하계 세미나

이재욱

Knowledge Editing

대규모 언어 모델의 성능은 계속 발전하고 있음

모델 성능과는 별개로, world knowledge는 계속 변화함

모델은 데이터를 학습할 때, world knowledge의 특정 시점을 학습

때문에 시간이 지나면 모델이 학습한 지식과 실제 지식이 맞지 않게 됨 (outdated)

이는 모델이 factual하지 않거나, outdated answer를 생성하는 원인으로 작용함

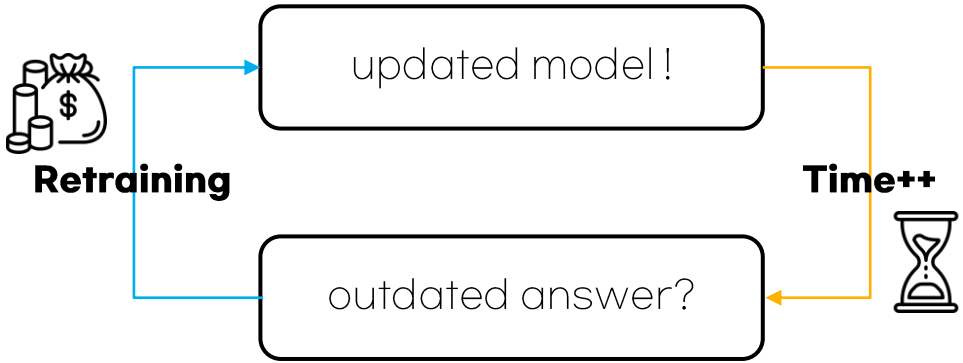
Knowledge Editing

그렇다고 지식이 새롭게 변할 때마다 모델을 다시 학습시킨다?

매번 모델 재학습하기에는 비용이 너무 비쌈

모델이 가지고 있는 지식 중 시간이 흐름에 따라 변화하는 지식의 일부만 반영해보자

-> Model Editing



Knowledge Editing

Knowledge Editing은 방법에 따라 여러 갈래로 나뉨

Preserve Parameters

1. Memory-based
   외부 메모리나 데이터베이스를 활용하여 정보를 저장하고 검색.
   모델의 핵심 파라미터를 변경하지 않고도 추가 정보에 엑세스

2. Additional Parameters
   AdaptorLr Side networks같은 메커니즘을 통해 추가
   파라미터를 도입하여 새로운 정보나 기능을 통합

3. Change LM's representation space
   모델이 작동하는 representation space를 변경하는 것을 포함.
   임베딩 수정이나 변환 기술을 통해 정보를 표현하는 방식을 변경

Modify Parameters

1. Finetuning
   새로운 데이터를 사용하여 pre-trained LM의 파라미터를 조정

2. Meta-learning
   최소한의 데이터로 새로운 task에 적응할 수 있게 모델을
   훈련시키는 방법. 주로 gradient 기반 방법이나 강화학습 기술을
   통해 파라미터를 수정

3. Locate-then-Edit
   특정 작업이나 정보와 관련된 모델의 파라미터를 식별한 후, 타겟
   파라미터를 편집하여 새로운 정보나 능력을 통합

Knowledge Editing

Knowledge Editing은 공통적으로 아래와 같은 기준으로 평가함

"Editing Factual Knowledge in Language Models" (Cao et al., 2021)
"Model Editing with MEMIT: Improving Locality and Generality while Preserving Reliability" (Mitchell et al., 2022)

Efficacy: KE methods가 목표한 지식을 정확히 수정했는지를 평가

$$\text{Efficacy} = \frac{1}{n}\sum_{i=1}^{n}\mathbb{I}[o_i^* = \arg\max_o \mathbb{P}_{G_i'}[o]]$$

Generality: 보통 수정된 지식에 대한 paraphrased query에도 잘 답하는지 평가

$$\text{Generality} = \frac{1}{n}\sum_{i=1}^{n}\mathbb{I}[o_i^* = \arg\max_o \mathbb{P}_{G'}[o_i]]$$

Specificity(Locality): 수정할 지식과 관련 없는 지식에 미치는 영향을 평가

$$\text{Specificity} = 1 - \frac{1}{n}\sum_{i=1}^{n}\mathbb{I}[\arg\max_o \mathbb{P}_{G'}[o_i] \neq \arg\max_o \mathbb{P}_{G}[o_i]]$$

Knowledge Graph Enhanced Large Language Model Editing

Mengqi Zhang[1*] , **Xiaotian Ye** [2*] , **Qiang Liu**[3] , **Pengjie Ren**[1], **Shu Wu**[3] , **Zhumin Chen**[1]
[1]School of Computer Science and Technology, Shandong University
[2]School of Computer Science, Beijing University of Posts and Telecommunications
[3]Center for Research on Intelligent Perception and Computing
State Key Laboratory of Multimodal Artificial Intelligence Systems
Institute of Automation, Chinese Academy of Sciences
{mengqi.zhang, renpengjie, chenzhumin}@sdu.edu.cn
yexiaotian@bupt.edu.cn
{qiang.liu,shu.wu}@nlpr.ia.ac.cn

Motivation

기존 Knowledge editing methods들은 수정한 지식과 연관된 지식들을 통합하는 것에 어려움을 겪음
이러한 현상은 post-edit model의 일반화 능력을 제한함

(수정이 잘 안되는 예시)

"LeBron James plays for the Miami Heat"
"LeBron James works in Miami"
"LeBron James lives in Miami"

"LeBron James plays for the Los Angeles Lakers"
"LeBron James works in Miami"
"LeBron James lives in Miami"

Motivation

편집된 지식과 연관된 지식도 잘 수정되어야 하지 않을까?

Motivation

기존 KE methods가 수정할 지식과 연관된 지식을 통합하지 못하는 것은

기존 방법들이 주로 {s, r, o} 형태의 단일 지식(triplet)을 편집하는데 중점을 두기 때문

== 편집된 지식과 관련된 다른 지식들이 모델 내에서 적절히 업데이트 되지 않음

이러한 문제를 해결하기 위해

> 외부 Knowledge graph를 기반으로 편집된 지식과 관련된 새로운 연관 지식을 추출하고

이를 모델에 통합하는 방법인 GLAME을 제안

methods

GLAME

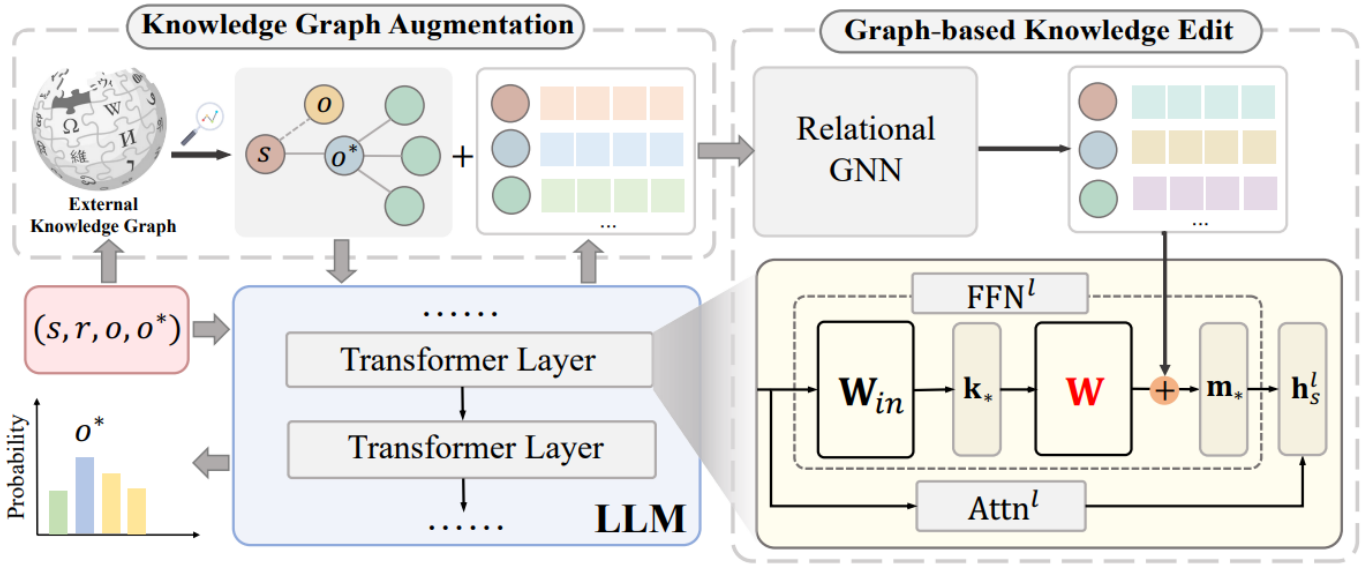
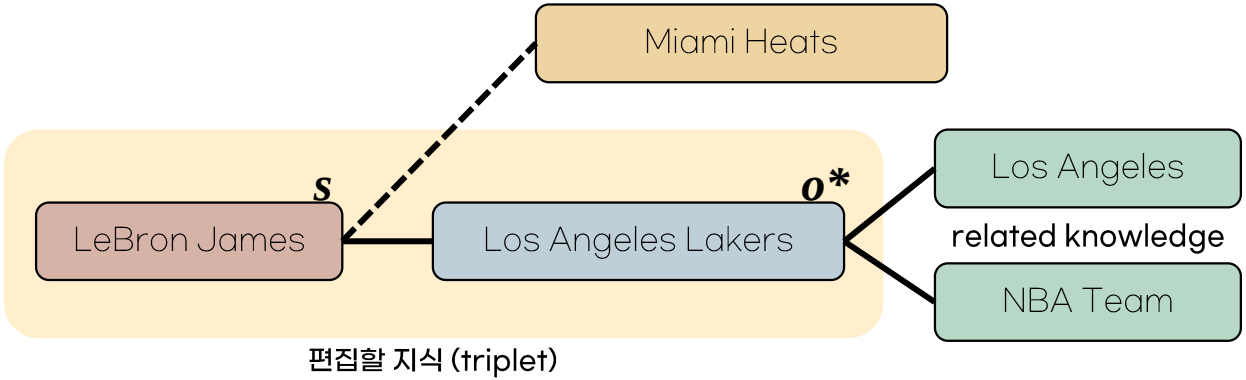


Figure 2: An illustration of GLAME architecture. We first utilize a Knowledge Graph Augmentation module to sample a high-order subgraph, recording the associated knowledge of changes caused by the edit (s, r, o, o^*). Subsequently, the entities and relations within the subgraph are encoded using the LLM, from which hidden vectors are extracted from the early layers as the initial representations of the entities and relations in the subgraph. Then, the well-designed Graph-based Knowledge Edit module leverages a relational graph neural network to incorporate new knowledge associations from the subgraph into the parameter editing process.

methods

Knowledge Graph Augmentation

외부 지식 그래프 (Wikidata)를 사용하여 편집된 지식과 관련된 새로운 연관 지식을 포함하는 서브그래프를 구성



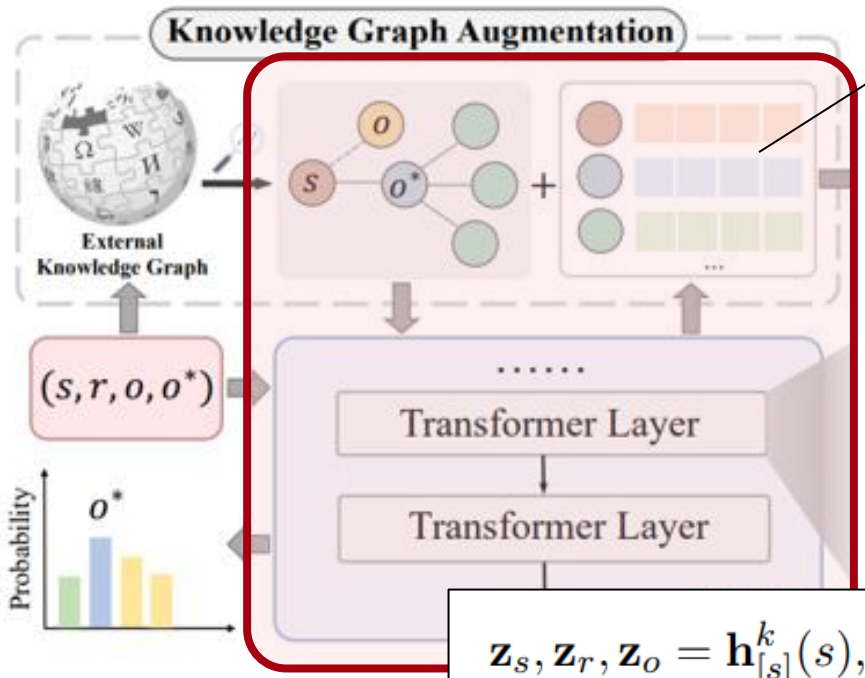
{LeBron James, works in, Los Angeles}
{LeBron James, lives in, Los Angeles}

편집한 지식의 subject, object와 관련된 연관 지식을 subgraph로 구성

methods

Knowledge Graph Augmentation

구성한 서브그래프를 LLM에 입력하여 initial representation을 획득



$$\mathbf{z}_s, \mathbf{z}_r, \mathbf{z}_o = \mathbf{h}^k_{[s]}(s), \mathbf{h}^k_{[r]}(r), \mathbf{h}^k_{[o]}(o), \qquad (4)$$

methods

Graph-based Knowledge Edit

Subgraph Encoding
- LLM에 입력해서 추출한 서브그래프의 initial representation을 RGNN을 통과시켜 인코딩

Relational Graph Neural Network

$$\mathbf{z}_s^{l+1} = g\left(\sum_{o \in \mathcal{N}_s} \mathbf{W}_1\left(\mathbf{z}_o^l + \mathbf{z}_r\right) + \mathbf{W}_2 \mathbf{z}_s^l\right)$$

RGNN의 레이어를 지나면 subject의 entity representation z_s^n을 획득

methods

Relational Graph Neural Network

더 자세히 살펴보면 아래와 같다...

/* Optimizing $\mathbf{m}_*$ */
while not converged **do**
  /* Subgraph encoding */
  $\mathbf{z}_s^n \leftarrow \mathrm{RGNN}(\mathcal{G}_n^m(e))$, Eq (5);
  /* Computing $\mathbf{m}_*$ */
  $\mathbf{m}_* \leftarrow$ Eq (6) ;
  /* Learning Objective */
  $\mathcal{L} \leftarrow \mathcal{L}_p + \lambda \mathcal{L}_a$, Eq (7);
  Update parameters of RGNN.
end
/* Computing $\mathbf{k}_*$ */

$$\mathbf{k}_* = \frac{1}{N} \sum_{j=1}^{N} f(\mathbf{W}_{in}^l \cdot \mathbf{h}_s^{l-1}). \quad (8)$$

$$\mathbf{z}_s^{l+1} = g\left(\sum_{o \in \mathcal{N}_s} \mathbf{W}_1 \left(\mathbf{z}_o^l + \mathbf{z}_r \right) + \mathbf{W}_2 \mathbf{z}_s^l \right), \quad (5)$$

subgraph encoding

$$\mathbf{m}_* = \mathbf{m}_s^l + \mathbf{z}_s^n, \quad (6)$$

computing m*

$$\mathcal{L} = \mathcal{L}_p + \lambda \mathcal{L}_a, \quad (7)$$

RGNN loss function

$$\mathcal{L}_p = -\frac{1}{N} \sum_{j=1}^{N} \log \mathbb{P}_{\mathcal{F}(m_s' := m_*)} \left[o^* \mid x_j \oplus p(s,r) \right]$$

$$\mathcal{L}_a = D_{KL} \left(\mathbb{P}_{\mathcal{F}(m_s' := m_*)} \left[x \mid p' \right] \parallel \mathbb{P}_{\mathcal{F}} \left[x \mid p' \right] \right)$$

methods

Graph-based Knowledge Edit

GLAME은 가중치를 업데이트하기 위한 아래의 최적화 문제를 적용함
(특정 레이어의 가중치를 업데이트하는 최적화 문제 - ROME)

$$\hat{\mathbf{W}} = \mathbf{W} + \frac{(\mathbf{m}_* - \mathbf{W}\mathbf{k}_*)(\mathbf{C}^{-1}\mathbf{k}_*)^{\mathrm{T}}}{(\mathbf{C}^{-1}\mathbf{k}_*)^{\mathrm{T}}\mathbf{k}_*},$$

where $\mathbf{C} = \mathbf{K}\mathbf{K}^{\mathrm{T}}$ represents a constant matrix,

아까 열심히 알아본 m*, k*를
여기 사용해서 모델의 지식을 편집하는 것...

Experiments

Dataset:
- COUNTERFACT(2022), COUNTERFACTPLUS(2023), MQUAKE(2023)

Models:
- GPT-2 XL(1.5B)
- GPT-J(6B)

Methods:
- Constrained Fine-Tuning(FT)
- MEND
- ROME (+ROME-KG)
- MEMIT (+MEMIT-KG)

Experiments - (1)

Editor	Effi.Score	Para.Score	Neigh.Score	Port.Score	Edit.Score
GPT-2 XL (1.5B)	22.20	24.70	78.10	10.18	20.35
FT	**100.00**	87.90	40.40	15.13	35.64
MEND	99.10	65.40	37.90	11.15	28.28
ROME	99.95	96.48	75.44	21.43	49.82
ROME-KG	73.85	72.41	74.65	5.24	17.27
MEMIT	93.79	80.22	77.05	18.71	44.67
MEMIT-KG	53.09	45.28	**77.90**	9.99	26.00
GLAME	99.84	**96.62**	76.82	**23.95**	**53.24**
GPT-J (6B)	16.30	18.60	83.00	11.44	18.64
FT	100.00	98.80	10.30	17.84	23.09
MEND	97.40	53.60	53.90	12.99	32.14
ROME	100.00	99.27	79.00	29.67	60.21
ROME-KG	68.90	67.12	78.59	13.68	34.55
MEMIT	100.00	95.23	81.26	29.77	60.24
MEMIT-KG	53.75	40.22	**82.80**	8.63	23.33
GLAME	**100.00**	**99.30**	81.39	**33.04**	**63.87**

Table 1: Performance comparison on COUNTERFACT in terms of Efficacy Score (%), Paraphrase Score (%), and Neighborhood Score (%), and COUNTERFACTPLUS in terms of Portability Score (%). The Editing Score (%) is the harmonic mean of the four evaluation metrics. The best performance is highlighted in boldface, and the second-best is underlined. Gray numbers indicate a clear failure on the corresponding metric.

Experiments - (2)

Editor	Effi.Score	Para.Score	Neigh.Score	Port.Score	Edit.Score
GPT-2 XL (1.5B)	22.20	24.70	78.10	10.18	20.35
GLAME w/ MLP	99.79	91.79	**77.05**	21.73	50.55
GLAME w/ GNN	99.79	94.95	77.02	22.59	51.41
GLAME w/o GKE	**99.95**	96.48	75.44	21.43	49.82
GLAME	99.84	**96.62**	76.82	**23.95**	**53.24**
GPT-J (6B)	16.30	18.60	83.00	11.44	18.64
GLAME w/ MLP	99.85	98.28	80.41	30.45	61.94
GLAME w/ GNN	100.00	98.20	81.03	30.16	60.90
GLAME w/o GKE	100.00	99.27	79.00	29.67	60.21
GLAME	**100.00**	**99.30**	**81.39**	**33.04**	**63.87**

Table 2: Ablation studies on COUNTERFACT in terms of Efficacy Score (%), Paraphrase Score (%), and Neighborhood Score (%), and COUNTERFACTPLUS in terms of Portability Score (%).

Experiments - (3)

Editor	Average Score	2-hops	3-hops	4-hops
GPT-2 XL (1.5B)	21.29	25.13	23.3	15.43
ROME	29.70	39.80	31.07	18.23
MEMIT	26.52	35.87	27.70	16.00
GLAME	**31.48**	**41.83**	**32.10**	**20.50**
$\Delta Improve$	5.98%	5.10%	3.32%	12.45%
GPT-J (6B)	16.83	15.80	23.60	11.10
ROME	33.15	42.80	38.37	18.27
MEMIT	27.46	35.77	33.03	13.57
GLAME	**35.11**	**44.13**	**39.87**	**21.33**
$\Delta Improve$	5.92%	3.11%	3.91%	16.75%

Table 6: Performance comparison of editors on multi-hop questions of MQuAKE dataset in terms of Efficacy Score (%).

MEMoE: Enhancing Model Editing with Mixture of Experts Adaptors

Renzhi Wang[1,2], **Piji Li**[1,2*]
[1] College of Computer Science and Technology,
Nanjing University of Aeronautics and Astronautics, China
[2] MIIT Key Laboratory of Pattern Analysis and Machine Intelligence, Nanjing, China
[1]{rzhwang,pjli}@nuaa.edu.cn

Motivation

Knowledge Edit은 특정 query에 대한 LLM의 출력을 효율적으로 수정함

그러면서도 Edit target knowledge와 상관없는 지식은 잘 유지하는 것이 중요함

하지만 최근의 KE methods -> Generality와 Locality 사이의 밸런스를 맞추는 데 어려움을 겪음

이러한 부분을 개선하기 위해 MoE 아키텍처와 Knowledge Anchor Router 전략을 활용한

Model Editing adaptor인 MEMoE를 제안

MEMoE

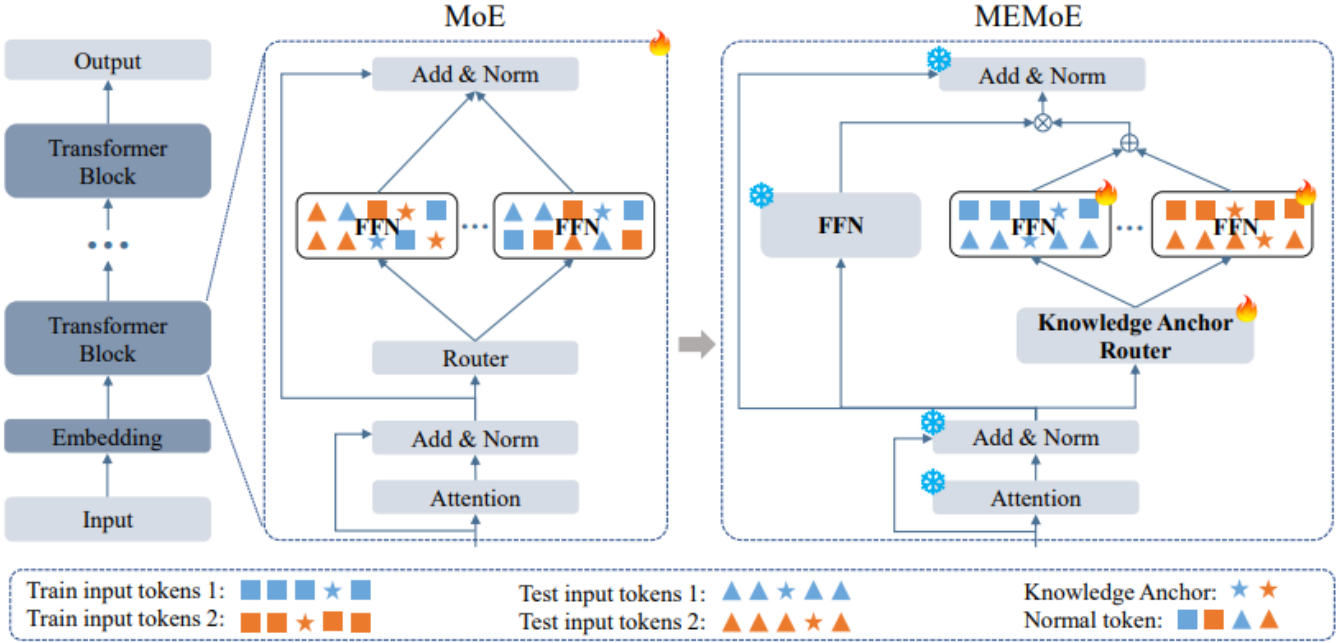
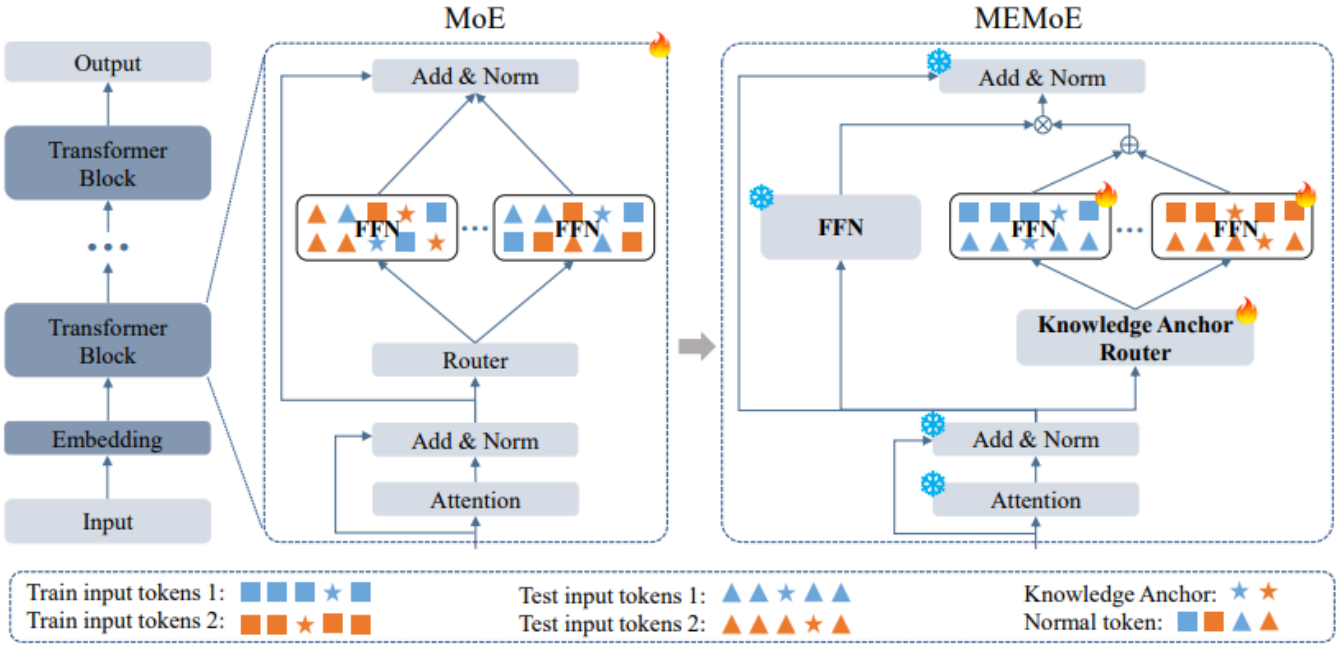


Figure 2: The architecture of MEMoE, compared with conventional MoE. Same color denote inputs requiring same knowledge. Pentagrams symbolize knowledge anchors within the input sentences, while squares and triangles represent ordinary input tokens during editing process and generality evaluation respectively. The distribution of tokens within the FFN illustrates that knowledge anchor consolidate inputs requiring same knowledge to the same experts.

MEMoE



MEMoE는 Bypass mechanism을 통해 원래의 LLM parameter를 보존하는 것으로, post-edit model의 locality를 보전하려고 함과 동시에 generality를 향상

methods

MEMoE

MEMoE는 아래와 같은 단계로 작동

1. Knowledge anchor 식별
2. 토큰 임베딩과 knowledge anchor 임베딩 결합
3. gate decision vector 계산
4. Experts output 계산
5. MEMoE layer의 최종 output 계산

MEMoE

Knowledge Anchor 식별

knowledge anchor routing을 위해 입력 문장에서 앵커를 식별

- 입력 문장에서 named entities를 앵커로 식별

예시)

“Who is the president of the United States?”라는 입력 문장에서는

“president”, ‘United States’가 knowledge anchor로 작동

MEMoE
token embedding과 결합

각 토큰의 local token representation과 anchor embedding을 결합

- anchor embedding은 모델의 임베딩 레이어에서 획득

$$R_{\text{anchor}}(x_i) = \text{concat}(x_i, \text{embed}(x_{\text{anchor}}))$$

예시)

["Who", "is", "the", "president", "of", "the", "United", "States“]

knowledge anchor 임베딩을 각 토큰 임베딩과 concat

MEMoE

Gate Decision Vector 계산

각 입력 토큰에 대해 Gate Decision Vector를 계산

- softmax와 top-k 연산을 통해, 특정 토큰을 어떤 전문가에게 할당할지를 균형 있게 배분

$$\mathcal{G} = \text{top}_k\left(\text{softmax}\left(\mathbf{W}_g \cdot R(x_i) + \epsilon\right)\right)$$

MEMoE

Expert's computation -> MEMoE output

gate decision vector G에 따라 각 experts의 출력을 가중 평균하여 최종 출력을 생성

$$h_i = \sum_{e=1}^{E} \mathcal{G}_e \cdot \mathbf{W}_e \cdot x_i$$

원래 모델의 파라미터 W_0와 전문가 모듈의 최종 출력을 결합한 것이 MEMoE layer의 최종 출력

$$h_i = \mathbf{W}_0 \cdot x_i + \lambda \sum_{e=1}^{E} \mathcal{G}_e \cdot \mathbf{W}_e \cdot x_i$$

Experiments

Datasets
- ZsRE
- COUNTERFACT

Model
- GPT2-XL
- LLaMA2-7B

Methods
- FT-L, FT-M
- LoRA
- MEMIT
- MEND
- COMEBA
- SERAC
- GRACE
- MEMoE

Experiments - (1)

Table 1: Batch editing results. **Bold** is the best result, and underline is the second-best result.

Method	Model	ZsRE				CounterFact			
		Reliability↑	Generality↑	Locality↑	Average↑	Reliability↑	Generality↑	Locality↑	Average↑
FT-L	GPT2-XL	16.85	16.34	71.55	34.91	0.27	0.34	85.18	28.60
FT-M		17.95	17.32	71.26	35.51	0.36	0.42	82.81	27.86
LoRA		30.10	29.08	80.54	46.57	5.64	3.46	69.45	26.18
MEMIT		61.19	49.97	97.51	69.56	81.01	27.67	95.80	68.16
MEND		2.16	2.11	20.34	8.20	0.13	0.03	4.22	1.46
COMEBA		82.21	66.61	99.40	82.74	88.28	40.38	97.66	75.44
SERAC		98.64	48.12	35.68	60.81	17.88	14.55	82.25	38.23
GRACE		95.56	39.76	99.93	78.41	94.23	32.56	94.58	73.79
MEMoE		**95.69**	**88.18**	**100.0**	**94.62**	**93.78**	**85.15**	**100.0**	**92.98**
FT-L	LLaMA2-7B	14.19	13.07	70.16	32.47	0.21	0.30	80.69	27.07
FT-M		16.57	15.62	70.15	34.11	0.29	0.38	81.83	27.50
LoRA		25.32	23.15	52.01	33.49	21.70	22.32	40.37	28.13
MEMIT		24.02	39.97	17.00	27.00	18.57	31.29	14.88	21.58
MEND		1.01	2.83	96.77	33.54	0.45	2.24	97.89	33.53
SERAC		89.08	16.29	81.82	62.39	80.67	17.34	82.05	60.02
GRACE		94.50	38.20	99.90	77.53	82.14	32.09	98.93	71.05
MEMoE		**100.0**	**90.30**	**100.0**	**96.77**	**99.69**	**88.30**	**100.0**	**96.33**

Experiments - (2)

Table 2: Sequential batch editing results. **Bold** is the best result, and underline is the second-best.

Method	Model	ZsRE				CounterFact			
		Reliability↑	Generality↑	Locality↑	Average↑	Reliability↑	Generality↑	Locality↑	Average↑
FT-L	GPT2-XL	3.79	2.48	6.60	4.29	1.00	1.00	6.00	2.67
FT-M		8.92	8.41	6.22	7.85	4.00	3.50	5.50	4.33
LoRA		0.96	1.29	0.03	0.76	0.50	0.02	0.50	0.34
MEMIT		34.88	32.96	70.74	46.19	56.00	37.00	31.00	41.33
MEND		20.95	18.29	93.69	47.01	0.01	0.00	0.08	0.03
COMEBA		66.91	56.11	97.23	73.42	86.00	38.00	59.00	61.00
SERAC		**100.0**	36.03	35.95	57.33	15.41	12.96	81.00	36.46
GRACE		**100.0**	0.04	**100.0**	66.68	**100.0**	0.40	**100.0**	66.80
MEMoE		74.69	**58.18**	98.93	**77.27**	88.12	**54.78**	99.45	**80.78**
FT-L	LLaMA2-7B	2.33	1.59	6.67	3.53	0.23	0.18	10.66	3.69
FT-M		6.72	4.37	7.78	6.29	0.33	0.70	8.54	3.19
LoRA		0.35	1.89	0.07	0.77	0.31	0.99	0.17	0.49
MEMIT		12.29	29.95	15.38	19.21	10.37	32.96	12.79	18.71
SERAC		67.78	33.98	34.55	45.44	20.21	14.05	34.90	23.05
GRACE		**89.70**	0.09	98.32	62.70	**74.41**	1.03	96.67	57.70
MEMoE		69.50	**42.63**	**99.70**	**70.61**	54.62	**43.40**	**99.69**	**65.9**

Experiments - (3)

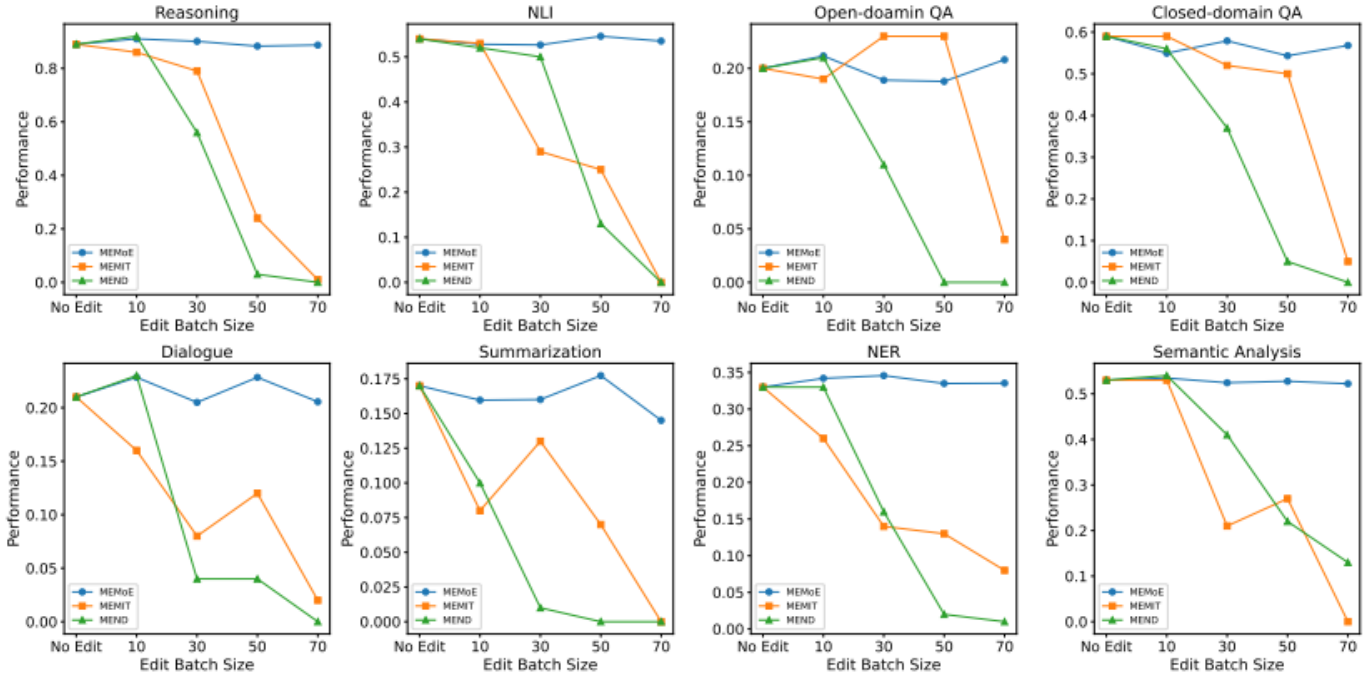


Figure 3: Performance on general tasks of edited models using MEMoE, MEMIT and MEND, with different batch sizes for edits.

Experiments - (4)

Table 3: Expert Specific Experimental Results. "Dynamic" indicates that we dynamically select input data, while "Static" refers to evaluation conducted using models trained on previous experiments. "id" stands for group id. "Similar" refers to similar knowledge, and "same" refers to same knowledge.

Type	Number of data from each group					Consistency↑		Generality↑	Reliability↑	Locality↑
	id=1	id=2	id=3	id=4	id=5	similar	same			
Dynamic	10	10	10	10	10	63.09	73.77	88.10	99.84	100
	20	10	10	10	0	65.47	73.79	89.05	99.84	100
	30	10	10	0	0	69.67	76.33	90.12	99.84	100
	40	10	0	0	0	74.63	80.13	92.01	99.84	100
	50	0	0	0	0	**79.69**	**84.82**	**94.78**	99.84	100
Static	-	-	-	-	-	65.14	77.77	90.00	99.84	100

다양한 지식 분야의 입력을 다룰 때, experts routing의 행동을 분석하기 위해
이전 실험에서 학습된 MEMoE 모듈을 기반으로 정적 분석을 수행

입력 지식 범주의 집중도가 높아질수록 Consistency, Generality 향상

마무리...

기존 방법이 잘 안 다루던 연관된 지식까지 개선하기 위해서 KG를 결합하기도 하고

MoE처럼 많이 쓰이는 방법론을 결합하는 등 아직 파 볼만한 것들이 많다는 생각을 하게 됨

Q&A

감사합니다