KOREA UNIVERSITY

2025 하계세미나
MoE

어수경

Paper

- **Self-MoE: Towards Compositional Large Language Models with Self-Specialized Experts (ICLR 2025 Poster)**

- **Your Mixture-of-Experts LLM Is Secretly an Embedding Model for Free (ICLR 2025 Oral)**

Paper

SELF-MOE: TOWARDS COMPOSITIONAL LARGE LANGUAGE MODELS WITH SELF-SPECIALIZED EXPERTS

Junmo Kang* Georgia Tech

Leonid Karlinsky MIT-IBM Watson AI Lab

Hongyin Luo MIT

Zhen Wang UCSD

Jacob Hansen MIT

James Glass MIT

David Cox MIT-IBM Watson AI Lab

Rameswar Panda MIT-IBM Watson AI Lab

Rogerio Feris MIT-IBM Watson AI Lab

Alan Ritter Georgia Tech

Introduction

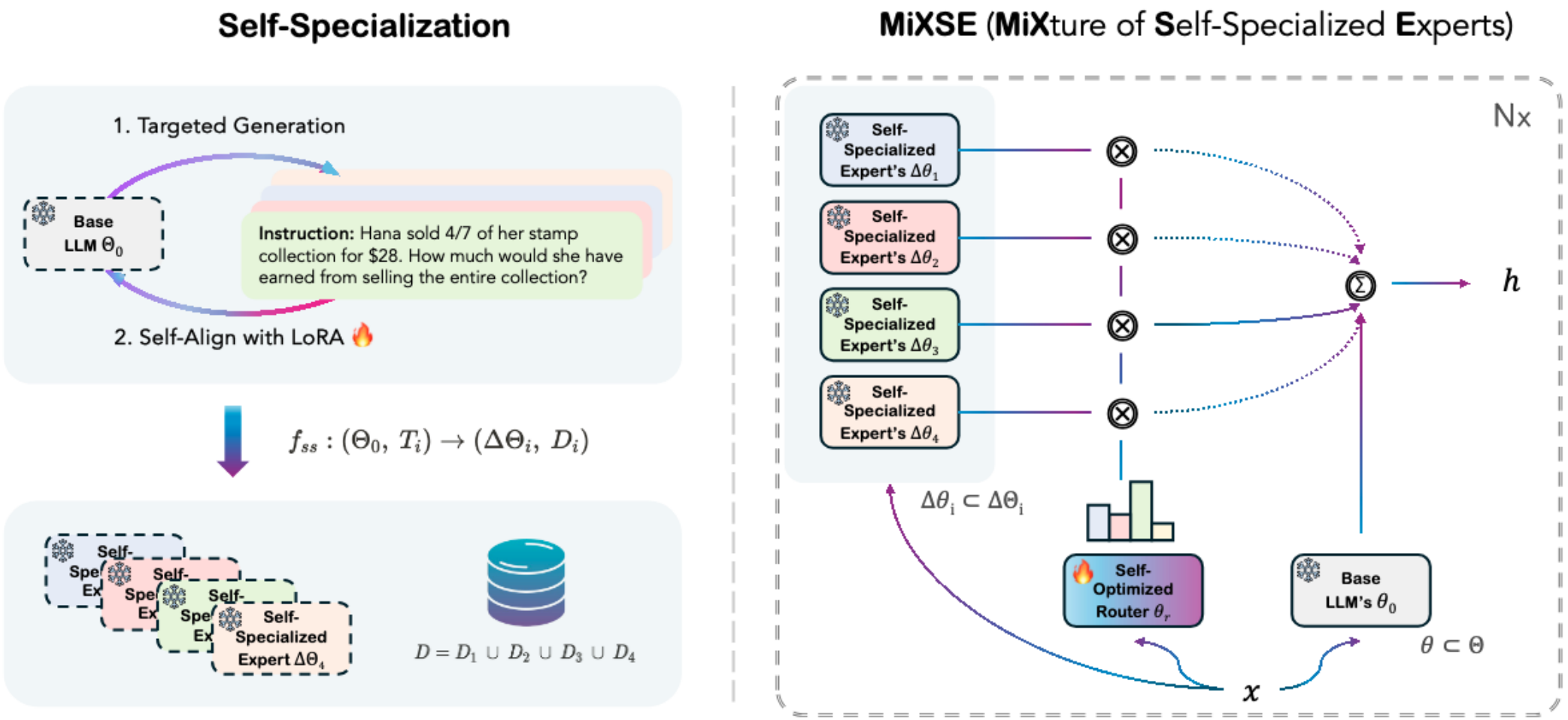
- domain-specific, expert-level tasks들에 대한 관심으로 인한 specialized LLM의 높은 수요
- 효과적인 튜닝을 위해서는 고퀄리티의 human-labeled data를 요구
  → 비용 및 확장성 측면에서의 challenge, 특히나 expert가 부족한 경우에는 더 큰 한계
- 이를 위해 등장한 것이 Self-specialization: self-generated synthetic data를 활용하여 models를 학습
- 이들은 target expert domain에서는 성능이 좋지만 out-of-domain에서는 낮은 성능
  → poor generalization capability

Question: *How can we build compositional LLMs that enjoy versatile expertise, while using minimal resources?*

→ Self-MoE 제안: MiXSE (MiXture of Self-specialized Experts)

MiXSE

- MiXSE 구조



MiXSE

1. Building expert modules through self-specialization: target expertise에 대한 self-specialized modules 생성

- Targeted generation: synthetic instruction-response data construction

- **(1) Seed Construction:**
  - Target domain과 관련된 100개 정도의 적은 양의 seed examples 준비
  - Seed examples를 demonstration처럼 활용, 이를 바탕으로 synthetic variation 생성
- **(2) Instruction Brainstorming:**
  - (1) 의 in-context를 바탕으로 Base LLM을 활용해 instruction과 corresponding input contexts를 새롭게 생성
- **(3) Response Generation:**
  - 생성된 instruction에 대해 response 생성.
  - Seed instruction-response pairs를 in-context demonstrations로 활용하여 base LLM이 가진 관련 지식들을 추출 및 response generation

MiXSE

1. Building expert modules through self-specialization: target expertise에 대한 self-specialized modules 생성

- Self-align with LoRA:
  - synthetic data를 활용하여 base model에 대한 self-alignment 진행
  - separate lightweight component 생성 (LoRA 학습)
  - $\theta_{spec} = \theta_0 + \Delta\theta_i = \theta_0 + \theta_{B_i}\theta_{A_i}$
  - (* θ_0→ weights at a certain layer where LoRA is attached, $\theta_{B_i} \in \mathbb{R}^{d \times rank}, \theta_{A_i} \in \mathbb{R}^{rank \times k}$)
  - (* update → $\Delta\Theta_i = \{\Delta\theta_i^{(1)}, \Delta\theta_i^{(2)}, \dots\}$)

- Target domain (knowledge, reasoning, math, coding)을 대상으로 위 과정 반복

MiXSE

2. Mixture of self-specialized experts (MiXSE)

- Self-specialization process를 통해 특화된 각 expert module → compound system인 MiXSE로 통합 with router
- Selected expert modules의 output과 router weight를 weighted sum

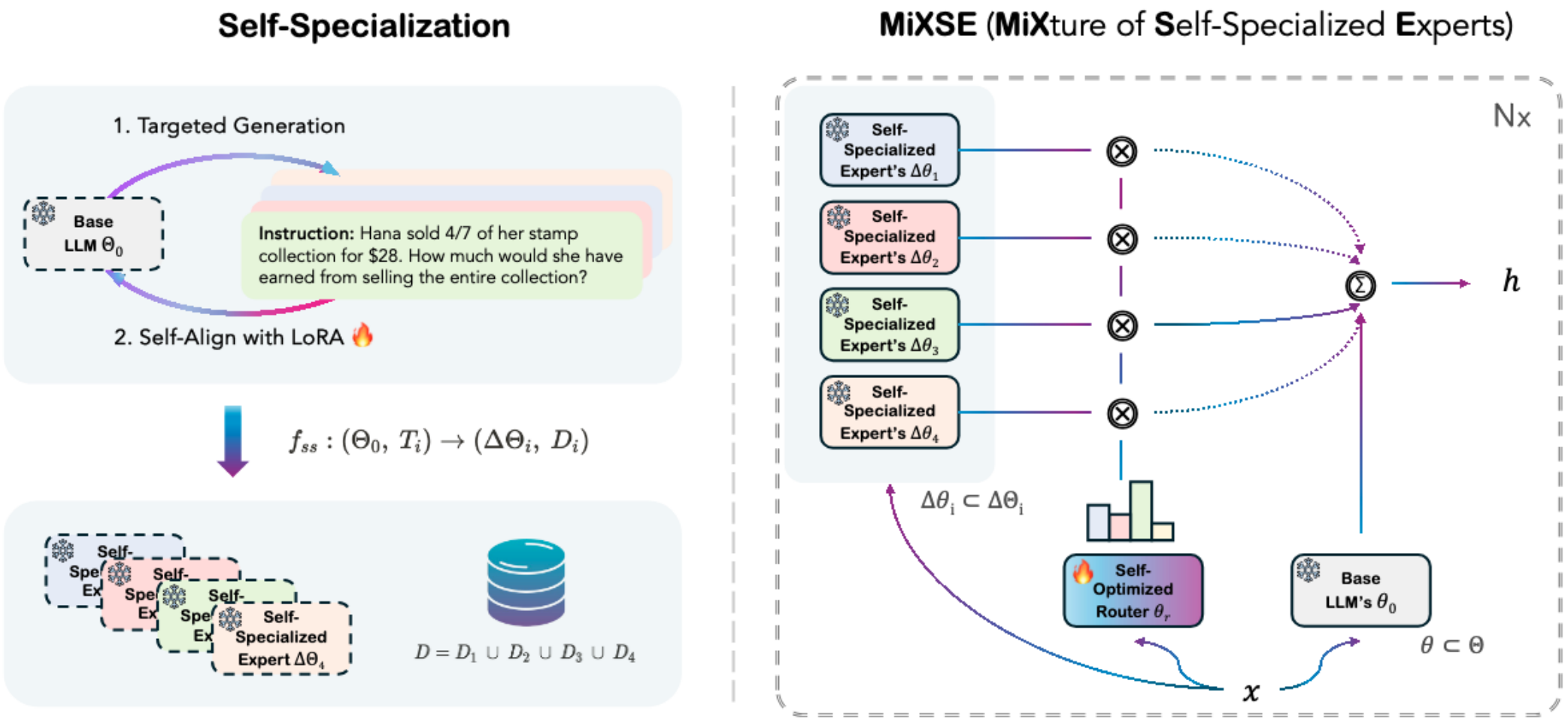
$$\mathrm{h} = \theta_0 x + \sum_{i=1}^{n} \alpha_i \, \Delta\theta_i x = \theta_0 x + \sum_{i=1}^{n} \alpha_i \, \Delta\theta_{B_i}\theta_{A_i} x,$$
$$\alpha = topk\big(softmax(\theta_r x)\big), \theta_r \in \mathbb{R}^{n \times k}$$

- Router θ_r → linear layer

$$L(\theta_r) = -\mathbb{E}_{(inst,resp)\sim D}[logP_{\Theta_0}(resp|inst; \theta_r, \{\Delta\Theta_i\}_{i=1}^{n})]$$

MiXSE

- MiXSE 구조



Experiments

Datasets

- Knowledge, reasoning, math, coding tasks
- Seed data 100 samples, 5K synthetic data for each domain

Models

- Gemma 7B / Llama2 7B & 13B / Mistral 7B / Llama3 8B
- LoRA -> 약 0.3% 파라미터 추가

Baselines

- Self-specialized models -> knowledge, reasoning, math, coding
- Instance merging (multi-task tuning) -> synthetic data를 활용하여 multi-task tuning
- TIES, DARE: multiple experts를 하나의 expert로 활용하는 weight merging methods

Experiments

Main results

Method	Active Params	Knowledge (MMLU)	Reasoning (BBH)	Math (GSM8K)	Coding (HumanEval)	Avg.
Base LLM	7B	58.4	56.1	42.5	34.1	47.8
Specialized LLM for Each Capabiility						
Knowledge Self-Spec.	7B + 0.3%	64.0	41.7	40.5	28.0	43.6
Reasoning Self-Spec.	7B + 0.3%	60.1	60.2	41.0	28.7	47.5
Math Self-Spec.	7B + 0.3%	59.3	58.9	50.0	36.0	51.1
Coding Self-Spec.	7B + 0.3%	57.2	57.2	46.0	37.2	49.4
Merging Methods						
Instance Merging	7B + 0.3%	62.6	57.6	**53.5**	36.0	52.4
TIES Merging	7B + 0.3%	63.7	56.3	38.5	32.9	47.9
DARE Merging	7B + 0.3%	37.7	59.6	45.0	34.8	44.3
MiXSE (Ours)	7B + 0.3%	**65.6** ↑7.2	**61.1** ↑5.0	52.5 ↑10.0	**37.8** ↑3.7	**54.3** ↑6.5

Category	Benchmark	Base LLM	Instance Merging	MiXSE
Target				
Academic Knowledge	MMLU	58.4	62.6	65.6
Reasoning	BBH	56.1	57.6	61.1
Math	GSM8K	42.5	53.5	52.5
Coding	HumanEval	34.1	36.0	37.8
Target Average		47.8	52.4	54.3
Non-Target (In-Expertise)				
Math	MATH	20.7	15.3	21.4
Coding	MBPP	37.8	37.6	39.6
Non-Target (Out-of-Expertise)				
World Knowledge	Natural Questions	24.2	22.3	24.5
	TriviaQA	63.9	58.6	62.5
Commonsense	Hellaswag	80.6	78.0	80.7
	PIQA	81.1	80.1	81.2
Safety	TruthfulQA	44.7	42.2	44.3
Non-Target Average		50.4	47.7	50.6

- Base LLM 대비 Self-specialization에서 우수한 성능
- Specialized LLMs -> in-domain에서는 우수한 성능 but out-of-domain에서는 poor generalization
- MiXSE는 모든 도메인에서 일관된 우수한 성능
- Merging methods와 비교해서도 일관된 우수한 성능
- Generalizability 측면에서도 우수

Experiments

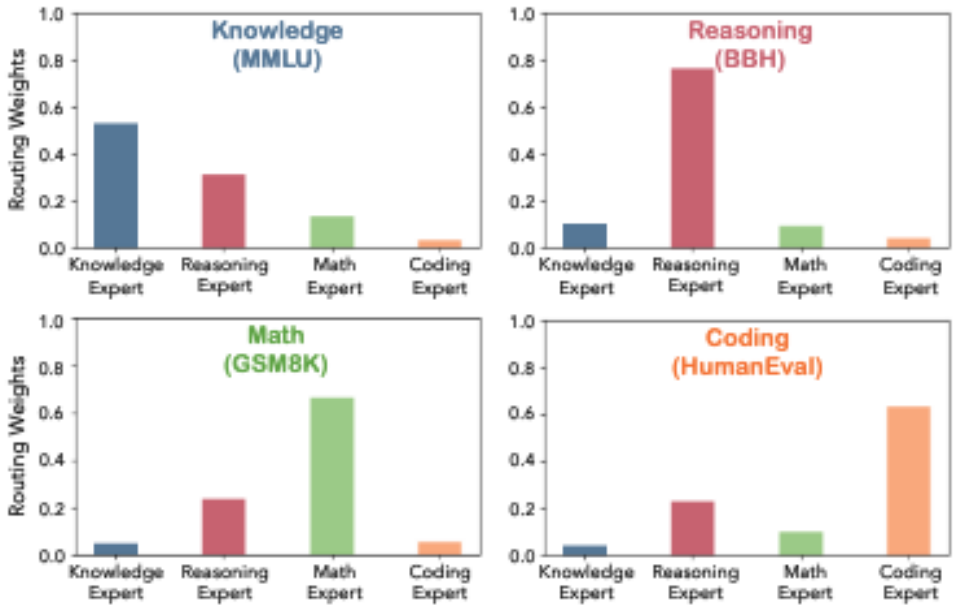
Ablation Study

Configuration	Knowledge (MMLU)	Reasoning (BBH)	Math (GSM8K)	Coding (HumanEval)	Avg.
Base LLM	58.4	56.1	42.5	34.1	47.8
Top-k Routing					
w/ Top-1 Expert	**65.6**	**61.1**	52.5	37.8	**54.3**
w/ Top-2 Experts	65.5	60.9	52.5	**38.4**	**54.3**
w/ All Experts	65.4	58.9	**54.0**	33.5	53.0
Routing Strategy					
w/o Self-Optimized Router	59.9	58.5	48.0	36.6	50.8
w/o Shared Router	59.5	59.1	50.5	32.9	50.5
Experts & Router Joint Training					
w/o Semantic Experts (Top-1)	64.5	58.1	46.0	33.5	50.5
w/o Semantic Experts (Top-2)	64.2	53.3	48.5	36.5	50.6

- Top-k routing strategy → Top1, top2에서 우수한 성능, 가장 관련있는 experts만을 선택하는 것이 효과적. 모든 expert를 사용하는 경우 오히려 noise를 주는 역할
- 다양한 configuration에서도 우수한 성능을 보이는 것은 robust함을 증명
- Random routing, layer-specific router를 적용한 경우 성능이 매우 떨어짐
- Router와 experts를 같이 학습시키는 경우 오히려 expert간 구분이 흐려짐

Experiments

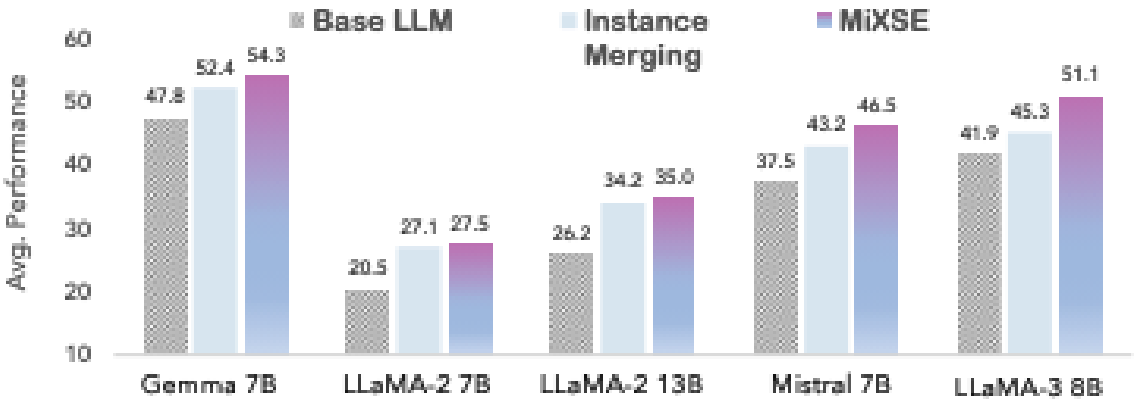
Routing Analysis



- Query와 domain experts가 올바르게 routing되었는지 분석
- 각 task마다 assign된 domain experts가 적절히 연결됨

  → 다른 domain experts들과도 연결 (특히 knowledge): 여러 도메인으로부터의 도움을 받는 시너지 효과
- System's decision → interpretable, trustworthy

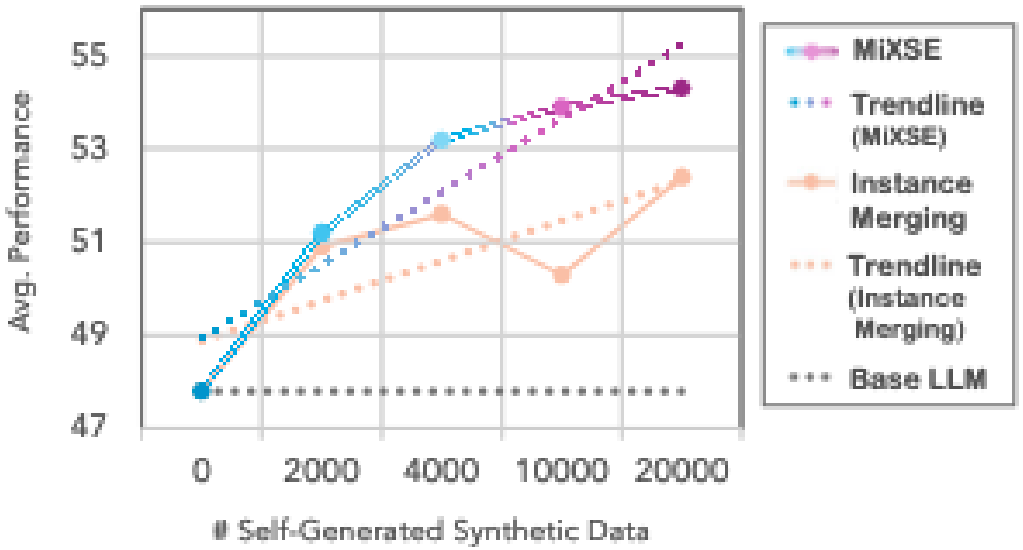
Applicability to other base LLMs



- 총 5개 모델
- 사이즈, 버전, 패밀리에 대해 모두 robust한 성능 향상

Experiments

Impact of the number of synthetic data



- Synthetic data의 수에 따른 성능 변화 양상
- 두 방법론 모두 데이터 증가에 따른 성능 향상 기록
- Performance gain은 MiXSE에서 두드러짐

Scaling the number of experts

(Knowledge, Reasoning, Math, Coding)

# Experts	Knowledge (MMLU)	Reasoning (BBH)	Math (GSM8K)	Coding (HumanEval)	Avg.
0 (Base LLM)	58.4	56.1	42.5	34.1	47.8
1 (K)	64.0	41.7	40.5	28.0	43.6
2 (K+R)	65.8	58.0	43.0	32.3	49.8
3 (K+R+M)	62.7	61.5	54.5	32.9	52.9
4 (K+R+M+C)	65.6	61.1	52.5	37.8	54.3

- Expert를 점진적으로 증가시켰을 때 성능이 향상되는 경향
- 처음에는 다른 도메인들과의 trade-off → expert를 늘릴수록 점진적으로 전체 도메인에서 성능 향상
- MiXSE의 adaptability, modularity의 장점 부각

Discussion & Conclusion

- Multiple domain experts를 생성하는 과정을 MoE 구조로 통합
- 각 expert를 튜닝한 후 router를 학습하는 방식
- 높은 도메인 특화 성능 및 일반화 성능 → 도메인 특화와 함께 각 expert 간 시너지 효과로 인한 일반화 성능 향상 역시 달성

- **Discussion on the overhead of self-moe**
- LoRA를 사용한다는 점 → 추가되는 파라미터의 수는 매우 적으며, 이마저도 sparsely activate됨
- FFN의 학습을 필요로하는 Mixtral, BTX에 비해 효율적

→ Better scalability, resource efficiency, expert scalability 측면에서 우수

Paper

YOUR MIXTURE-OF-EXPERTS LLM IS SECRETLY AN EMBEDDING MODEL FOR FREE

Ziyue Li, Tianyi Zhou
Department of Computer Science
University of Maryland, College Park
{litzy619,tianyi}@umd.edu
Project: https://github.com/tianyi-lab/MoE-Embedding

Introduction

- MoE는 dynamic routers를 통해 입력의 unique characteristics에 따라 연산을 조정함으로써 효율성과 효과를 모두 최적화
- 그러나 LLMs 또는 MoE LLMs는 생성 태스크에서 성능이 좋지만 decoder-only architecture라는 점에서 embedding model로 활용되기에는 제약이 있음
  - hidden state는 입력 토큰의 key features와 모든 정보들을 커버할 수 없음 (입력의 미묘한 의미적 차이를 잘 감지 못함)
  - next output token의 정보에 bias되어있음
  - last token을 활용해도 심지어는 작은 인코더 기반 모델보다 성능이 낮게 나오기도 함
- 학습을 통해 embedding models의 capability를 높일 수 있으나, 이는 추가적인 학습을 요구
- 그렇다면 LLM을 추가 학습 없이 바로 embedding모델로 쓸 수 있는 방법이 있을까?

  → 제안: **MoE Embedding (MoEE)**

Mixture-of-Experts Embedding (MoEE)

- **MoE routing weights (RW) as embedding**

- MoE model at layer l consists of $N^{(l)}$ experts, denoted by $E_i^{(l)}$, where $i = 1,2,\dots,N^{(l)}$
- 각 expert는 레이어마다 특정 input의 characteristics에 특화된 sub-network
- Dynamic routing mechanism of MoE: A gating function $\boldsymbol{g}^{(l)}\left(\boldsymbol{H}^{(l)}\right) \in \mathbb{R}^{N^{(l)}}$

  - 주어진 입력 토큰을 가장 우수하게 처리하는 experts를 선택. Specifically,
  - logits $\boldsymbol{z}^{(l)}\left(\boldsymbol{H}^{(l)}\right)$ 에 대해 softmax: $g_{\mathrm{i}}^{(l)}\left(\boldsymbol{H}^{(l)}\right) = \frac{exp(\boldsymbol{z_i^{(l)}}(\boldsymbol{H}^{(l)}))}{\sum_{j=1}^{N^{(l)}} exp\left(\boldsymbol{z_j^{(l)}}(\boldsymbol{H}^{(l)})\right)}$

  → probability distribution으로 주어지는 routing weights $g_{\boldsymbol{i}}^{(l)}\left(\boldsymbol{H}^{(l)}\right)$ 를 얻게 됨

  - Expert final output: $\sum_{i=1}^{N^{(l)}} g_i^{(l)}\left(\boldsymbol{H}^{(l)}\right) E_i^{(l)}(\boldsymbol{H}^{(l)})$

- RW 기반 임베딩 (concatenation): $e_{RW} = \left[\boldsymbol{g}^1\left(\boldsymbol{H}^{(1)}\right); \boldsymbol{g}^{(2)}\left(\boldsymbol{H}^{(2)}\right); \dots; \boldsymbol{g}^{(L)}\left(\boldsymbol{H}^{(L)}\right)\right] \in \mathbb{R}^{\sum_{l=1}^{L} N^{(l)}}$

Mixture-of-Experts Embedding (MoEE)

- **MoE routing weights (RW) as embedding**

- RW 기반 임베딩 (concatenation): $e_{RW} = \left[\boldsymbol{g}^{1}\left(\boldsymbol{H}^{(1)}\right); \boldsymbol{g}^{(2)}\left(\boldsymbol{H}^{(2)}\right); \ldots; \boldsymbol{g}^{(L)}\left(\boldsymbol{H}^{(L)}\right)\right] \in \mathbb{R}^{\sum_{l=1}^{L} N^{(l)}}$

즉, **모든 layer**의 routing weights를 활용하여 embedding을 생성

- **RW를 임베딩으로 활용**하면, **중간 레이어마다 선택된 expert 패턴** (중간 추론 선택) 까지 함께 반영
  → 모든 layer 깊이에서 **shallow and deep contextual features를 모두 고려한, 섬세하고 richer representation**을 얻을 수 있음
  → low-level과 high-level에서의 입력 디테일에 대한 sensitivity를 요구하는 임베딩 모델에 적합

Mixture-of-Experts Embedding (MoEE)

- **Comparative & Complementary analysis of Routing Weights & Hidden State**

- 그렇다면 이 RW embedding만을 활용할 것인가? Nope.
- RW과 HS embedding간 complementary한 역할을 하는 것을 밝힌 후 이 둘을 조합해서 활용

- 저자들의 hypothesis: **HS와 RW는 입력의 서로 다른 부분들을 포착할 수 있어 서로 complementary한 정보를 제공**

(1) 두 임베딩을 clustering한 후 cluster structures간 correlation을 정량화

(1-1) BERTopic framework를 활용해 각 cluster의 주요 topic들을 추출

(2) Semantically 유사한 문장 쌍 기반 분석

Mixture-of-Experts Embedding (MoEE)

- **Comparative & Complementary analysis of Routing Weights & Hidden State**

(1) RW and HS embedding exhibit distinct clustering behaviors and encode different topics:

- Clustering 결과 RW와 HS embedding 간 Jaccard similarity는 0.06, exact matching은 45.54%로 매우 낮음
  → 입력을 구조화하는 데 있어 두 임베딩은 서로 구별됨

- 각 클러스터 간 topic들 역시 다른 주제들을 지니고 있음

→ 입력 데이터에 대해 divergent한 측면을 반영하고 있음

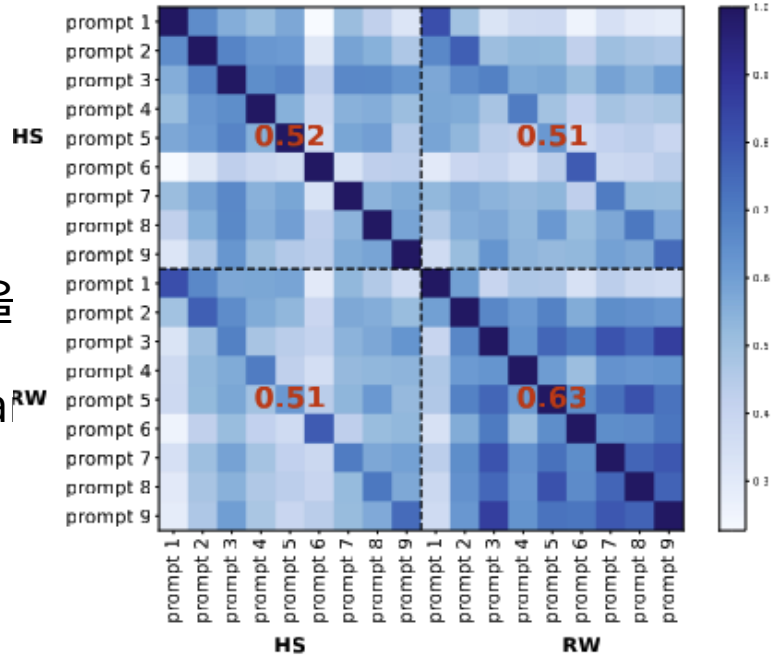
Mixture-of-Experts Embedding (MoEE)

- **Comparative & Complementary analysis of Routing Weights & Hidden State**

(2) Complementary nature of RW and HS embedding:

- STS12 데이터셋에 대해 각 문장쌍마다 HS, RW 임베딩을 생성
- Prompt에 따른 성능 변동성을 고려해 총 9개 prompts에 대해 위 임베딩들을 생성
- 문장1(HS)-문장2(HS), 문장1(RW)-문장2(RW) 간 prompt1~promp9 조합별 similarity score를 생성
- 두 임베딩으로부터 생성된 Similarity scores 간 correlation 확인

→ HS-RW 임베딩 간 similarity score correlation은 0.51로 낮음
→ RW와 HS는 데이터들 중에서도 연관성 없는 측면들을 각각 포착함
→ 서로 다른 trend를 보인다는 것은 높은 complementary nature에 대한 evidence가 될 것



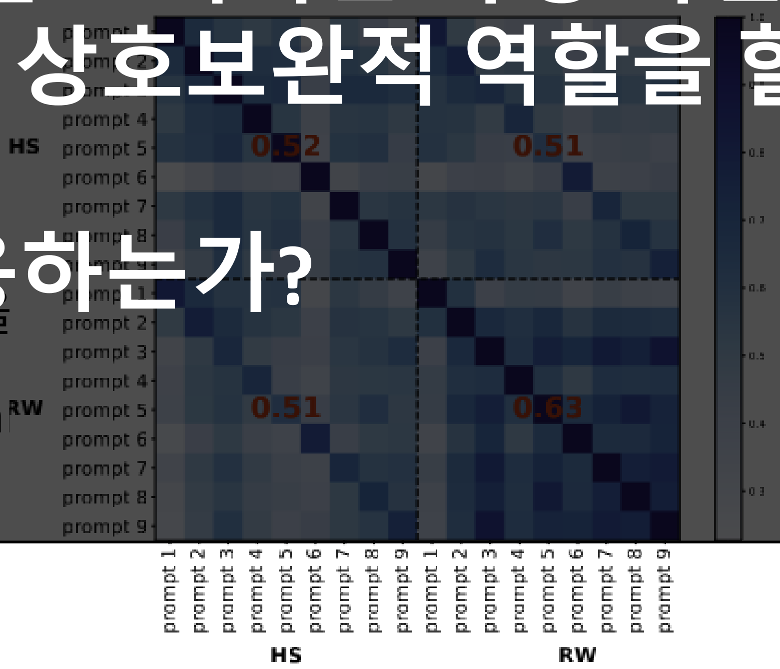
Mixture-of-Experts Embedding (MoEE)

- Comparative & Complementary analysis of Routing Weights & Hidden State

(2) Complementary nature of RW and HS embedding

- STS12 데이터셋에 대해 각 문장쌍마다 HS, RW 임베딩을 생성
- Prompt에 따른 성능 변동성을 고려해 총 9개 prompts에 대해 위 임베딩들을 생성
- [illegible]를 생성
- 두 임베딩으로부터 생성된 Similarity scores 간 correlation [illegible]

→ HS-RW 임베딩 간 similarity score correlation은 0.51로 낮음
→ RW와 HS는 데이터들 중에서도 연관성 없는 측면들을 각각 포착함
→ 서로 다른 trend를 보인다는 것은 높은 complementary nature에 대한 evidence가 될 것



즉, 클러스터링 결과 & 클러스터별 토픽 & 유사도 trend는 RW 와 HS에서 모두 연관성이 거의 없음

- 두 임베딩은 서로 다른 특징들을 포착하는 특성 확인

- 이 둘을 임베딩으로 함께 활용하면 상호보완적 역할을 할 것

→어떻게 함께 활용하는가?

Mixture-of-Experts Embedding (MoEE)

- **최종 임베딩 (2가지 버전)**

1. Concatenation-based combination

- HS와 모든 layers의 dynamic routing weights를 concatenation → HS + comprehensive routing-based embedding $\mathrm{e}_{final} = [\mathrm{e}_{HS}; \mathrm{e}_{RW}] \in \mathbb{R}^{d_{HS}+d_{RW}}$

2. Weighted Sum Integration

- 앞선 STS12 방식과 유사: 문장쌍 (s_1, s_2) 에 대해 임베딩 생성
  $\mathrm{e}_{HS}(s_1), \mathrm{e}_{HS}(s_2), \mathrm{e}_{RW}(s_1), \mathrm{e}_{RW}(s_2)$
- $sim_{HS} = cosine\ similarity\big(e_{HS}(s_1), e_{HS}(s_2)\big),\ sim_{RW} = cosine\ similarity\big(e_{RW}(s_1), e_{RW}(s_2)\big),$
- 최종 similarity: $sim_{final} = sim_{HS} + \alpha \cdot sim_{RW}$
- 알파 – RW의 contribution을 제어하는 hyperparams
- Task-specific needs를 반영하여 다른 tasks에도 적용가능 (future work)

Experiments

- MTEB (Clustering, Classification, Pair classification, Re-ranking, Retrieval, STS, Summarization)
- Model:
  - DeepSeekMoE-16B: 28 layers, 64 experts per layer
  - Qwen-1.5-MoE-A2.7B: 24 layers, 60 experts per layer
  - OLMoE-1B-7B: 16 layers, 64 experts per layer
- 추가학습 x, 바로 임베딩 모델로 활용

Results

- Main results

MTEB Tasks	CLF	Clust.	PairCLF	Rerank	STS	Summ.	Avg.
DeepSeekMoE-16b							
Hidden State (HS)	44.79	25.87	44.34	38.13	34.54	24.51	35.36
Routing Weight (RW)	44.06	17.53	50.59	35.94	41.11	26.22	35.91
MoEE (concat)	44.93	24.15	51.88	41.20	46.82	**31.17**	40.03
MoEE (sum)	**48.74**	**32.83**	**52.12**	**47.88**	**48.34**	29.89	**43.30**
Qwen1.5-MoE-A2.7B							
Hidden State (HS)	46.41	24.31	44.43	44.91	28.36	22.65	35.18
Routing Weight (RW)	38.99	10.55	42.26	33.53	23.97	27.44	29.46
MoEE (concat)	44.81	26.75	49.79	49.23	37.93	**27.61**	39.35
MoEE (sum)	**50.70**	**31.35**	**51.87**	**49.82**	**45.75**	24.00	**42.25**
OLMoE-1B-7B							
Hidden State (HS)	44.23	23.79	47.56	45.60	35.44	20.94	36.26
Routing Weight (RW)	43.54	17.66	**53.12**	40.91	44.68	28.68	38.10
MoEE (concat)	44.62	22.83	51.64	46.58	48.84	**31.67**	41.03
MoEE (sum)	**48.54**	**30.67**	50.93	**47.77**	**49.45**	28.77	**42.69**

MTEB Tasks	CLF	Clust.	PairCLF	Rerank	STS	Summ.	Avg.
Self-Supervised Methods							
Glove⋆ (Reimers, 2019)	51.04	23.11	62.90	48.72	60.52	28.87	45.86
Komninos⋆ (Reimers, 2019)	50.21	**24.96**	66.63	50.03	61.73	30.49	47.34
BERT⋆ (Devlin, 2018)	52.36	23.48	66.10	48.47	52.89	29.82	45.52
SimCSE-BERT-unsup⋆ (Gao et al., 2021)	**54.80**	22.59	**70.79**	**52.42**	**75.00**	**31.15**	**51.13**
Supervised Methods							
SimCSE-BERT-sup⋆	**58.98**	29.49	**75.82**	53.61	**79.97**	23.31	53.53
coCondenser-msmarco⋆ (Gao & Callan, 2021)	53.89	**32.85**	74.56	**60.08**	76.41	**29.50**	**54.55**
SPECTER⋆ (Cohan et al., 2020)	42.59	27.94	56.24	55.87	60.68	27.66	45.16
DeepSeekMoE-16b							
Hidden State (HS)	58.24	24.64	48.76	38.13	59.66	24.38	42.30
Routing Weight (RW)	49.52	19.97	68.30	37.48	59.52	**29.26**	44.01
MoEE (concat)	54.21	26.10	**72.44**	53.31	67.59	28.89	50.42
MoEE (sum)	**58.31**	**34.52**	70.95	**55.99**	**70.66**	29.22	**53.28**
Qwen1.5-MoE-A2.7B							
Hidden State (HS)	59.34	29.50	**74.29**	56.51	67.39	23.01	51.67
Routing Weight (RW)	47.84	16.74	64.85	43.55	51.71	27.74	42.07
MoEE (concat)	54.23	27.18	73.93	56.12	68.52	28.57	51.43
MoEE (sum)	**59.57**	**38.33**	72.21	**56.25**	**72.78**	**31.09**	**55.04**
OLMoE-1B-7B							
Hidden State (HS)	**58.18**	32.83	**72.10**	58.31	72.91	27.96	53.72
Routing Weight (RW)	45.02	19.93	61.58	43.91	54.33	29.49	42.38
MoEE (concat)	52.59	33.92	71.85	56.69	71.13	30.21	52.73
MoEE (sum)	57.46	**36.46**	71.26	**60.43**	**74.63**	**30.71**	**55.16**

- Sum 기반이 가장 좋은 점수: 결합하는 것의 complementary한 효과
- 기존 self-supervised/supervised methods 보다 우수한 성능

Results

- Ablation study

- RW는 마지막 토큰 & all layers에 대해서 수행 (multiple depths) / HS는 마지막 토큰 & last layers
- **의문1: 그렇다면 HS를 여러 layers에서 뽑아서 multiple depths를 보게 하면 되는 거 아닐까?**
- **의문2: last token은 압축된 sequence information을 주지만 every tokens에 대해 mean pooling을 하면 더 넓은 view를 제공하지 않을까?**

Results

- Ablation study

STS Datasets	STS12	STS13	STS14	STS15	STS16	Avg.
DeepSeekMoE-16b						
HS - last token, last layer	51.99	**69.56**	**54.68**	**58.04**	**68.47**	**60.40**
HS - last token, all layers	59.82	60.59	45.20	51.08	58.88	55.03
HS - all tokens, last layer	30.95	34.42	26.77	34.90	37.11	32.78
HS - all tokens, all layers	**60.81**	62.46	46.90	52.38	59.99	56.34
RW - last token	**61.97**	**65.86**	**51.38**	**65.86**	**62.49**	**61.18**
RW - all tokens	50.76	46.42	41.47	43.68	48.37	46.03
MoEE (best)	67.39	81.43	68.98	67.76	74.26	71.75

의문1) 그렇다면 HS를 여러 layers에서 뽑아서 multiple depths를 보게 하면 되는 거 아닐까?

- 실험 결과 HS에 이를 적용하면 성능이 매우 떨어짐 – 오히려 noise로 작용
- RW는 layer마다 선택된 expert의 패턴을 포함하는 임베딩, shallow and deep contextual information을 반영 → HS가 하지 못하는 섬세하고 역동적인 정보를 더 잘 포착할 수 있음

의문2) last token은 압축된 sequence information을 주지만 every tokens에 대해 mean pooling을 하면 더 넓은 view를 제공하지 않을까?

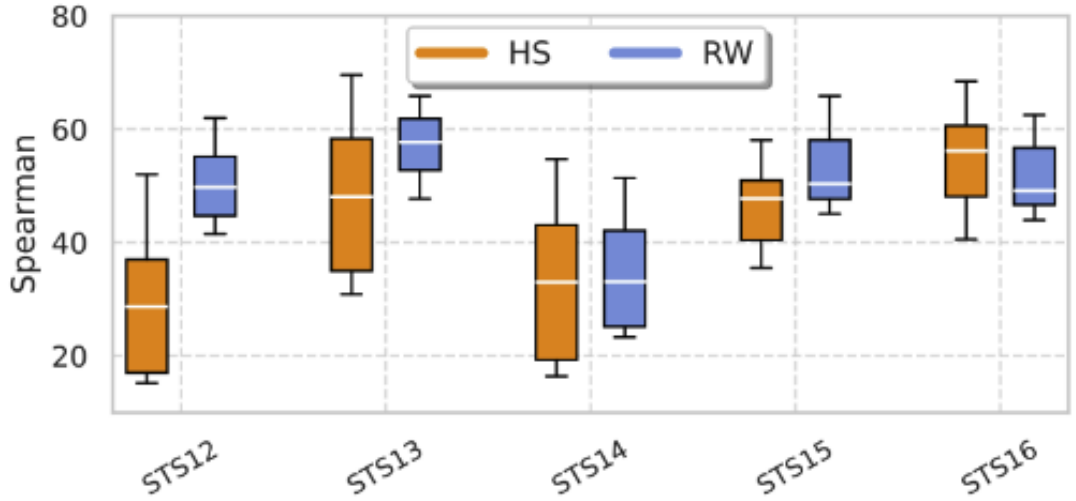
- 실험 결과 last token을 적용하는 것이 HS, RW 모두에서 우수한 성능
  → last token이 가장 critical한 semantic information을 포착할 수 있음

Results

- A stability Comparison of RW and HS using different prompts

- prompt에 따라 임베딩 모델의 성능은 다양하게 나타남
- RW와 HS의 prompt에 따른 sensitivity를 평가하기 위한 실험
- HS가 매우 높은 분산을 보여줬음 → 성능이 사용하는 prompt에 따라 매우 크게 변동
- RS는 높은 stability를 보였음 → robust to prompt choice

→ MoEE는 reliable한 option



Results

- Case study: When HS outperforms RW & vice versa

- HS와 RW가 각각 더 성능이 높았던 instance에 대한 질적 비교

	Sentence 1	Sentence 2
1	the vote will take place today at 5.30 p.m	the vote will take place at 17h30
2	the standards are scarcely comparable, let alone transferable	the norms are hardly comparable and still less transferable
3	that provision could open the door wide to arbitrariness	this point of procedure opens the door to the arbitrary
4	A woman puts flour on a piece of meat	A woman is putting flour onto some meat.
5	the fishermen are inactive, tired and disappointed	fishermen are inactive, tired and disappointment

- HS 임베딩은 **문장구조가 피상적으로만 변경될 때 형식적인 언어적 일관성을 포착하는 데 탁월**
  → 문장의 전반적 구조와 의미를 효과적으로 나타냄, 의미 변화가 적은 케이스에서 효과적

Results

- Case study: When HS outperforms RW & vice versa

- HS와 RW가 각각 더 성능이 높았던 instance에 대한 질적 비교

	Sentence 1	Sentence 2
1	He did, but the initiative did not get very far.	What happened is that the initiative does not go very far.
2	then perhaps we could have avoided a catastrophe	we might have been able to prevent a disaster
3	it increases the power of the big countries at the expense of the small countries	it has the effect of augmenting the potency of the big countries to the detriment of babies
4	festive social event, celebration	an occasion on which people can assemble for social interaction and entertainment.
5	group of people defined by a specific profession	organization of performers and associated personnel (especially theatrical).

- RW embedding은 paraphrasing, synonym, nuanced stylistic shifts의 변화 같은 걸 잘 포착
  → **더 deep한 contextual change에 있어 sensitive** (심층적인 맥락적 변화를 잘 포착)

Conclusion

- MoE는 임베딩 모델로 활용될 수 있음

- HS는 입력에서 최종 예측 결과에 초점을 맞춤
- RW는 Layers마다 어떤 experts로 routing되는지를 보여줌
  → 각 layer 입력에서 MoE의 중간 추론 선택들을 반영할 수 있음
  → **shallow and deep contextual features를 모두 고려한 richer representation**을 얻을 수 있음
  → 한층 더 high-level에서의 의미적 특성을 잘 포착

Future Plans

- MoE는 연구들마다 configuration에 따라 다른 주장을 펼침

- Load balancing loss 적용
- Top-1, 2 vs Soft merging vs Random routing
- Shared expert 활용 vs specialized experts만 활용
- Shared router vs layer-specific router

[Probing]
→MoE의 구조를 뜯어보면서 어떤 기준에서 뚜렷한 routing patterns를 지니는지 평가
→각 configuration들에 대해 비교실험을 통한 결론 도출
→도메인 특화 성능을 높일 수 있는 구조 추가 제안 (routing granularity)

Thank you

Q&A