Lifelong Knowledge Editing

전용찬

Natural Language Processing
& Artificial Intelligence

Knowledge Editing Overview

- Knowledge Editing 이란?
  - 모델이 가지고 있는 지식을 수정하는 것
  - 즉, (Subject, Relation)가 주어졌을 때 원하는 Object로 답할 수 있도록 만드는 것
  - 예시: 미국의 현재 대통령은? ~~조 바이든~~ 트럼프

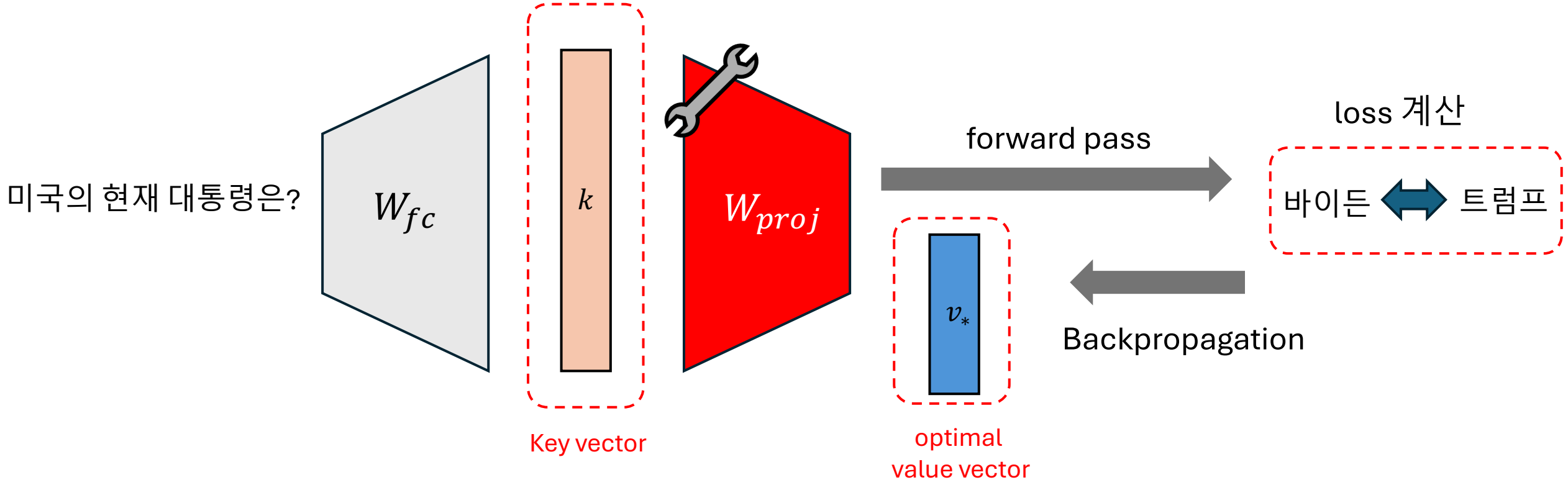
    subject relation object

- Locate-and-edit Knowledge Editing:
  - MLP 가 지식을 저장하고 있다는 이론을 기반으로 특정 layer의 MLP만 업데이트하여 지식 편집을 하는 것
  - 대표적인 방법: ROME, MEMIT,…

- 소개할 논문:
  1. PRUNE: PERTURBATION-RESTRAINED SEQUENTIAL MODEL EDITING
  2. AlphaEdit+: MODEL EDITING IN THE PRESENCE OF CONFLICTING AND INCONSISTENT KNOWLEDGE

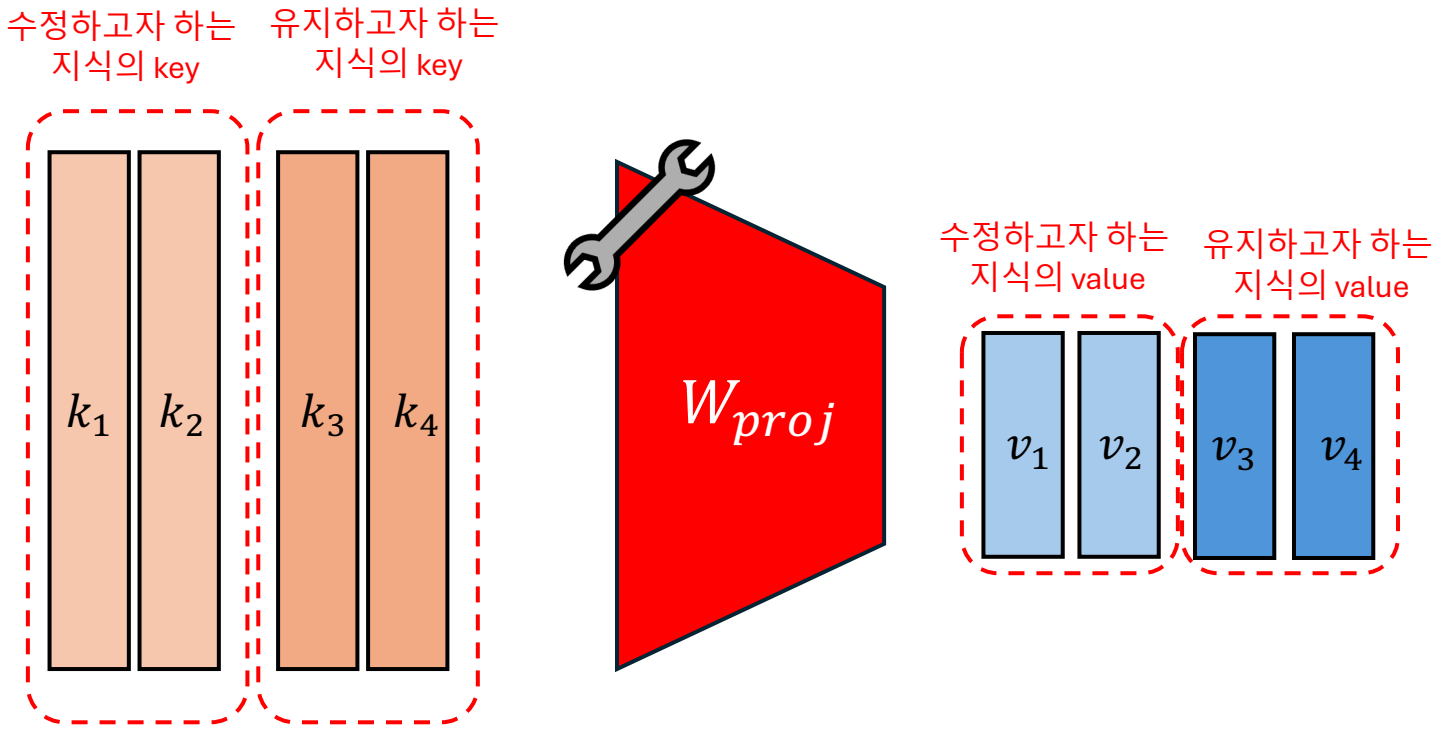
Review of Locate-and-edit based Knowledge Editing

1. LLM의 MLP layer가 지식을 포함하고 있다는 Linear associative memory 이론을 기반으로 하는 방법
2. MLP layer의 projection up layer가 지식을 활성화시키는 Key vector를 만들고 projection down layer는 Key vector를 바탕으로 지식이 포함된 Value vector를 만듬



Review of Locate-and-edit based Knowledge Editing

3. 편집하고자 하는 지식과 유지하고자 하는 지식을 이어 붙여서 matrix로 만든 후 Least square로 최적의 W^*_{proj} 를 구함



$$W^*_{proj} = \mathrm{a}rgmin\,(V - W_{proj}K)$$

Review of Locate-and-edit based Knowledge Editing

4. 우리가 원하는 건 새로운 지식을 online 으로 간단하게 추가하는 것이므로 새로운 지식을 들어왔을 때 기존의 W_{proj}에 업데이트 되는 양 ΔW_{proj} 을 구하는 것이 목적임

$$\Delta W_{proj} = (V_E - WK_E)K_E^T(K_oK_0^T + K_EK_E^T)^{-1}$$

K_o: 유지하고 싶은 지식의 Key들

K_E: 수정하고 싶은 지식의 Key들

V_E: 수정하고 싶은 지식의 Value들

PERTURBATION-RESTRAINED SEQUENTIAL MODEL EDITING

ICLR 2025

Natural Language Processing
& Artificial Intelligence

PRUNE

1. 기존의 locate-and-update 방식의 지식편집은 지식을 수정할수록 유지하고 싶은 지식이 유지되지 않고 다른 task에서의 성능이 하락하는 현상이 발생함

2. 이러한 현상이 발생하는 원인을 “Matrix Perturbation Theory” 관점에서 분석하고, 이를 완화할 수 있는 regularization 방법을 제시함

Analysis in failure of lifelong editing

1. **Matrix Perturbation Theory:** 행렬이 수정될 때 행렬의 고유값과 고유벡터가 어떻게 변화하는지에 대한 이론
2. **Matrix Perturbation Theory에서 중요하게 다뤄지는 것:** condition number
3. **Condition number:** 최적화 문제에서 함수가 예측한 결과값의 에러에 얼마나 민감한지 나타내는 척도이다.
   행렬 A의 condition number는:

$$\kappa(A) = \|A^{-1}\| \, \|A\| = \frac{\sigma_{max}}{\sigma_{min}}$$

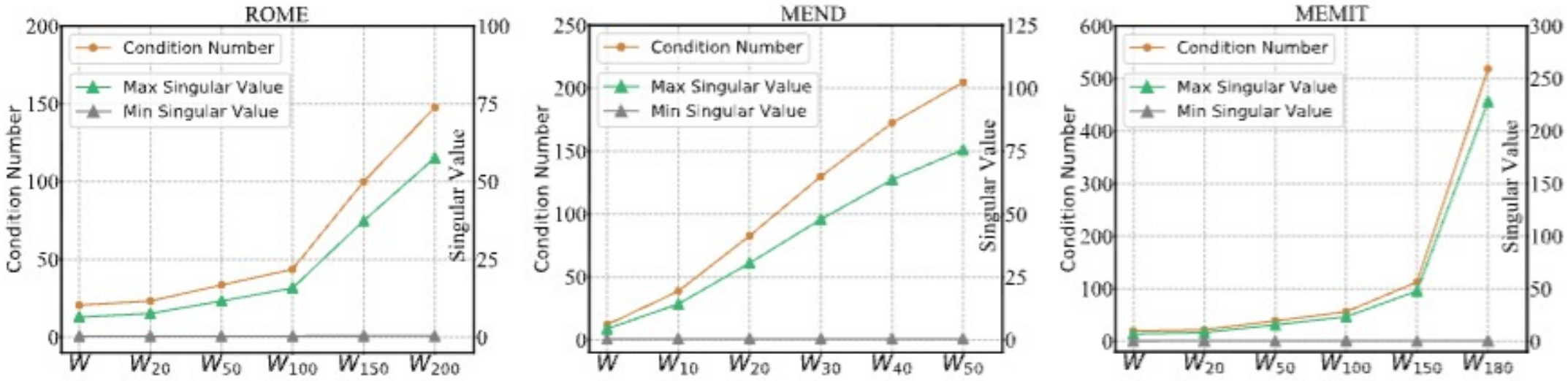
σ_{max}: 행렬 A의 가장 큰 singular value

σ_{min}: 행렬 A의 가장 작은 singular value

Analysis in failure of lifelong editing

1. Knowledge editing 방법인 ROME, MEMIT과 MEND로 각각 지식 편집을 할 때 W_{proj} 의 max와 min singular value, condition number의 변화를 확인해 보았는데, max singular value와 condition number가 계속 증가함.

2. 즉, 지식 편집을 할 수록 W_{proj} 가 에러에 더 민감해져 더 과감하게 W_{proj}를 업데이트 하게 되고, 따라서 다른 지식과 task에도 영향을 미치게 됨



PRUNE: Perturbation Restraint on Upper bouNd for Editing

1. 기존 분석에서 condition number가 증가하는 것은 max singular value가 증가하기 때문이라는 알아냄
2. 따라서 ΔW_{proj} 의 max singular value의 크기를 제한하는 regularization 방법을 제안함:

$$\overline{\sigma}_i = \begin{cases} F(\hat{\sigma}_i), & \text{if } \hat{\sigma}_i > max\{\sigma_i\}, \\ \hat{\sigma}_i, & \text{if } \hat{\sigma}_i \leq max\{\sigma_i\}. \end{cases}$$

$$F(\hat{\sigma}_i) = \log_\alpha(\hat{\sigma}_i) - \log_\alpha(max\{\sigma_i\}) + max\{\sigma_i\}$$

$max\{\sigma_i\}$: 지식편집을 하기 전 W^0_{proj} 의 max singular value

$\hat{\sigma}_i$: ΔW_{proj} 의 i번째 singular value

Experiment

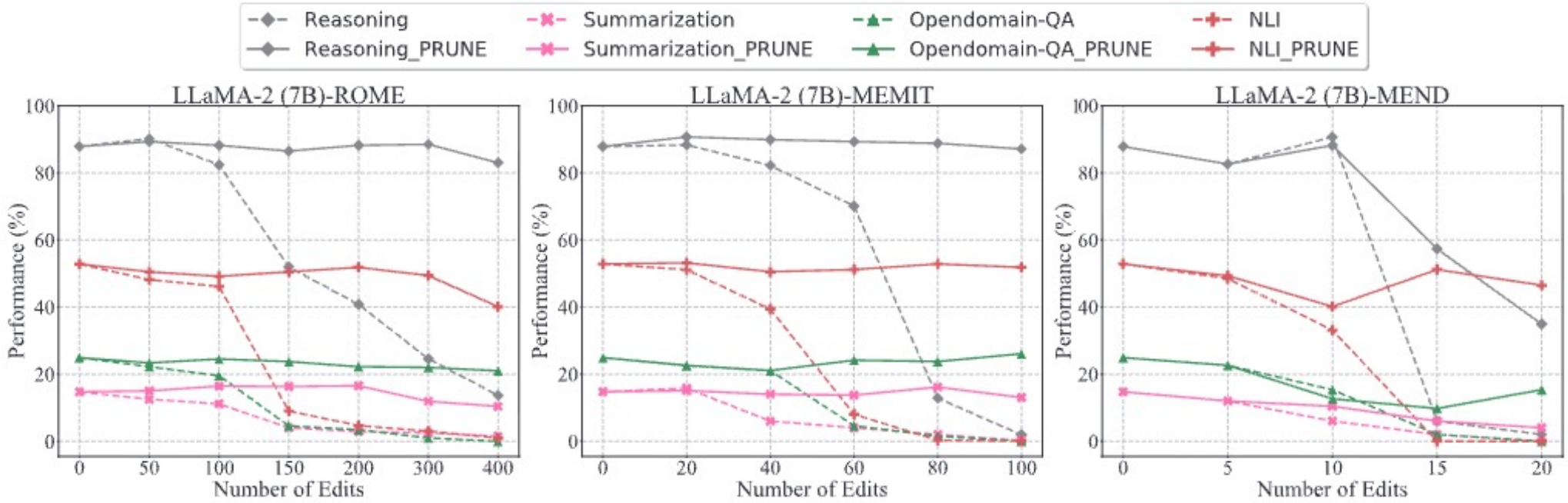
1. Baseline:
    1. ROME: 한개의 지식을 편집함
    2. MEMIT: 여러 개의 지식을 배치로 편집함
    3. MEND: weight를 얼마나 업데이트 할 지 결정하는 hypernetwork를 통해 지식을 편집함

2. Model:
    1. GPT-2-XL
    2. LLaMA-2
    3. LLaMA-3

3. Evaluation Metric:
    1. Efficiency: 수정하고 싶은 지식을 얼만큼 잘 수정하는지
    2. Generality: 수정하고 싶은 지식을 다르게 표현했을 때도 이를 반영하여 잘 대답하는지
    3. Specificity: 다른 지식은 건드리지 않는지
    4. Performance: 위 metric들의 평균

Experiment

1. PRUNE을 각각의 지식 편집 방법에 적용하였는데, 적용하지 않았을 때보다 성능이 좋게 유지됨
2. MEND는 다른 지식 편집 방법에 비해 PRUNE의 효과가 잘 나타나지 않음



Experiment

1. ROME으로 지식 편집 후에 다른 task에서 성능을 확인해보았는데, PRUNE을 적용하자 성능이 크게 유지가 됨
2. 마찬가지로 지식 편집을 평가하는 4개의 metric 모두에서 PRUNE을 적용하자 성능이 크게 유지가 됨

Table 2: Evaluation results (%) of LLaMA-2 (7B) edited by ROME on the ConceptEdit dataset.

Mode		General Abilities				Editing Performance			
Method	Edits	Reasoning	Summa	Open-QA	NLI	Efficacy	General	Locality	Instance
ROME	20	75.13	11	6.50	24.7	49.15	52.58	35.68	25
	50	20.67	4.90	1.50	0.7	55.42	49.45	19.94	12
	100	12.29	4.7	0.77	0	28.25	30.18	5.68	10
	200	0	4.62	0	0	10.14	8.65	5.31	-8.99
ROME+PRUNE	20	89.38	14.34	23.37	63.54	75.66	58.35	71.7	25
	50	85.15	14.06	25.29	50.52	56.51	45.55	73.16	8
	100	90.78	13.75	21.46	53.17	46.22	42.26	64.06	20
	200	72.9	10.55	22.22	46.15	35.82	34.95	46.65	32

AlphaEdit+:
MODEL EDITING IN THE PRESENCE OF CONFLICTING AND INCONSISTENT KNOWLEDGE

ICLR 2026 under review

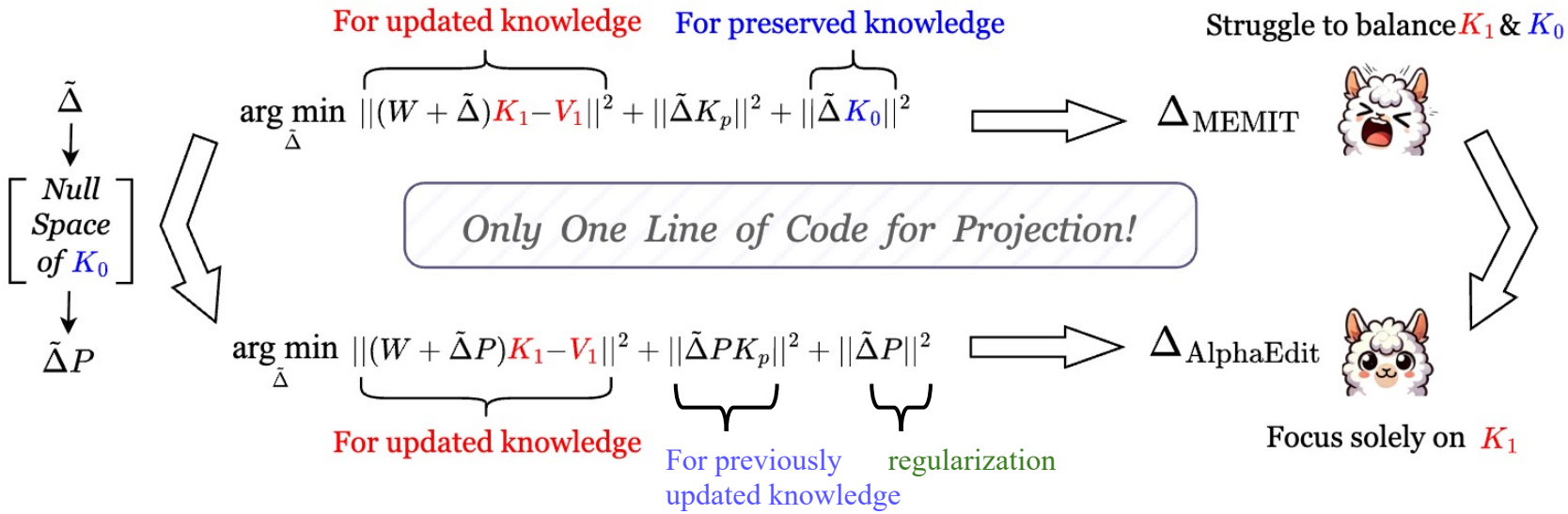
Natural Language Processing
& Artificial Intelligence

Review of AlphaEdit

1. 기존의 locate-and-edit 방식의 지식편집은 지식을 수정할수록 유지하고 싶은 지식이 유지되지 않는 현상이 발생함

2. 이를 해결하기 위해 유지하고 싶은 지식의 null space, 즉 유지하고 싶은 지식과 orthogonal 한 space으로 ΔW_{proj} 를 projection 시킨 후에 업데이트를 진행함.

3. 즉, null space의 projection matrix를 P라고 하면 업데이트 되는 정도는 $\Delta P \coloneqq \Delta W_{proj} P$



AlphaEdit+

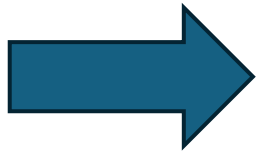
1. 그렇지만.. AlphaEdit도 다음과 같은 한계가 있다는 것을 밝혀냄:

    1. 유지하고 싶은 지식이 유지되지 않음
    2. 지식을 batch로 편집할 때 지식 사이에 충돌이 발생함
    3. 기존에 편집한 지식이 유지되지 않음

2. 이러한 한계가 AlphaEdit이 지식을 지나치게 강하게 업데이트를 수행 (즉, 오버피팅) 하기 때문이라고 가정함
3. 이를 해결하기 위해 다음의 방법들을 iterative 하게 수행할 것을 제안함

1. 유지하고 싶은 지식이 유지되지 않음
2. 지식을 batch로 편집할 때 지식 사이에 충돌이 발생함
3. 기존에 편집한 지식이 유지되지 않음

문제



1. P를 relax 함
2. Value vector들을 smoothing 함
3. 기존에 편집한 지식과의 충돌하는 정도를 고려하여 업데이트 되는 양의 scale을 조절함

해결책

Methodology

***P*를 relax 함**

1. P 에 iterative 하게 노이즈를 추가하면서 perturb 함
2. 이때, 노이즈는 P를 SVD로 분해한 다음 가장 낮은 고유 값에 해당하는 고유 벡터들을 차례로 loss가 특정 임계치를 넘지 않을 때까지 더해줌

for *each* $\boldsymbol{u}_l \in \check{\mathbf{U}}$ *in order* **do**
  $\tilde{\mathbf{P}} \leftarrow \tilde{\mathbf{P}} + \boldsymbol{u}_l \boldsymbol{u}_l^\top$;
  Recompute the objective in Eq. 15 with $t = 0$;
  if *loss reduction* $< \epsilon$ **then** $\mathbf{P}^*_{\text{mod}} \leftarrow \mathbf{P} + \tilde{\mathbf{P}} - \boldsymbol{u}_l \boldsymbol{u}_l^\top$; break;

Methodology

Value vector들을 smoothing 함

1. Value vector들을 다른 value vector와 loss가 특정 임계치를 넘지 않을 때까지 섞어줌

$$\boldsymbol{v}_{1,i}^{t} = \boldsymbol{v}_{1,i} + \beta_i(t)\left(\boldsymbol{v}_{0,i} - \boldsymbol{v}_{1,i}\right), \qquad \beta_i(t) = \begin{cases} 0, & \text{if } \|\boldsymbol{r}_i\| \leqslant \tau_r, \\ \dfrac{T-t}{2T}, & \text{otherwise,} \end{cases}$$

$$\text{with } \boldsymbol{r}_i = \boldsymbol{v}_{1,i} - \mathbf{W}\boldsymbol{k}_i$$

Methodology

기존에 편집한 지식과의 충돌하는 정도를 고려하여 업데이트 되는 양의 scale을 조절함

1. 지식을 업데이트할 때 이미 수정한 지식과의 유사도 (Key vector들의 벡터곱) 를 구해 충돌이 많이 일어나는 지식은 작게, 충돌이 적게 일어나는 지식은 크게 가중치를 줘서 업데이트함.

$$\left\| \tilde{\Delta} \mathbf{P}_{\text{mod}} \mathbf{K}_p \mathbf{\Lambda}_p^{1/2} \right\|_F^2$$

$\mathbf{P}_{\text{mod}}$: Relax한 null space projection 행렬

$\mathbf{K}_p$: 기존에 수정한 지식

$\mathbf{\Lambda}_p$: 가중치 행렬

Experiment

1. Baseline:
    1. MEMIT: 여러개의 지식을 배치로 편집함
    2. RECT: ΔW_{proj}의 원소 중에 큰 원소만 고려하여 지식 편집을 함
    3. PRUNE
    4. AlphaEdit

2. Model:
    1. GPT2-XL
    2. GPT-J

3. Evaluation Metric:
    1. Efficiency: 수정하고 싶은 지식을 얼만큼 잘 수정하는지
    2. Generality: 수정하고 싶은 지식을 다르게 표현했을 때도 이를 반영하여 잘 대답하는지
    3. Specificity: 다른 지식은 건드리지 않는지
    4. Score: 위 metric들의 평균

Experiment

1. AlphaEdit에 비해서 성능이 상승했으나 크게는 향상하지 않음

Table 1: Mean conflict/inconsistency levels and overall results on five datasets. Best per block in **bold**.

Dataset	Method	GPT2-XL (1.5B)				GPT-J (6B)			
		Eff↑	Gen↑	Spe↑	Score↑	Eff↑	Gen↑	Spe↑	Score↑
ZsRE	MEMIT	87.22	83.21	26.41	65.61	98.97	98.54	27.73	75.08
$\text{avg}(s_{K_0}) = 0.53$	RECT	77.90	69.85	25.60	57.78	96.95	93.07	28.04	72.69
$\text{avg}(s_{K_p}) = 0.62$	PRUNE	84.85	82.09	27.47	64.80	96.04	95.84	**34.17**	75.35
$\text{avg}(\|r\|) = 21.6$	AlphaEdit	97.85	93.26	26.56	72.56	99.66	**99.15**	27.70	75.50
	AlphaEdit$^+$	**98.81**	**94.32**	**27.73**	**73.62**	**99.76**	99.09	32.42	**77.09**
Counterfact	MEMIT	87.00	48.75	**12.95**	49.57	92.50	67.50	14.35	58.11
$\text{avg}(s_{K_0}) = 0.23$	RECT	67.00	31.75	12.00	36.92	96.00	48.75	14.50	53.08
$\text{avg}(s_{K_p}) = 0.47$	PRUNE	90.00	69.25	10.40	56.55	93.32	**72.50**	12.65	59.49
$\text{avg}(\|r\|) = 88.54$	AlphaEdit	95.50	66.75	10.75	57.67	97.50	70.75	14.15	60.80
	AlphaEdit$^+$	**96.63**	**69.50**	11.45	**59.19**	**98.78**	71.00	**14.75**	**61.51**

Experiment

1. 지식을 수정해도 다른 Task들 (MMLU, MRPC 등...)의 성능의 하락이 크지 않음

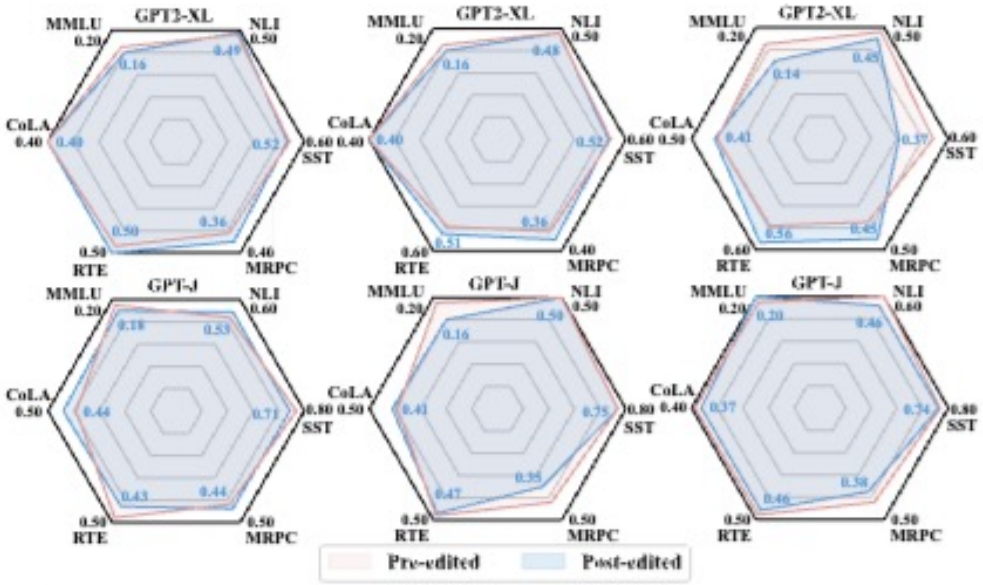


Figure 4: The impact of AlphaEdit$^+$ on the general F1 scores of two models, evaluated on the AlphaSet, ZsRE, and CounterFact datasets. Best viewed in color.

1. 해당 논문에서 제시한 방법들을 하나씩 제거하자 성능이 하락함. 즉, 제시한 모든 방법이 효과가 있음

Table 3: Comprehensive ablation analysis of AlphaEdit$^+$ on AlphaSet: optimal results per metric highlighted in **Bold**.

Variant	GPT2-XL (1.5B)				GPT-J (6B)			
	Eff↑	Gen↑	Spe↑	Score↑	Eff↑	Gen↑	Spe↑	Score↑
AlphaEdit$^+$	**93.33**	**82.69**	**26.21**	**67.41**	**96.25**	**87.84**	**29.50**	**71.20**
w/o $\tilde{\mathbf{P}}$	92.60	82.61	25.85	67.02	95.75	87.42	29.29	70.82
w/o $\mathbf{\Lambda}_p$	93.31	81.90	26.19	67.13	95.62	86.96	28.94	70.51
w/o Smoothing	93.14	80.43	25.77	66.45	95.25	87.15	29.39	70.60

Conclusion

1. 효과적인 Lifelong editing을 위한 방법들을 제시하는 PRUNE과 AlphaEdit+를 소개함

2. PRUNE을 분석하는 과정과 제시한 방법이 깔끔하고 직관적이나, AlphaEdit+은 다소 난해하고 복잡한 방법에 비해 얻을 수 있은 성능 향상은 상당히 미미함

3. Lifelong editing의 핵심은 LLM이 작동하는 과정을 분석하며 우리가 변경하고 싶은 weight만을 fine-grained하게 변경하는 것이 목적이므로 연구관점에서 상당히 유망하고, 아직도 발전할 수 있는 여지가 많아 보임

감사합니다