Banya AI 5

Cache-Dictionary Tokenization and Dynamic Embedding: Near-Free Vocabulary Expansion via a Low-Rank Head

캐시 사전 토큰화와 동적 임베딩: 로우랭크 헤드로 어휘 확장을 거의 무비용으로

한혁진 (bokkamsun@gmail.com) · ORCID 0009-0007-4132-1707

반야 연구 시리즈 제5편

초록

토큰화(tokenization, 글을 모델이 다루는 낱말 단위로 쪼개는 일)를 두 층으로 둔다. 아래층은 음절 원자(syllable atom, 한글 소리 단위)이고, 위층은 자주 쓰는 낱말을 통째로 담는 묶음 사전(bundle dictionary)이다. 측정한 모델의 어휘는 음절 2724개 위에 묶음 8000개를 얹어 총 10724개이다. 보통 어휘를 4배로 증가시키면 출력 헤드(output head, 다음 낱말을 고르는 마지막 층)의 파라미터가 그만큼 증가해 비싸지지만, 본고는 출력 헤드를 로우랭크(low-rank, 큰 행렬을 두 작은 행렬의 곱으로 근사)로 두어 어휘 확장을 거의 무비용으로 만든다. 어휘 10724, 채널 폭 1024, 랭크 128에서 로우랭크 헤드는 약 150만 개의 파라미터로, 밀집 헤드의 약 1098만 개 대비 약 7.3배 적다. 나아가 묶음층 토큰도 데이터면과 연산면의 직교 구조를 그대로 유지한다(제3편의 연산자 창발이 묶음에서도 방향 일관성 8.1배로 나타난다). 즉 캐시 사전은 압축을 얻으면서도 앞 편들이 확립한 데이터-연산 구조를 깨지 않는다. 절감과 품질의 동행도 측정했다. 정상 경로인 묶음 켬은 홀드아웃 텍스트를 글자당 5.08비트로 예측하면서 같은 글을 21.5% 적은 토큰으로 담고, 묶음을 끄면 낱말 내부 자리에서 예측이 무너져(토큰당 11.49비트 대 경계 6.69비트) 이 시스템의 낱말 지식이 묶음 경로에 담겨 있음이 드러난다.

측정

이 글의 정량 수치는 집필 환경에서 측정한 실측값이다(초등 단계 얼린 모델, 환경은 부록 B). 로우랭크 헤드의 파라미터 수처럼 구조에서 곧바로 계산되는 값은 사실로 표기한다.

목차

용어 범례

용어를 아래 표에 정의한다. 같은 개념에는 같은 이름만 사용하며, 일곱 편의 본문과 그림이 모두 이 표를 따른다. 코드 기준은 동봉 코드에서 그 개념을 구현한 식별자이고, 일반 용어는 학계에서 통용되는 가장 가까운 이름이다. 다를 때만 칸을 채웠다.

용어 범례 1. 엔진과 신용

용어	뜻	코드 기준	일반 용어
정확한 전치	각 연산의 수반을 손으로 유도해 닫힌형으로 구현한 역방향 계산. 이 엔진과 신용 사슬의 고유명	`banya_core.py`	수동 역전파
수반	순전파의 짝 연산. 기울기를 정확히 거꾸로 되돌려 흘린다	`bwd` 커널들	adjoint
신용	어떤 지점의 은닉에 대한 손실 기울기. 파라미터가 결과에 진 책임	델타(δ)	그레이디언트
신용 사슬	캐시를 역순으로 되짚으며 수반을 곱해 신용을 입력까지 되돌리는 골격	`backward`	오차역전파
수반 정합	수반 식이 유한차분과 같은 값을 내는지 확인하는 검사	각 편 프로브	gradient check
순환행렬 채널 혼합	채널 혼합을 순환행렬로 두고 rfft 성분곱으로 계산하는 연산		순환 합성곱
상대 거리 혼합	거리에만 의존하는 공유 계수로 자리들을 섞는 혼합	`m_mat_w_lag`	Toeplitz 혼합
시간혼합	다른 자리의 정보가 현재 자리로 섞여 드는 경로의 총칭		토큰 혼합
혼합 헤드	혼합을 병렬로 나누는 머리 수		어텐션 헤드
필터 사다리	넓은 필터 몇 개부터 좁은 필터 여러 개까지 스케일이 깊이를 따라 변하는 콘볼루션 채널 변환	`m_mat_w_filter`	콘볼루션 필터뱅크
창	모델이 한 번에 보는 토큰 자리들. 측정 단위로는 고정 길이 텍스트 조각	`T`	컨텍스트 윈도
프로브	논문 수치를 목표값과 함께 재생하는 검증 스크립트	`probe` 폴더	재현 스크립트
얼린 모델	학습을 멈춰 고정한 발달 단계 모델	`model` 폴더 npz	frozen checkpoint
홀드아웃	학습에 쓰지 않은 평가 텍스트		held-out set

용어 범례 2. 바이토큰과 게이트

용어	뜻	코드 기준	일반 용어
바이토큰	토큰 하나를 데이터면과 연산면의 직교 쌍으로 두는 구조	`USE_BITOKEN`	
데이터면	토큰의 내용을 담는 임베딩 면. 더하기 자리로 들어간다	`m_mat_w_data_axis`	토큰 임베딩
연산면	토큰의 연산 지시를 담는 임베딩 면. 곱하기 자리로만 들어간다	`m_mat_w_operate_axis`	
더하기 자리, 곱하기 자리	데이터면이 실리는 잔차 덧셈 위치와 연산면이 작용하는 게이트 곱셈 위치		잔차 연결, 게이팅
게이트	각 채널의 통과량을 0에서 1로 정하는 밸브. 직전 토큰의 연산면이 주입된다	`m_mat_w_gate`	게이팅
한 칸 이동	직전 토큰의 연산면이 다음 자리의 게이트로 주입되는 한 자리 밀림		
연산이 곧 가중치	연산면 임베딩이 라벨 없이 가중치 역할로 창발하는 현상		
일관 연산자	어떤 데이터에도 일관된 방향으로 작용하는 연산자		
방향 일관성	같은 연산이 여러 데이터를 꺾는 방향들의 평균 코사인. 연산자 창발의 지표	`p_direction_consistency`	
회전 연산자 게이트	연산면을 오일러 회전각으로 읽는 게이트. 본문 축약은 회전 게이트	`paper7_rotation.py`	
기본 연산	게이트가 한 홉 지시로 곧바로 표현할 수 있는 연산		프리미티브
깊이 배제 판별	깊이를 1블록으로 제한해 게이트만 변수로 남긴 의도된 편파 측정	제7편 프로브	
자기 되먹임	자기 출력을 입력으로 되먹이며 끝 상태를 채점하는 채점 방식		자기회귀 롤아웃

용어 범례 3. 되새김과 발달

용어	뜻	코드 기준	일반 용어
되새김	기울기 없이 임베딩을 이웃끼리 당겨 뭉치게 하는 자기조직화 학습	제2편 프로브	자기조직화
동종군	씨앗에서 코사인 최근접 이웃을 모은 뜻 가까운 개념 집합		클러스터
무게중심 당김, 감쇠	평균 쪽으로 당기고 원본 쪽으로 되돌리는 되새김의 두 힘. 균형에서 평형이 창발한다	`ALPHA`, 감쇠율	
월드먼저	언어 이전에 감각, 공간, 논리 축을 먼저 발달시키는 커리큘럼 순서	`banya_world_data`	커리큘럼 학습
발달 단계	스텝 수로 정의하고 사람 성장 이름을 붙인 체크포인트 단계		
축	임베딩 안에 방향으로 형성되는 성질의 눈금. 감각, 순서, 범주		표현 축

용어 범례 4. 어휘와 메타인지

용어	뜻	코드 기준	일반 용어
음절 원자	어휘 바닥층의 음절 단위 토큰	`AtomTokenizer`	문자 단위 토큰
묶음	음절 원자를 묶어 사전에 올린 개념 토큰. 어휘 단위로만 사용한다		서브워드
캐시 사전 토큰화	원자 아래층에 묶음 사전 위층을 얹은 2층 어휘 체계	제5편 프로브	서브워드 토큰화와 대비
확실, 모호, 모름	자기 출력 분포로 갈리는 세 앎 상태	pmax, 엔트로피	불확실성 추정
라우팅	상태에 따라 답의 처리 경로를 배정하는 것		routing
재해석	모호 히든을 문맥과 함께 재순전파해 확실로 끌어올리는 절차	제6편 프로브	
유사도 정리	데이터면 유사도로 개념이 정돈되는 축		
추론 순회	연산면을 따라 추론을 이어 가는 축		
자기 발화	유사도 정리와 추론 순회를 엮어 스스로 말을 잇는 목표 상태		
반야프레임	시리즈의 배경을 이루는 공리 15개 체계. 본문 주장의 근거는 아님		

기호 범례

이 편의 수식과 본문이 함께 사용하는 기호를 정의한다. 한 식 안에서만 쓰는 국소 기호는 각 식 아래의 기호 줄에 적었다.

기호	한글 명칭	정의 및 의미	자료형
V	어휘 크기	모델 어휘의 토큰 개수. 측정한 모델은 음절 원자 2724개에 묶음 8000개를 얹어 총 10724	정수
H	채널 폭	모델 은닉 채널의 폭. 측정한 모델의 설정값 1024	정수
R	랭크	로우랭크 분해의 랭크. 측정한 모델의 설정값 128	정수
W_{head}	출력 헤드 행렬	다음 토큰을 고르는 마지막 층의 큰 행렬. 두 작은 행렬의 곱 $W_a W_b$로 근사한다	행렬
W_a	로우랭크 앞 인자 행렬	로우랭크 분해의 앞쪽 행렬. 크기 $V \times R$	행렬
W_b	로우랭크 뒤 인자 행렬	로우랭크 분해의 뒤쪽 행렬. 크기 $R \times H$	행렬
$V \times H$	밀집 헤드 파라미터 수	밀집 출력 헤드의 파라미터 개수. 어휘 크기와 채널 폭의 곱으로 약 1098만 개	정수
$(V+H) \times R$	로우랭크 헤드 파라미터 수	로우랭크 출력 헤드의 파라미터 개수. 어휘와 채널의 합에 랭크를 곱해 약 150만 개	정수
$\times$	곱셈 기호	차원끼리 곱해 파라미터 개수를 계산할 때 사용하는 곱셈 표기	연산자
$\in$	원소 기호	행렬이 어떤 실수 공간에 속함을 나타내는 소속 관계 표기	연산자
$\mathbb{R}$	실수 집합	행렬 성분이 실수임을 나타내는 집합 기호. $\mathbb{R}^{V \times H}$ 꼴로 행렬이 속하는 공간을 표기	집합 표기
$\ll$	훨씬 작음 기호	랭크가 채널 폭보다 훨씬 작다는 조건 $R \ll H$를 표기	연산자
$\approx$	근사 기호	절감 배율 계산 결과가 약 7.3임을 나타내는 근사 표기	연산자
$\pm$	플러스마이너스 기호	3.3절 표의 조각별 차이를 평균과 표준편차로 표기	연산자

1. 서론

어휘를 키우면 표현력이 증가하지만 출력 헤드가 비싸진다. 밀집 헤드는 어휘 크기 V와 채널 폭 H의 곱 $V \times H$만큼 파라미터를 가지므로, 어휘를 키우면 그만큼 커진다. 본고는 토큰화를 음절 원자 위에 묶음 사전을 얹은 2층 캐시 사전으로 두어 어휘를 키우되, 출력 헤드를 로우랭크로 분해해 그 비용을 거의 없앤다. 나아가 묶음층 토큰이 앞 편들이 확립한 데이터면과 연산면의 직교 구조를 그대로 유지하는지를 측정으로 확인한다. 학습에는 제1편 [2]의 엔진을 쓴다. 이론적 배경은 별도 공리 문헌 [1]에 있으나, 본고는 그와 독립적으로 측정에서만 결론을 낸다.

2. 방법

아래층 음절 토큰화는 글을 소리 단위로 쪼갠다. 위층 묶음 사전은 자주 쓰는 낱말을 최장 일치(greedy longest match, 가장 긴 것부터 맞춰 통째로 잡기)로 한 토큰으로 묶는다. 입력과 출력이 같은 사전을 쓴다. 출력 헤드는 큰 행렬 $W_{\text{head}} \in \mathbb{R}^{V \times H}$를 두 작은 행렬의 곱 $W_a W_b$로 근사한다($W_a \in \mathbb{R}^{V \times R}$, $W_b \in \mathbb{R}^{R \times H}$, 랭크 $R \ll H$).

식 2-1. 로우랭크 헤드의 파라미터 비용

$$\text{밀집} : V \times H, \qquad \text{로우랭크} : (V + H) \times R$$

$$\frac{V \times H}{(V + H) \times R} = \frac{10724 \times 1024}{(10724 + 1024) \times 128} \approx 7.3$$

수식

기호: V는 어휘 크기(10724), H는 채널 폭(1024), R은 랭크(128)이다.
작동 방식: 밀집 헤드는 어휘와 채널의 곱만큼 크지만, 로우랭크 헤드는 어휘와 채널을 각각 랭크로만 곱한 합이다. 어휘가 커져도 랭크가 고정이면 비용은 어휘에 선형으로만 증가한다.
실제 동작: 어휘 10724에서 밀집 헤드는 약 1098만 개, 로우랭크 헤드는 약 150만 개로 약 7.3배 적다. 어휘를 더 키워도 로우랭크 헤드는 $V \times R$만 증가해 훨씬 완만하게 커진다.

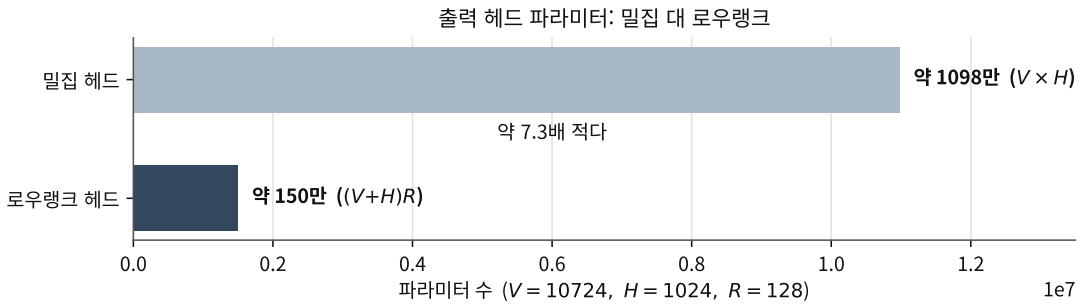
3. 결과

3.1 2층 캐시 사전과 로우랭크 헤드의 비용

측정

측정한 모델의 어휘는 음절 2724개에 묶음 8000개를 얹어 총 10724개이다. 이 어휘에서 로우랭크 헤드(랭크 128)의 파라미터는 약 150만 개로, 같은 어휘의 밀집 헤드 약 1098만 개 대비 약 7.3배 적다(구조적 계산값).

그림 3-1. 어휘 10724에서 밀집 헤드 대 로우랭크 헤드 (구조적 계산)



같은 어휘 10724에서 로우랭크 헤드는 밀집 헤드 대비 약 7.3배 적은 파라미터로 출력을 낸다. 어휘를 키워도 로우랭크 헤드는 완만하게만 커지므로, 2층 캐시 사전의 어휘 확장이 거의 무비용이 된다.

3.2 묶음층도 데이터-연산 구조를 유지한다

어휘를 묶음으로 키워도 앞 편들이 확립한 데이터면과 연산면의 직교 구조가 깨지지 않는지 측정했다.

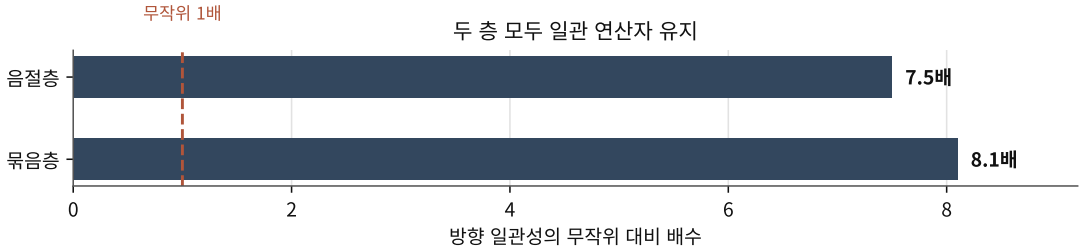
측정

측정한 결과, 묶음층 토큰의 연산면도 일관 연산자로 작용한다.

- 방향 일관성 : 묶음 토큰의 방향 일관성은 0.252로 무작위 0.031의 8.1배 이다(음절 토큰은 7.5배). 제3편 [3]에서 본 연산자 창발이 묶음층에서도 유지된다.
- 연산 직교성 : 묶음 토큰의 서로 다른 연산 쌍의 77퍼센트 가 거의 직교(음절 79퍼센트)이고, 같은 방향 쌍은 0퍼센트이다. 묶음마다 고유한 연산이 보존된다.

즉 어휘를 묶음으로 키워도 데이터-연산 직교 구조가 깨지지 않는다.

그림 3-2. 음절층과 묶음층의 방향 일관성



음절층(7.5배)과 묶음층(8.1배) 모두 방향 일관성이 무작위보다 훨씬 높다. 어휘를 묶음으로 키워도 데이터-연산 직교 구조가 유지된다.

재현

3.1절과 3.2절의 수치는 부록 C의 준비를 마친 뒤 다음 한 줄로 재생된다.

```
cd probe && python3 paper5_cache_token_probe.py
```

출력에서 어휘 10724(음절 2724 + 묶음 8000), 로우랭크 헤드 7.3배 절감, 묶음층 방향 일관성 8.1배, 직교쌍 79%와 77%가 본문 수치와 대응한다.

3.3 절감이 품질과 같이 가는가, 그리고 묶음 끔 진단

묶음 사전의 실익은 같은 글을 더 적은 토큰으로 담는 것이다. 그 절감이 예측 품질과 같이 가는지를 확인하기 위해, 학습에 쓰지 않은 홀드아웃 텍스트 두 벌(각 30조각, 총 6,571글자)을 같은 얼린 모델에 두 인코딩으로 통과시켰다. 묶음 켬은 캐시 사전의 최장 일치 인코딩이고, 묶음 끔은 음절 원자만으로 푼 인코딩이다. 토큰 단위가 서로 달라 토큰당 교차엔트로피는 비교가 되지 않으므로, 예측된 토큰들이 덮는 글자 수로 나눈 글자당 비트로 정규화했다. 모든 조각이 두 인코딩 모두 문맥창 하나에 들어가므로 문맥 절단은 양쪽 다 없었다.

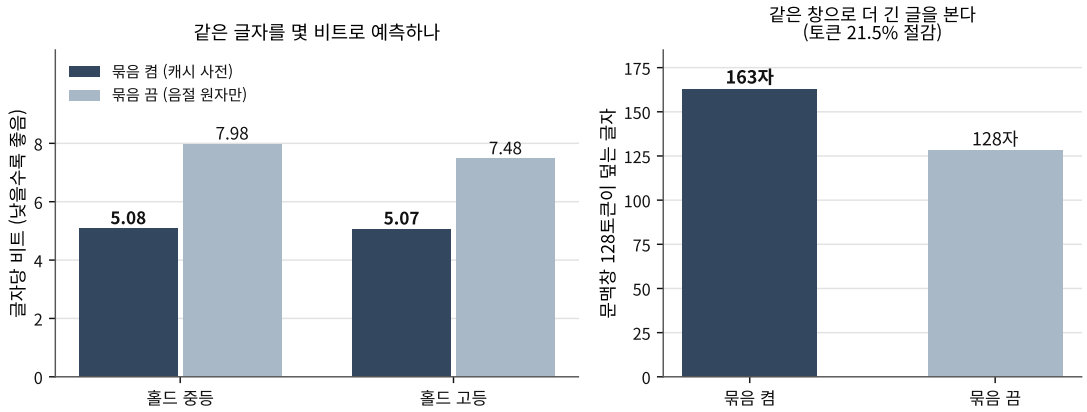
측정

정상 경로인 묶음 켬은 글자당 5.08비트(홀드 중등)와 5.07비트(홀드 고등)로 동작하면서 같은 글을 21.5% 적은 토큰으로 담는다. 창 128토큰이 덮는 글자 수로 환산하면 켬 약 163자 대 끔 128자인데, 이는 토큰 대 글자 비율의 산술 환산이며 이번 측정에서 문맥창 차이가 비트 수치에 작용한 적은 없다.

평가셋	묶음 켬	묶음 끔	끔(원자 어휘 재정규화)	조각별 차이
홀드 중등 (30조각)	5.08	7.98	7.58	+2.90±0.95, 양수 30/30
홀드 고등 (30조각)	5.07	7.48	7.07	+2.41±0.82, 양수 30/30

묶음 끔 수치는 품질 비교가 아니라 분포 밖 진단으로 읽어야 한다. 이 모델은 묶음 켬 스트림으로 학습됐고, 최장 일치가 결정적이라 자주 쓰는 낱말을 원자로 풀어 쓴 형태는 학습에 구성상 존재하지 않는다. 실제로 끔의 붕괴는 낱말 내부 자리에 몰린다. 낱말 내부 자리는 토큰당 11.49비트인데 경계 자리는 6.69비트다. 묶음 끔(원자 스트림)에서 모델은 묶음 토큰들에 평균 17.6%의 확률 질량을 주고 12.9%의 자리에서는 최대 확률 토큰이 묶음이다. 즉 모델이 원자 자리에서 묶음을 내려 하고 있으며, 이 낱말들의 지식이 묶음 경로에 담겨 있다는 뜻이다.

그림 3-3. 묶음 켬과 끔의 글자당 비트, 그리고 창이 덮는 글자



왼쪽은 같은 홀드아웃 텍스트의 글자당 비트로, 정상 경로인 묶음 켬이 5.08과 5.07비트로 동작한다. 묶음 끔은 분포 밖 진단이며 원자 어휘로 재정규화하면 7.58과 7.07로 내려간다. 오른쪽의 창당 글자 수는 토큰 대 글자 비율의 산술 환산이다.

이 측정의 해석 한계를 명시한다. 묶음 끔으로 학습한 대조 모델이 없으므로 묶음이 품질의 원인이라는 인과 주장은 하지 않는다. 원자 전용으로 학습한 모델은 끔 수치보다 훨씬 좋을 수 있다. 이 측정이 보이는 것은 정상 경로가 절감과 함께 동작한다는 것과, 이 시스템의 낱말 지식이 묶음 경로에 일방향으로 담겨 있다는 것까지다. 글자당 비트는 최장 일치 단일 분할의 부호 길이라 분할을 합산한 참확률의 상계이며, 절감률과 비트 수치는 이 홀드아웃 두 벌 기준이다.

재현

이 소절의 품질 대조는 부록 C의 준비를 마친 뒤 다음 한 줄로 재생된다.

```
cd probe && python3 paper5_quality_contrast_probe.py
```

출력에서 묶음 켬 5.08 과 5.07 비트, 끔 진단 7.98 과 7.48(원자 어휘 재정규화 7.58 과 7.07), 낱말 내부 11.49 대 경계 6.69 비트, 토큰 절감 21.5%가 본문 수치와 대응한다.

4. 한계와 다음 측정 과제

기준선은 1인 개발, 자작 정확한 전치 역전파 엔진, 소비자 그래픽 처리장치 한 대라는 레퍼런스 클래스이다. 아래는 결함이 아니라 다음 측정 과제이다.

- 표준 토큰화 대비 : 같은 어휘 크기의 표준 부분어(subword) 토큰화와 나란히 놓아, 캐시 사전이 그 이상인지 확인하는 것이 다음 과제이다.
- 동적 성장 : 학습 중 묶음 사전을 키우는 동적 성장은 설계되어 있으며, 그 실측은 다음 과제이다. 어휘가 자라면 임베딩과 헤드가 따라 늘어나는 동적 임베딩은 이 동적 성장 설계가 담당한다.
- 압축과 하류 과제 : 3.3절이 글자당 비트 기준으로 절감과 품질의 동행을 측정했으나, 질답 같은 하류 과제와 원자 전용으로 학습한 대조 모델과의 나란한 비교는 다음 과제이다.

5. 결론

본고는 음절 원자 위에 묶음 사전을 얹은 캐시 사전 토큰화와, 어휘 확장을 거의 무비용으로 만드는 로우랭크 출력 헤드를 제시했다. 어휘 10724에서 로우랭크 헤드는 밀집 헤드 대비 약 7.3배 적은 파라미터를 쓰고, 묶음층 토큰도 데이터면과 연산면의 직교 구조를 방향 일관성 8.1배로 유지한다. 어휘를 키우면서도 앞 편들이 확립한 구조를 깨지 않는다. 절감은 품질과 같이 간다. 정상 경로는 홀드아웃을 글자당 5.07에서 5.08비트로 예측하며 토큰 21.5%를 절감하고, 묶음 끔은 낱말 내부에서 무너져 낱말 지식이 묶음 경로에 담겨 있음을 보인다. 원자 전용 학습 대조 모델과의 인과 비교는 다음 과제로 남긴다. 어휘가 자라도 직교가 지켜져야 하는 이유는 본편의 효율 계산 바깥에 있다. 그 이유는 시리즈의 끝이 밝힌다.

6. 후속 논문 예고

후속 논문 예고

- 제6편 — 자기 분포와 기하 모니터링에서 창발하는 제어 신호 : 모델이 자기 출력 분포와 임베딩 기하를 읽어 확실, 모호, 모름을 라우팅하는 신호를 측정된 통계량으로 다룬다.
- 마지막 편(미완) — 유사도 정리와 추론 순회의 결합 : 데이터면 유사도 정리(제2편)와 연산면 추론 순회를 엮는 자기 발화. 최종 조립이 끝나지 않아 아직 다루지 않는다.

참고문헌

[1] 한혁진 (2026). Banya Framework: An Axiom-Based Science Mining Engine for Deriving Fundamental Physical Constants (반야프레임 공리 15개). Zenodo. https://doi.org/10.5281/zenodo.20301304 . (본 시리즈의 배경 자연관, 선행 공리 문헌)

[2] 한혁진 (2026). 자동미분 없이 수동으로 유도한 정확한 전치로 학습하는 언어모델 엔진. 반야 연구 시리즈 제1편. Zenodo. https://doi.org/10.5281/zenodo.21385447

[3] 한혁진 (2026). 바이토큰: 직교 데이터-연산 토큰과 라벨 없이 창발하는 연산자. 반야 연구 시리즈 제3편. Zenodo. https://doi.org/10.5281/zenodo.21385676

부록 A. 어휘 구성

표 A-1. 측정한 모델의 2층 캐시 사전

층	개수	내용
음절 원자	2724	한글 소리 단위
묶음 사전	8000	자주 쓰는 낱말 통째
합계 어휘	10724	입출력 공용

부록 B. 측정 환경

표 B-1. 측정에 사용한 컴퓨터 사양

구분	사양
그래픽 처리장치(GPU)	NVIDIA GeForce RTX 3090 Ti (24 GB)
운영체제	Ubuntu 24.04.4 LTS
소프트웨어	Python 3.12.3, numpy 2.5.1, scipy 1.18.0, cupy 14.1.1, CUDA 13.3.1
측정 대상	초등 단계 얼린 모델의 어휘, 음절과 묶음 연산면

3절의 실측 수치는 위 환경에서 측정했다. 로우랭크 헤드 파라미터 수는 구조적 계산값이다.

부록 C. 재현 안내

본고의 실측 수치는 공개 재현 패키지로 재생된다. 아래 다섯 단계를 순서대로 하면 된다.

1. 요구 환경. NVIDIA 그래픽 처리장치(GPU)와 CUDA, Python 3.12 가 필요하다. 파이썬 패키지는 압축을 푼 폴더에서 다음으로 설치한다. cupy 는 설치된 CUDA 버전에 맞는 배포판을 사용한다(예: CUDA 13 은 cupy-cuda13x).

```
pip install -r requirements.txt
```

2. 코드 받기. 압축 묶음(zip)을 내려받아 푼다. 저장소로 받으려면 github.com/ubmscoin/banya/banya_ai_paper_code 폴더다.

```
wget https://ubmscoin.github.io/banya/banya_ai_paper_code/
banya_ai_paper_code.zip
unzip banya_ai_paper_code.zip && cd banya_ai_paper_code
```

3. 모델 받기. 얼린 모델 3개(합계 471MB)는 제노도 doi.org/10.5281/zenodo.21383724 에 있다. 다음 한 줄이 내려받기와 무결성 검증(sha256, 파일 지문 대조)을 함께 한다. 손으로 받으려면 위 기록에서 세 파일을 받아 model 폴더에 넣고 `sha256sum -c checksums.sha256` 으로 검증한다.

```
cd model && bash download.sh && cd ..
```

4. 말뭉치 만들기. 말뭉치는 원문 배포 없이 생성기가 만든다. 다음 한 줄이 banya_world_data 폴더에 말뭉치 23종을 전부 생성한다.

```
cd corpus_build && bash build_all.sh && cd ..
```

5. 실측 재생. probe 폴더의 프로브가 각 절의 수치를 재생한다. 본고의 수치는 다음 프로브가 담당한다. 출력에 목표값이 함께 인쇄되므로 본문과 바로 대조된다.

프로브	재생하는 수치	확인할 출력
paper5_cache_token_probe.py	3.1 비용, 3.2 묶음층 구조	어휘 10724, 로우랭크 7.3배, 묶음층 일관성 8.1배, 직교쌍 79% 와 77%
paper5_quality_contrast_probe.py	3.3 절감과 품질	켬 5.08 과 5.07 비트, 끔 7.98 과 7.48, 재정규화 7.58 과 7.07, 절감 21.5%

패키지의 폴더 구성과 파일별 설명은 목차 페이지에 있다.

반야 연구 시리즈 제5편(캐시 사전 토큰화, 동적 임베딩). 모든 수치는 이 글에서 부록 B 환경으로 직접 측정하거나 구조에서 계산한 값이다. 배경 자연관은 선행 공리 문헌 [1]로 인용하되 이 글 주장의 근거가 아니다. 제1편, 제3편을 인용한다. 창발 제어 신호는 제6편에서, 자기 발화 결합은 마지막 편에서 다룬다.