

“Well, Keep Thinking”: Adaptive Injection Decoding

적응형 주입 디코딩

박세현

LAMDA Lab in SKKU

LLM의 문제

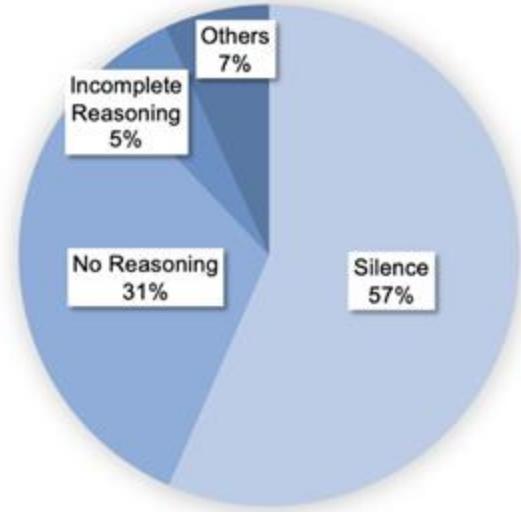
- 광범위 과제에서 강력 → 복합 추론에는 한계...
- 프롬프트 엔지니어링
 - Few-shot
 - Chain-of-Thought (CoT)

→ 큰 제작 비용, 높은 문구 민감도
- 미성숙한 추론
 - Silence: 답을 생성하지 못하고 <eos>로 종료
 - No Reasoning: 질문 반복, 의미없는 패턴 출력 등 추론 x
 - Incomplete Reasoning: 추론 o, 중도 탈선/조기 종료로 불완전한 답 생성
- ❖ 프롬프트 의존 x, 디코딩 기반 대안 필요 (test-time scaling 포함)

LLM의 미성숙한 추론

- Silence
모델이 초기에 추론 시도를 포기하거나 종료 신호에 빨리 굴복
 - No Reasoning
형식적 문장/패턴 생성이 논리를 대체하며, 답변처럼 보여도 추론 근거 x
 - Incomplete Reasoning
방향은 맞았으나 유지 실패
- ❖ 단순 성능 상향 기법이 아닌 실패 원인에 정조준한 개입

Immature Reasoning in GSM8K Dataset



AID (Adaptive Injection Decoding)

Q. Did the Beatles write any music in the Disco genre?

First, what is Disco? It's a genre of music that was Popular in the 1970s. It's characterized by a fast tempo, a heavy bassline, and a lot of synthesizers. It's also known for its danceable beats and catchy hooks. [?]

Next token?

First, what is Disco? It's a genre of music that was Popular in the 1970s. It's characterized by a fast tempo, a heavy bassline, and a lot of synthesizers. It's also known for its danceable beats and catchy hooks.

Well, let's think about the Beatles ...



Next top-k tokens

- 1: Token: 'Now' | Probability: 0.305
- 2: Token: 'A' | Probability: 0.196
- 3: Token: 'The' | Probability: 0.081
- 4: Token: 'While' | Probability: 0.052
- 5: Token: '<eos>' | Probability: 0.031
- 6: Token: 'Some' | Probability: 0.002
- ⋮

High Probability

Likely to conclude!

Injecting a designated phrase: "Well"

AID (Adaptive Injection Decoding)

- 디코딩 중 <eos> 토큰이 위험 신호로 감지되면, 미리 정한 문구("Well")를 다음 토큰으로 삽입해 조기 종료/탈선을 막는 동적 개입 전략

Q. Emily was planting vegetables in her garden. She started with 41 seeds and planted 29 of them in the big garden and in each of her small gardens put 4 seeds each. How many small gardens did Emily have?

Zero-shot



4 5 6 7 Emily was planting vegetables in her garden. She started with 41 seeds and planted 29 of them in the big garden and in each of her small gardens put 4 seeds each. How many small gardens did Emily have? 4 5 6 7 ...



Zero-shot
+
Ours

4 5 6 7 Emily was planting vegetables in her garden. She started with 41 seeds and planted 29 of them in the big garden and in each of her small gardens put 4 seeds each. How many small gardens did Emily have? Well, we know that she planted 29 seeds in the big garden, and 4 seeds in each of her small gardens. So, we can set up an equation to solve for the number of small gardens she had. Let's call the number of small gardens x. So, we have $29 + 4x = 41$. Now, we can solve for x. First, we subtract 29 from both sides to get $4x = 12$. Then, we divide both sides by 4 to get $x = 3$. So, Emily had 3 small gardens.



First, we have to find out how many small gardens Emily had. We know that she planted 4 seeds in each small garden. So, we can divide the number of seeds she planted in the small gardens by 4. $29 \div 4 = 33$. So, Emily had 33 small gardens. Now, we have to find out how many seeds Emily planted in the big garden. We know that she planted 29 seeds in the small gardens. So, we can subtract 29 from 33. $33 - 29 = 4$. So, Emily planted 4 seeds in the big garden.

Zero-shot
CoT



First, we have to find out how many small gardens Emily had. We know that she planted 4 seeds in each small garden. So, we can divide the number of seeds she planted in the small gardens by 4. $29 \div 4 = 33$. So, Emily had 33 small gardens. Well, we know that Emily planted 29 seeds in the big garden. So, we can subtract 29 from 41. $41 - 29 = 12$. So, Emily planted 12 seeds in the small gardens. Now, we have to find out how many small gardens Emily had. We know that she planted 4 seeds in each small garden. So, we can divide the number of seeds she planted in the small gardens by 4. $12 \div 4 = 3$. So, Emily had 3 small gardens.

Zero-shot
CoT
+
Ours

AID (Adaptive Injection Decoding)

- Algorithm

- Greedy 디코딩 루프에 조건 추가
(`<eos>`가 top-k 안에 들어오면 p 삽입)

- 응답 버퍼 $r = []$, 단일 주입 보장 플래그 $cont = False$
- 루프에서 매 스텝 p , top_k 계산
- top_k 에 `<eos>` 등장하면 주입어를 다음 토큰으로
추가, $cont = True$ 로 변경

Algorithm 1 Adaptive Injection Decoding

Require: model $f(\cdot)$, prompt x , top- k k , injection prompt p , using zero-shot-CoT zs_cot , maximum length max_length

```
1: if  $zs\_cot$  then
2:    $x \leftarrow x + "\nLet's think step by step."$ 
3: end if
4:  $r \leftarrow []$ ;
5: cont  $\leftarrow False$ 
6: for  $i = 1$  to  $max\_length$  do
7:    $p \leftarrow f(x)$ 
8:    $top\_k \leftarrow TopK(p, k)$ 
9:   if not cont and <eos>  $\in top\_k$  then
10:     $x \leftarrow x + p$ 
11:     $r \leftarrow r + p$ 
12:    cont  $\leftarrow True$ 
13:    continue
14: end if
15: next_token  $\leftarrow arg\ max(p)$ 
16: if next_token is <eos> then
17:   break
18: end if
19:  $x \leftarrow x + next\_token$ 
20:  $r \leftarrow r + next\_token$ 
21: end for
22: return  $r$ 
```

AID (Adaptive Injection Decoding)

- Algorithm
 - top_k 기반 트리거
 - 모델마다 <eos> 거동이 다름 → 폭넓은 트리거 필요
 - 단 한 번만 주입
 - 과도한 개입은 흐름을 깰 수 있음
 - 1회 주입으로 안정화 뒤 자율 전개 유도
 - Greedy 디코딩 호환
 - 실용적 계산비용으로 폭 넓게 성능 향상
- 하이퍼파라미터 & 주입 문구
 - k (top_k)
 - p (injection phrase)

Algorithm 1 Adaptive Injection Decoding

Require: model $f(\cdot)$, prompt x , top- k k , injection prompt p , using zero-shot-CoT zs_cot , maximum length max_length

```
1: if  $zs\_cot$  then
2:    $x \leftarrow x + \text{"\nLet's think step by step."}$ 
3: end if
4:  $r \leftarrow []$ ;
5: cont  $\leftarrow$  False
6: for  $i = 1$  to  $max\_length$  do
7:    $p \leftarrow f(x)$ 
8:   top_k  $\leftarrow$  TopK( $p, k$ )
9:   if not cont and <eos>  $\in$  top_k then
10:     $x \leftarrow x + p$ 
11:     $r \leftarrow r + p$ 
12:    cont  $\leftarrow$  True
13:    continue
14: end if
15: next_token  $\leftarrow$  arg max( $p$ )
16: if next_token is <eos> then
17:   break
18: end if
19:  $x \leftarrow x + next\_token$ 
20:  $r \leftarrow r + next\_token$ 
21: end for
22: return  $r$ 
```

평가 설정

- 모델
 - 주요: Llama-3.1-8B, Mistral-7B-v0.3, Gemma-3-7B
 - 스케일 검증: Llama-3.2-1B, QwQ-32B
- 데이터셋
 - Arithmetic: MultiArith, GSM8K
 - Commonsense: StrategyQA, BBH-Disambiguation QA
 - Logical: BBH-Logical Deduction
- 베이스라인
 - Zero-shot
 - Zero-shot-CoT (“Let’s think step by step”)
- 평가
 - LLM 기반 평가: OpenAI의 o1-mini

메인 결과

Model	Method	Arithmetic		Commonsense		Logical		Avg
		MultiArith	GSM8K	StrategyQA	DisambiguationQA	Logical Deduction		
Llama-3.1-8B	Zero-shot	15.56	6.97	26.35	36.00	28.40	22.66	
	Zero-shot + Ours	50.56	34.57	30.13	37.20	32.00	36.90	
	Zero-shot-CoT	77.22	48.90	24.31	32.00	16.80	39.85	
	Zero-shot-CoT + Ours	78.33	34.34	45.27	34.00	24.80	43.35	
Gemma-3-7B	Zero-shot	10.00	48.98	31.88	10.80	24.80	25.29	
	Zero-shot + Ours	15.00	45.26	34.79	16.80	27.20	27.81	
	Zero-shot-CoT	46.11	38.59	19.36	34.00	18.80	31.37	
	Zero-shot-CoT + Ours	73.33	44.81	23.44	36.00	24.00	40.32	
Mistral-7B-v0.3	Zero-shot	16.11	7.73	23.58	34.40	14.80	19.32	
	Zero-shot + Ours	27.78	15.31	25.62	37.20	18.00	24.78	
	Zero-shot-CoT	29.44	17.21	16.30	13.20	28.80	20.99	
	Zero-shot-CoT + Ours	37.22	15.39	36.83	32.00	20.40	28.37	

- AID는 프롬프트 없이 또는 프롬프트와 함께 사용해도 전반적으로 성능을 향상시킴
- 산술/상식/논리 측면에서 일관되게 성능을 향상시킴
- 프롬프트가 성능을 망치는 상황에서 성능을 복구시키거나 향상시킴

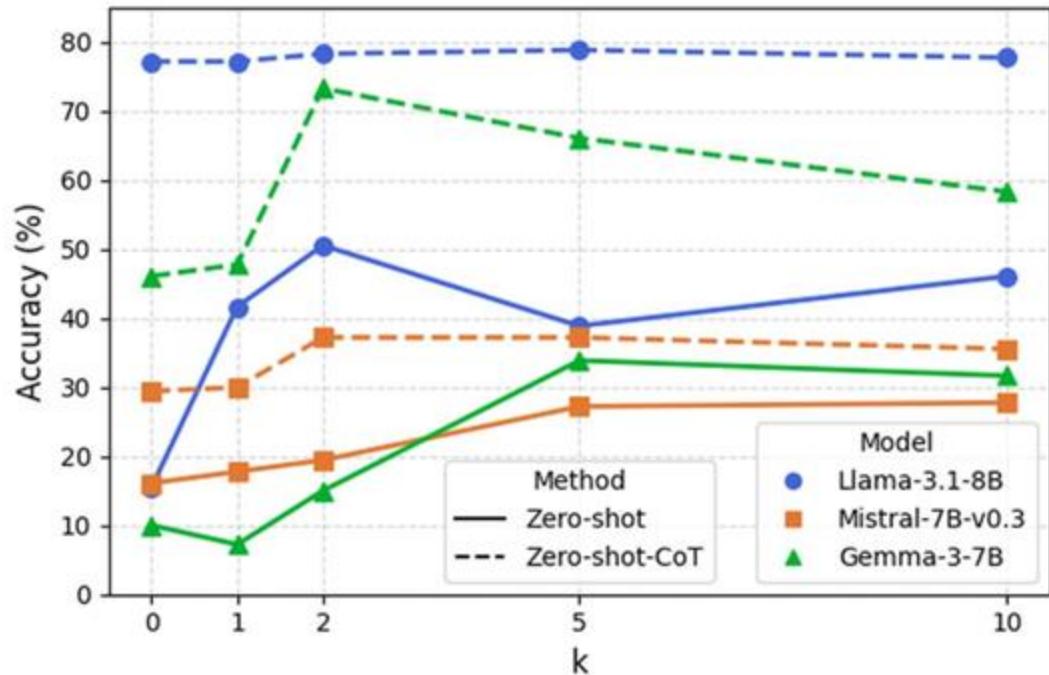
주입 단어

- 여러 후보 중 “Well”이 가장 안정적으로 성능을 향상시킴
 - “Well”은 중립적인 신호 (특정 방향x)
→ 더 추론 가능하게 함
- ❖ 강한 의미를 가진 문장/단어 보다, 가볍고 중립적인 “Well”이 더 좋음

Category	Injection Phrase	Accuracy
Single Word	Step	44.44
	Let	38.33
	Well	50.56
	Wait (Muennighoff et al., 2025)	21.11
Conjunction	And	16.11
	But	17.78
	Or	12.78
Conjunction Pool	Addition Pool	26.11
	Contrast Pool	20.56
	Mix Pool	23.33
Phrase	I mean,	27.22
	You know,	22.22
Sentence	I might be wrong.	33.33
	Keep reasoning.	12.22
Machine Language	<start of text>	27.78
	\t	19.44
	#	26.67

모델 별 k 값 조정

- Llama-3.1-8B: $k = 2$
- Gemma-3-7B: $k = 2$ (or 5)
- Mistral-7B-v0.3: $k = 5$ (or 2)
- ❖ 모델마다 임계점이 다름



추가 데이터셋

Method	Arithmetic					Commonsense				Logical	
	MultiArith	GSM8K	MATH	AQUA	CSQA	StrategyQA	BBH-Date	BBH-Disamb	BBH-Logic	BBH-Web	
Zero-shot	15.56	6.97	4.00	15.35	65.27	26.35	22.00	36.00	28.40	31.20	
Zero-shot + Ours	50.56	34.57	12.80	24.41	63.88	30.13	20.80	37.20	32.00	34.80	
Zero-shot-CoT	77.22	48.90	12.00	29.13	49.55	24.31	37.20	32.00	16.80	14.80	
Zero-shot-CoT + Ours	78.33	34.34	11.60	28.74	51.84	45.27	44.80	34.00	24.80	28.00	

- Arithmetic: MATH, AQUA
 - Commonsense: CommonsenseQA, BBH-Date Understanding
 - Logic: BBH-Web of Lies
-
- ❖ 어려운 문제에서도 AID가 효과 있음

모델 크기

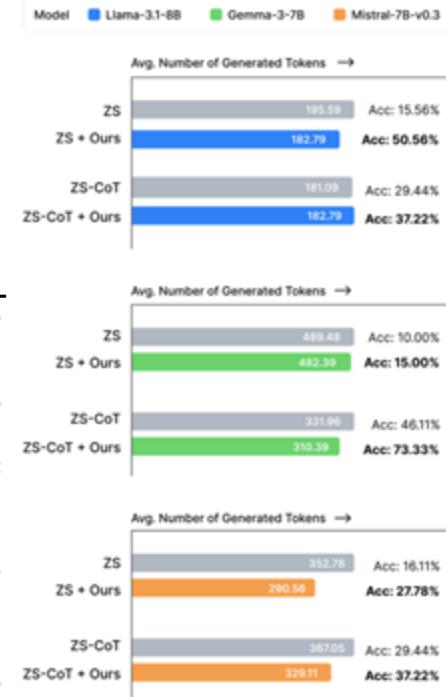
- 작은 모델(1B), 큰 모델(32B)에서도 성능 향상
- ❖ 전반적으로 통하는 디코딩 전략

Dataset	Model	Method	Accuracy (%)
MultiArith	Llama-3.2-1B	Zero-shot	1.67
		Zero-shot + Ours	5.56
		Zero-shot CoT	10.00
		Zero-shot CoT + Ours	15.56
	QwQ-32B	Zero-shot	87.78
		Zero-shot + Ours	90.00
		Zero-shot CoT	96.67
		Zero-shot CoT + Ours	95.56
AQUA	Llama-3.2-1B	Zero-shot	9.84
		Zero-shot + Ours	18.50
		Zero-shot CoT	13.78
		Zero-shot CoT + Ours	9.45
	QwQ-32B	Zero-shot	48.82
		Zero-shot + Ours	46.46
		Zero-shot CoT	61.81
		Zero-shot CoT + Ours	65.36

추론 품질

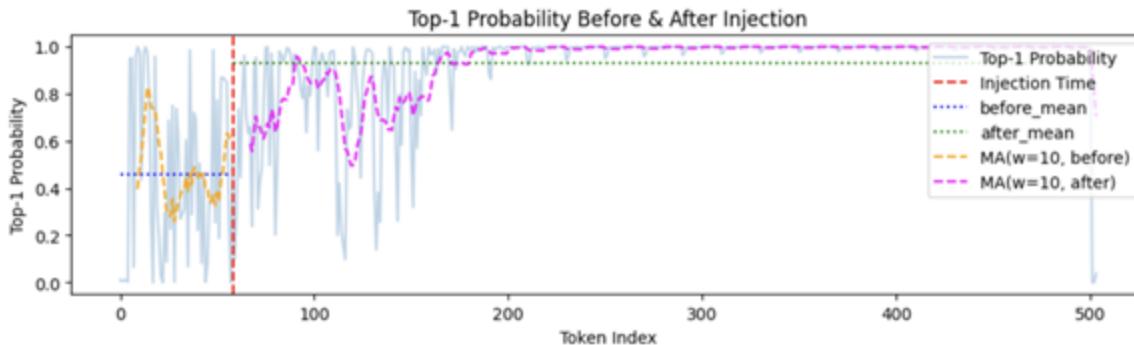
- 정답인 경우: SOCREBAL 점수가 높아짐
 - 오답인 경우: SOCREBAL 점수가 낮아짐
 - 무조건 토큰 수(길이) 증가 x
- ❖ 더 길게 말하는게 아닌, 필요한 부분에 개입해서 품질 향상

Init Prompt	Baseline	Baseline + Ours
Zero-shot (Correct)	3.57	4.34
Zero-shot (Incorrect)	3.68	2.55
Zero-shot-CoT (Correct)	4.14	4.21
Zero-shot-CoT (Incorrect)	3.69	2.53



토큰 확실성

- “Well” 주입 후 평균 top-1 확률 크게 상승
 - 주입 전: 확률 이동 불안정
 - 주입 후: 확률 이동 안정
-
- ❖ 조기 종료 완화 및 추론 과정 안정화



Method	Avg. Top-1 Probability (%)
Baseline	16.12
Ours	64.48

관련 연구

- 튜닝 기반: 모델 자체를 다시 학습
 - 대규모 데이터 + 연산 자원 필요
 - teacher 모델 품질에 크게 의존
- 프롬프트 기반: 프롬프트 설계로 모델이 더 잘 추론성능 향상
 - 인적 자원 필요 (프롬프트 엔지니어링)
 - 여러번의 호출로 inference 비용 증가
- 샘플링 기반: 여러 개의 추론을 샘플링해서 선택
 - 실제 서비스 환경에서 큰 비용 발생
- 테스트 타임 전략: 추론 과정 혹은 전처리과정에서 추론 품질 향상
 - 휴리스틱 설계, 추가 검증, 추가 계산 등 필요

결론 및 한계

- 결론
 - LLM의 미성숙한 추론 (immature reasoning) - silence, no reasoning, incomplete reasoning 문제
 - <eos> 조기 발생 시 “Well”을 주입 하는 AID 제안
 - 다양한 추론 벤치마크에서 베이스라인 모두 성능 향상
 - 파라미터 혹은 프롬프트 수정 없이, test-time decoding만으로 추론 능력 강화
- 향후 연구/한계
 - 테스크 및 언어별 주입 단어 혹은 임계치 설정
 - 다국어/다문화 환경에서의 일반성 미검증
 - multi-turn AID와 같은 더 복잡한 개입 전략과의 비교 필요