$\hat{y}_1$ (Asia)	$\hat{y}_2$ (Europe)	$\hat{y}_3$ (Antarctica)
y_1 (Asia)	**-1.0**	0.0	0.0
y_2 (Europe)	0.0	**-1.0**	0.0
y_3 (Africa)	0.0	0.0	**0.0**

2. 두 set 사이의 (O-O*) 매칭 비용 계산하여 비용 행렬(Cost Matrix) 구축
   . pair 별 비용함수 C_match()

$$C_{match}(i,j) = -P(\hat{y}_j = y_i)$$

~ P(목표|예측)

: 정답 set에서 i-th index, 예측 set에서 j-th index 골랐을 때,
모델의 예측 y^_j가 실제 정답 y_i와 일치할 확률
→ 확률 앞에 마이너스(-)가 붙어 있으므로, 일치 확률이 높을수록 비용(C_match)은 낮아짐.

Method

*** SetKE 프레임워크: Set Knowledge Editor** - B. Bipartite Matching (이분 매칭)

3. 헝가리안 알고리즘으로 "최소 비용" 매칭
 - 목표값(Workers)과 예측값(Jobs) 사이에서 전체 비용의 합이 가장 낮은(즉, 확률의 합이 가장 높은) 최소 조합 찾기.

 - 최적 매칭 결과:

Jobs

Workers

정답(y) \ 예측($\hat{y}$)	$\hat{y}_1$ (Asia)	$\hat{y}_2$ (Europe)	$\hat{y}_3$ (Antarctica)
y_1 (Asia)	**-1.0**	0.0	0.0
y_2 (Europe)	0.0	**-1.0**	0.0
y_3 (Africa)	0.0	0.0	**0.0**

* 시나리오 A (채택: 비용 최소)

- $(y_1, \hat{y}_1) : Asia \leftrightarrow Asia \Rightarrow -1.0$ (매우 좋음)
- $(y_2, \hat{y}_2) : Europe \leftrightarrow Europe \Rightarrow -1.0$ (매우 좋음)
- $(y_3, \hat{y}_3) : Africa \leftrightarrow Antarctica \Rightarrow 0.0$ (보통)
- **Total Cost** = -2.0

* 시나리오 B (불채택)

- $(y_1, \hat{y}_3) : Asia \leftrightarrow Antarctica \Rightarrow 0.0$ (나쁨 - $Asia$가 나올 확률이 없음)
- $(y_2, \hat{y}_2) : Europe \leftrightarrow Europe \Rightarrow -1.0$ (매우 좋음)
- $(y_3, \hat{y}_1) : Africa \leftrightarrow Asia \Rightarrow 0.0$ (나쁨 - $Africa$가 나올 확률이 없음)
- **Total Cost** = -1.0

Method

*** SetKE 프레임워크: Set Knowledge Editor** - B. Bipartite Matching (이분 매칭)

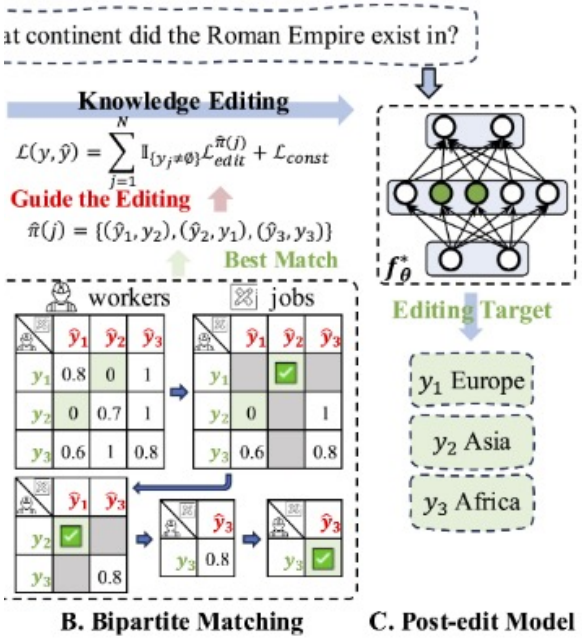
- 최종 손실 함수 : 각 쌍에 대해 모델 가중치 업데이트

$$L(y, \hat{y}) = \sum_{j=1}^{N} 1_{\{y_j \neq \emptyset\}} L^{edit}_{\hat{\pi}(j)} + L_{const}$$

즉, 매칭된 쌍에 대해서만 손실 계산

. L^edit_pi^(j): 선택된 매칭에 대해 편집 목표를 달성했는지 측정하는 손실
. L_const: 편집 범위를 벗어난 다른 지식(Locality)이 파괴되지 않도록 제약하는 KL Divergence 기반의 손실

→ "기존에 Antarctica (y^_3)를 내뱉던 3번 예측 위치 pi_j 에서, 이제는 정답인 Africa (y_3)가 나올 확률을 높여라"
= "로마 제국"이라는 뉴런 뭉치 전체를 무작정 바꾸는 게 아니라, 그중에서 Antarctica를 담당하던 부분만 골라내어 Africa로 갈아끼우기.
= Asia↔Asia, Europe↔Europe 이미 잘 매칭된 쌍들에 대해서는, 손실이 거의 발생하지 않으므로, 기존의 올바른 지식은 그대로 유지



plified illustration of the SetKE framework.

Experiment

*** Main Result**

- KEO 상황에서는 KEO 전용 KE 방법론을 써야 성능 잘 나온다.
- Efficacy: 편집된 지식(Target Edit)에 대해 모델이 의도한 정답을 얼마나 정확하게 출력하는지
  Generalization: 동일한 지식을 묻는 다른 형태의 질문(패러프레이징 등)에 대해서도 올바르게 답하는지
  Locality: 편집 대상이 아닌 관련 없는 지식들에 대해 모델의 기존 성능이 변하지 않고 잘 유지되는지
  Score: 지표 종합 평균 성능 점수

	GPT2 Large (760M)				GPT2 XL (1.5B)			
Editor	**Score**	**Efficacy**	**Generalization**	**Locality**	**Score**	**Efficacy**	**Generalization**	**Locality**
FT-W	51.46	58.13 (0.5)	45.29 (0.5)	52.58 (0.4)	55.18	65.66 (0.5)	50.97 (0.5)	51.24 (0.4)
KN	42.21	38.22 (0.5)	37.46 (0.5)	54.89 (0.4)	40.65	35.93 (0.5)	35.86 (0.5)	55.29 (0.4)
MEND	–	–	–	–	38.75	89.58 (0.3)	81.58 (0.4)	18.52 (0.3)
PMET	43.34	40.82 (0.5)	39.24 (0.5)	51.98 (0.4)	44.13	41.15 (0.5)	40.08 (0.5)	53.39 (0.4)
MEMIT	49.65	49.53 (0.5)	45.64 (0.5)	54.59 (0.4)	56.60	61.12 (0.5)	55.07 (0.5)	54.11 (0.4)
ROME	64.08	72.03 (0.4)	69.63 (0.4)	53.84 (0.4)	65.89	75.27 (0.4)	73.58 (0.4)	53.62 (0.4)
SetKE	71.47	88.57 (0.3)	80.91 (0.4)	54.77 (0.4)	75.28	95.90 (0.2)	91.68 (0.2)	54.14 (0.4)

Table 3: Numerical results on EDITSET for 10,000 edits (95% confidence intervals in parentheses).

감사합니다