

Bekko Embedding: 超小型エンコーダによるパラメータ効率の高い多言語検索

Yuichi Tateno

連絡先: hotchpotch@gmail.com

Abstract

情報検索 (IR) を支える dense retrieval モデルは大規模化が進み、数百M~数Bパラメータのモデルが主流となった。CPU のみのサーバ、エッジデバイス、ブラウザでは、こうしたモデルの推論時間とメモリが大きな負担となる一方、高い検索性能・広い多言語対応・小型高速の3つを兼ね備えたモデルはほとんど存在しない。本稿では、超小型の文脈エンコーダを核とする、パラメータ効率の高い多言語検索モデルファミリー Bekko Embedding を提案する。推論の演算量 (FLOPs) を主に決めるのは、全トークンが通過する Transformer 層のパラメータであり、本稿ではこれを Active Parameters (AP) と呼ぶ。Bekko が極小化するの、まさにこの AP である。リリースモデル bekko-embedding-v1-a8m と、より高性能な bekko-embedding-v1-a25m (以下 a8m / a25m) は、それぞれ AP 8M 弱、25M 弱である。

学習レシピは意図的に単純である。多言語エンコーダ mmBERT-small の22層を4層/13層へプルーニングし、プルーニング済みモデルを base model として二段階で学習する。第一段階では公開多言語コーパス由来の約11億ペア (二つの相補的な LLM 合成データセットを含む) で大規模対照学習を行い、第二段階では hard-negative ファインチューニングを行う。両段階では、ペアタイプ別に損失方向を切り替えるマスク付き対照損失と Matryoshka 表現学習を組み合わせた共通の損失を用いる。教師モデルによる蒸留は用いず、全学習は単一GPU で完了する (a8m で約3日)。この小ささにもかかわらず、両モデルは AP が1~2桁大きい既存モデルに検索性能で匹敵または上回る。本稿の比較では、公式 MMTEB Multilingual v2 Retrieval (nDCG@10) で a8m (56.2) が multilingual-e5 系や AP 40倍の BGE-M3 を上回り、a25m は 57.5 に達する。軽量評価の Multilingual NanoBEIR (14言語) でも同じ傾向を確認した。出力は 384 次元 (256/128/64 次元へ削減可) であり、埋め込み後の類似度計算とインデックスも低コストに保てる。

小さな AP は推論速度に直結する。同一ワークロード・同一ハードウェアで実測した比較モデルの中で、a8m は CPU・GPU の双方で最速であり、multilingual-e5-small 比で x86 CPU 約1.6倍、GPU 約1.5倍、Raspberry Pi 5 でも最速である。汎用の PyTorch 重みに加え、語彙埋め込み行列を row-wise int8 圧縮した ONNX / OpenVINO 版も提供する。後者は a8m のモデルファイルを fp32 配布比 約1/3 の 124 MiB に縮め、Transformers.js によるブラウザ実行を可能にする。効率的な多言語検索の再現可能な研究に向けて、モデル重み・第一段階コーパスの全量・独自にマイニングした第二段階 hard negatives を公開している。

1. はじめに

1.1 encoder モデルの実務的価値

BERT (Devlin et al., 2019) 以降、encoder-only Transformer は情報検索 (IR)、分類、固有表現認識などの非生成的応用において中核的な役割を担い続けている。LLM の急速な発展にもかかわらず encoder モデルが実運用で使われ続ける理由は、その推論コストの低さとサイズあたりの品質の高さにある。LLM は強力な埋め込みソースとなりうるが、推論時のレイテンシとメモリフットプリントの大きさから産業応用ではしばしば非実用的である (Granite Team, 2025a)。Weller et al. (2026) は、同一サイズの encoder/decoder を揃えた Ettin スイートを用いた体系的比較で、encoder-only モデルが分類・検索タスクで decoder-only モデルを上回ることを報告している。特に IR 領域では、bi-encoder による埋め込み検索が、大量の文書集合から関連文書を高速に絞り込む一次検索の中核として (Reimers & Gurevych, 2019; Karpukhin et al., 2020)、また Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) の検索基盤として不可欠である。

1.2 効率の正しい軸: 総パラメータではなく Active Parameters

近年の埋め込みモデル研究の主流は大規模化に向かい、GTE-multilingual-base (Zhang et al., 2024) は 305M、BGE-M3 (Chen et al., 2024) は 568M、Qwen3 Embedding (Zhang et al., 2025) は 0.6B~8B のパラメータを用いる (パラメータ数は各公開モデル・モデルカードの実測値)。本稿はこの潮流に対し、まず「サイズ」をどの軸で測るかを問い直す。

多言語モデルでは、総パラメータの大半を語彙埋め込み行列 (語彙 × 隠れ次元) が占める。例えば mmBERT-small (Marone et al., 2025) (総 140M) では $256,000 \times 384 \approx 98M$ がこの行列で、非埋め込みは 42M に過ぎない。しかしこの行列は推論

時にトークン id で1行を引くだけの静的 lookup テーブルであり、行列積を伴わず、トークンあたり 1 行 (hidden 次元分) の読み出しに過ぎない (語彙サイズに依存しない) (出力側も hidden state を pooling するため vocab $\times$ hidden の射影は走らない)。一方、全トークンが必ず通過する Transformer 層のパラメータが、推論の演算量 (FLOPs) を主に決める。本稿では「埋め込みテーブルを除く、文脈エンコーダが推論時に用いる学習パラメータ (Transformer ブロック・正規化・pooling/射影)」を Active Parameters (アクティブパラメータ; 以下 AP) と定義する (Mixture-of-Experts 文脈で使われる active parameters とは異なる、本稿での定義である)。非埋め込みパラメータを効率/スケールの軸とする見方自体は既出である——Kaplan et al. (2020) は scaling 則でパラメータ数 N を「語彙・位置埋め込みを除く数」と定義し、ALBERT (Lan et al., 2019) は語彙埋め込みを隠れ次元から分離し圧縮した。本研究はこの軸に立ち、AP の小型化による FLOPs 削減 (推論の高速化) を主軸とし、さらにメモリ・配布サイズ削減のために lookup テーブル (語彙埋め込み行列) の int8 圧縮による小型化も併せて行う (§7)。なお FLOPs は系列長・attention パターン・hidden 次元にも依存し、実レイテンシは backend カーネルやメモリ帯域にも左右されるため、AP は推論コストの唯一の決定因子ではなく主要な相関軸である。同程度の AP でもアーキテクチャや推論実装によって実速度は変わるため、本稿では実測スループットも併せて示す (§7.2)。

以上を踏まえ、本論文では埋め込みモデルの演算効率の軸として、総パラメータではなく AP を用いる。総パラメータ基準の比較は、巨大な多言語トークナイザを持つ小型モデルを過大に「大きく」見せる。例えば multilingual-e5-small (Wang et al., 2024) は総 118M とされるが、語彙埋め込みを除いた AP は約 21M に過ぎない。本論文はこの軸を一貫して採用し、(i) AP を極小化したモデルが一桁～二桁大きいモデルに匹敵しうる (§5)、(ii) エッジ・低スペック環境向けの配布では、配布サイズの大半を占める静的な語彙埋め込みテーブルを int8 化することで、Transformer の演算に影響を与えずに配布サイズ・ロード時メモリを大きく削減できる (§7)、を示す。

もう一つの効率軸が出力埋め込みの次元数である。埋め込みモデルの推論コストに加えて、埋め込み後の検索フェーズでは、ベクトル類似度計算とインデックスのメモリが出力次元に比例して増える。近年の大規模モデルは 1024～4096 次元級の出力を持つのに対し、Bekko の標準出力はわずか 384 次元 (Matryoshka でさらに 256/128/64 へ削減可; §3.4) であり、この小ささは大規模文書集合に対する検索・インデックス運用のコスト削減に直結する。

AP 軸での Bekko の位置づけを図1に先取りして示す。検索品質 (多言語 IR ベンチマーク HAKARI-Bench の総合値; §4.1) 対 AP の平面で、a8m・a25m は統一比較集合 (§4.2) の効率フロンティア上に位置し、フロンティアを AP 8M～25M の超小型帯まで押し広げる——AP が1～2桁大きいモデルと同水準の検索品質を、はるかに小さい推論コストで達成する。

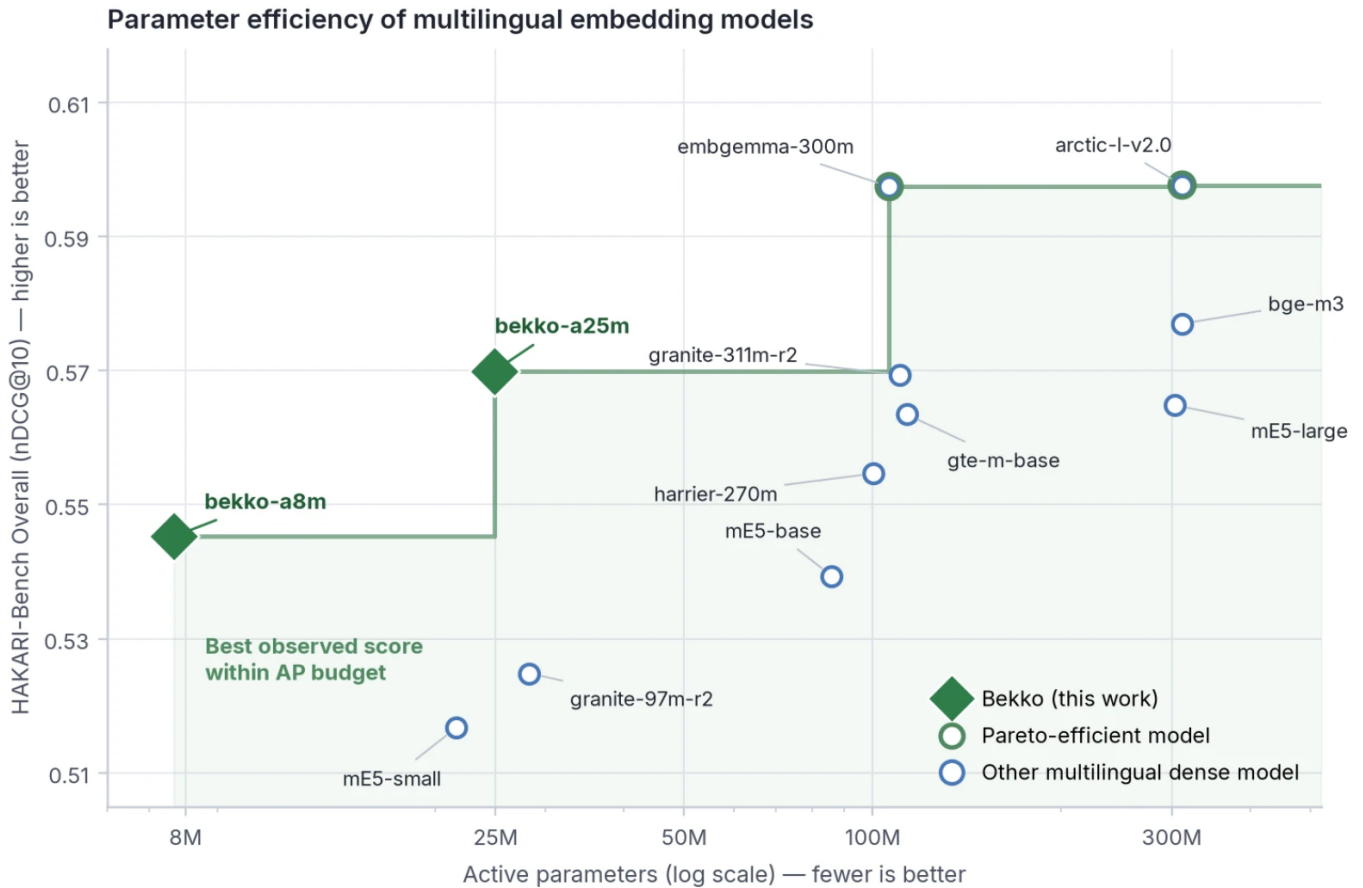


図1: 多言語埋め込みモデルのパラメータ効率——検索品質 (HAKARI-Bench Overall; 著者作成の軽量ベンチ, §4.1) 対 Active Parameters (対数軸)。階段線は所与の AP 予算内で観測された最高スコアを示し、実測モデル間を補間しない。a8m・a25m は統一比較集合 (§4.2) の効率フロンティアを AP 一桁~二桁小さい帯へ押し広げる。主評価である公式 MMTEB Multilingual v2 でも同じ構図を確認する (§5.1, 図4)。

1.3 小型・オンデバイス需要と、現代的な多言語・超小型モデルの不在

小型・高速・ローカル実行可能な検索モデルへの需要は、実世界の IR システムで複数の観点から高まっている。(1) 数百万〜数十億件の文書埋め込みでは、推論コストが処理時間とインデックス構築費を直接規定する。(2) ブラウザ・スマートフォン・CPU のみのサーバといった、ハイスpek的な GPU を持たない手元の環境で動く AI エージェントが実用域に入り、外部に送らず端末内で完結する低レイテンシ検索の需要が顕在化している——これらの環境の推論は、CPU / WebAssembly、あるいはブラウザの WebGPU やスマートフォンの内蔵アクセラレータといった限られた計算資源で行われる。実際、EmbeddingGemma (Schechter Vera et al., 2025) は "low-latency and high-throughput use cases such as on-device applications" を、Granite Embedding R2 (多言語; Granite Team, 2026) は AP 28M の小型版を "latency-sensitive production workloads, edge deployment" 向けと明記する (granite-embedding-97m-multilingual-r2 モデルカード)。(3) 新規 (未キャッシュ) の query の埋め込みは検索時にオンラインで計算されるため、低レイテンシ推論が検索の体感速度を左右する (Reimers & Gurevych, 2019)。

ところが、多言語かつ超小型の帯域は世代が古い。小型 AP (30M 以下) の多言語モデルとしては、XLM-R (Conneau et al., 2019) から蒸留された multilingual MiniLM を初期値とする Multilingual E5-small (Wang et al., 2024) などが長らく主流であり——mE5-small の実体は AP 約 21M の 2020 年世代 MiniLM である——、2026 年になって ModernBERT 系設計の Granite Embedding R2 (AP 28M) (Granite Team, 2026) が登場した。しかし、少なくとも本稿の比較基準 (商用利用可能なライセンスで重みが公開された多言語 dense 埋め込みモデル; §4.2) の範囲では、本研究が狙うさらに小さい AP 10M 以下の超小型帯を担う多言語文脈エンコーダは見当たらない。さらに小さい帯では Model2Vec (Tulkens & van Dongen, 2024) や Sentence Transformers の静的埋め込み (Aarsen, 2025) が CPU 高速性で存在感を持つが、これらは文脈を持たない静的トークン埋め込みであり、文脈エンコーダとは品質帯が異なる。

1.4 本研究のアプローチと貢献

本研究は、最新のアーキテクチャと学習手法を超小型モデルへ統合する設計最適化で、上記ギャップに応える。鍵は2点。第一に、検索に有用な多言語知識は基盤 mmBERT-small の事前学習済み重みに既に宿っており、それを構造的レイヤープルーニングで保持したまま層数だけ削り、プルーニング済みモデルを base model（学習の初期値とする基盤モデル）として対照学習することで、蒸留なしで AP を一桁 M 台（800万弱）に圧縮したまま学習を成功させられる。第二に、小型モデルが取りこぼす表現力は、広範なドメイン・言語の対照学習データと高品質な合成・hard negatives で補える（データ設計の詳細と先行研究への依拠は §3.3）。貢献は次の通り：

1. AP 軸の一貫適用: 非埋め込みパラメータを効率軸とする見方（Kaplan et al., 2020; Lan et al., 2019）を、多言語埋め込みモデルの比較・超小型化・int8 圧縮・エッジ配布へ一貫適用する (§1.2, §4.2, §7)。軸自体は既出だが、埋め込みモデルへの体系的適用と、それに基づく配布最適化が本研究の立場である。
2. 蒸留なしの構造的レイヤープルーニング: mmBERT-small の22層を 4層/13層へ削り、プルーニング済みモデルを base model として使うことで、AP 7.67M の `bekko-embedding-v1-a8m` と 24.93M の `bekko-embedding-v1-a25m` の学習を成功させた。教師モデルを要さないため、プルーニングと公開データの対照学習のみでどこまで到達できるかを切り分けて示せる (§3.2)。プルーニングパターンのアブレーション評価では、先頭の連続レイヤーに加え、最終層ではなく深層の Global レイヤー（特に18）を残すことが有効という観測を得た (§6.1; 検索知識の因果的局在を主張するものではない)。
3. 多言語・多ドメインデータと二系統 LLM 合成: 100言語超をカバーする多ドメイン対照学習データに、文脈を読む高品質合成 (Qwen3.5-35B-A3B) と高速合成 (自作 query-crafter-multilingual) の二系統を組み合わせる (§3.3)。
4. 単一GPU 学習とオープン公開: AP の小型化は学習の計算量も減らし、全学習を単一GPU で完了する (§3.5)。第一段階データ・hard negatives・モデル重みを公開している。
5. オンデバイス実行の実証: 嵩む語彙埋め込みテーブルを row-wise int8 化（テーブル単体で約 1/4）した ONNX / OpenVINO 配布物でモデルファイルを大きく圧縮し（a8m 124 MiB, fp32 配布比 約 1/3）、Raspberry Pi 5・CPU・ブラウザでの実用動作を示す (§7)。

本稿の構成: §2 関連研究、§3 設計（3つの原理）、§4 評価設定、§5 検索品質、§6 分析（プルーニング・学習レシピ・QAT）、§7 効率とデプロイ、§8 限界、§9 結論。本稿の全数値は固定済みの評価スナップショット (§4.2, 付録 G) に基づく。

2. 関連研究

2.1 Bi-Encoder と多言語埋め込み

Sentence-BERT (Reimers & Gurevych, 2019) は BERT を bi-encoder として再設計し、効率的な文埋め込み検索を確立した。多言語化では、並列データによる知識蒸留 (Reimers & Gurevych, 2020) と、XLM-R (Conneau et al., 2019) に代表される大規模多言語事前学習が主要な系譜である。LaBSE (Feng et al., 2022) は100言語超の共有埋め込み空間を、Multilingual E5 (Wang et al., 2024) は英語 E5 のレシピを多言語へ拡張し small/base/large を提供した。日本語では Ruri (Tsukagoshi & Sasano, 2026) が、体系的なベア設計と合成データ・reranker フィルタにより小型でも高い汎用性を示した。Bekko はこの流れに対し、巨大化ではなく効率的なアーキテクチャ設計と Ruri に倣ったデータ戦略によって小型帯での競争力を追求する。

2.2 データ中心の多段階対照学習

E5 (Wang et al., 2022) は弱教師あり大規模対照学習の有効性を示し、GTE (Li et al., 2023) はこれを多段階学習（大規模弱教師あり事前学習+多タスク教師ありファインチューニング）として体系化した。GTE は対照事前学習のバッチサイズが約1万で性能が飽和すると報告しており、本研究のバッチサイズ設計の根拠となる (§3.4)。mGTE (Zhang et al., 2024) は GTE の多言語拡張として RoPE と unpadding を導入し8192トークン学習を実現した。Qwen3 Embedding (Zhang et al., 2025) は LLM を基盤とした最新世代の多段階学習を報告し、in-batch negatives の偽陰性を抑えるマスク付き対照損失を採用している（本研究の損失設計の直接の参照, §3.4）。BGE-M3 (Chen et al., 2024) は dense/sparse/multi-vector を単一モデルで統合し、トークン長ごとのグループ学習と段階移行を提案した（本研究のパイプライン・トークン長グループに影響, §3.5）。FineWeb (Penedo et al., 2024) と FineWeb-2 系 (Messmer et al., 2025) は学習データの品質選別が性能を大きく左右することを示した。文書から LLM でクエリを合成する IR データ拡張は InPars (Bonifacio et al., 2022)・Promptagator (Dai et al., 2023) などが先駆けとなり、SWIM-IR (Thakur et al., 2024) が多言語へ拡張した (§3.3)。

2.3 効率的エンコーダ設計とモデル圧縮

ModernBERT (Warner et al., 2024) は encoder-only 設計を現代的最適化で刷新し、ネイティブ8192トークン・flash attention・unpadding 等で Pareto 改善を実現した。mmBERT (Marone et al., 2025) はそのアーキテクチャを3兆トークンの多言語データで事前学習した。Bekko はこの mmBERT-small を基盤とする。

モデル圧縮の主要手法には、DistilBERT (Sanh et al., 2019) に代表される知識蒸留と、構造的レイヤープルーニング（層削除）がある。Bekko は後者を基盤モデルの構築に用いており（設計を構成する一要素; §3.2）、先行研究の系譜がある。Sajjad et al. (2023, "Poor Man's BERT") は、事前学習済み encoder から蒸留なしで層を削っても下流性能を概ね維持できることを示した。decoder LLM では ShortGPT (Men et al., 2024) や Gromov et al. (2024) が、深層の多くが冗長であり、層を削除したのち軽い継続学習で「治癒 (heal)」させれば性能劣化は小さいと報告している。本研究はこの知見を多言語検索エンコーダへ適用し、さらに「どの層を残すか」を検索タスクで検証する (§6.1)。

私たちの知る限り最も近い先行は Granite Embedding Multilingual R2 (Granite Team, 2026) である。その小型版 granite-embedding-97m-multilingual-r2 は、ModernBERT 系の多言語エンコーダを基盤とし、大型版（311M・22層）に対して model pruning と vocabulary selection による小型化（12層・語彙は GPT-OSS 系 200K→180K トークン）と論文では説明される（重みの初期化には英語版の 12層 small モデル (Granite Team, 2025b) を用いる）。総パラメータ数は 97M で、論文の表では Active Parameters が 28M と報告されている。学習面では、対照学習に加えて大型 decoder 系教師からの knowledge distillation を併用し、英語教師には English R2 と同じ Mistral-7B-Instruct 系教師を、多言語教師には Granite 8B 系教師を用いる (Granite Team, 2025b; Granite Team, 2026)。Bekko は次の3点で差別化される：(a) 教師モデル・蒸留を一切用いない、(b) より激しい削減（22→4層, AP 8M 弱）、(c) AP の小型化により学習に必要な計算資源も減り、単一GPU（一般的なワークステーション）で全学習を完了できる。EmbeddingGemma (Schechter Vera et al., 2025) は Gemma 3 を基盤に encoder-decoder 初期化・埋め込みの幾何蒸留で AP 106M の on-device 向け SOTA を達成するが、これも蒸留に依存し AP も一桁大きい。Model2Vec / 静的埋め込み (Tulkens & van Dongen, 2024; Aarsen, 2025) は CPU 高速だが文脈を持たない点で Bekko と品質帯が異なる。

表1 に主要なマルチリンガル埋め込みモデルとの位置づけを示す。Bekko は、AP が一桁小さく（7.67M）、文脈エンコーダで、蒸留を用いず、重みも多言語学習データも公開という組み合わせで、表中の他のどのモデルにもない位置を占める。特に、多言語の大規模事前学習（第一段階）データまで公開している埋め込みモデルは私たちの知る限りほとんどなく、この公開性は多言語小型モデル研究の再現・発展に直接寄与する。

表1: 主要マルチリンガル埋め込みモデルとの位置づけ（AP = 非埋め込みパラメータ）。本表は設計方針（蒸留・公開性）の位置づけを示す代表例であり、性能比較の統一モデル集合（AP ≤ 350M の多言語 dense + BM25、計13モデル）は §4.2 に定義する。

モデル	AP	総パラメータ	最大長	蒸留の利用§	重み公開	学習データ公開
bekko-a8m	7.67M	106M	8192	不使用	○	○（一段・二段とも◇）
mE5-small	21.6M	118M	512	初期値のみ（MiniLM が蒸留由来）	○	△ 一部
bekko-a25m	24.9M	123M	8192	不使用	○	○（一段・二段とも◇）
granite-97m-r2	28.3M	97M	32768	学習時（教師スコア蒸留）	○	× 非公開†
embgemma-300m	106.3M	308M	2048	学習時（教師埋め込みへの整列）	○	× 非公開††
gte-m-base	113.3M	305M	8192	—	○	△ 一部
bge-m3	311.8M	568M	8192	自己蒸留（統合スコアが教師）	○	△ 一部
Model2Vec ‡	—（静的）	～数十M	—	静的化のための蒸留	○	—‡

§: 「蒸留の利用」は、構築過程のどこで教師由来の蒸留が入るかを区別する。mE5-small は学習自体に蒸留を用いないが、初期値の multilingual MiniLM 自体が蒸留で作られたモデルである。granite R2 は fine-tuning 時に大型教師（Mistral-7B / Granite-8B 系を埋め込みモデル化したもの）の類似度スコア分布への蒸留損失を用いる。EmbeddingGemma は教師 Gemini Embedding の埋め込み空間へ直接整列させる蒸留を学習に用いる。bge-m3 は外部教師ではなく、自身の dense / sparse / multi-vector の統合スコアを教師信号とする損失レベルの自己蒸留である（hard negatives のマイニングとは別の機構）。◇: Bekko の学習データ公開の内訳は、第一段階は使用データ全量を単一の公開データセットとして公開し、第二段階は独自マイニング分（hard negatives）を公開、残りは既に公開されている上流データセット（Ruri v3 FT・BGE 系サブセット等; 付録 C）で構成する——すなわち両段階とも全構成要素が公開データである（第二段階を単一データセットとして再配布しているわけではない）。†: 学習データは「permissive で enterprise 向きのライセンス」と説明されるが、IBM 内部データ・IBM 生成合成データを含み、データセット自体は公開されていない（モデルカード Data Collection 節）。††: 公開されるのはチェックポ

イントのみで、学習データは構成（Web の title-body ペア、Gecko の academic datasets の subset、Gemini Embedding の合成データ等）が概説されるにとどまり、データセット・混合比は非公開。‡: Model2Vec のみ文脈を持たない静的埋め込みで、他は全て文脈エンコーダ。静的化の蒸留は語彙と教師モデルのみで行われ対照学習データを持たない（派生の Potion 系が用いる Tokenlearn 事前学習は公開コーパス C4 を使用）。表中の全モデルは多言語対応（mE5 は約100言語、gte-m-base は70+、bge-m3 は100+、granite は52 言語を強化（評価は200+ 言語）、embgemma・Bekko は100+; Model2Vec はvariant による）。

2.4 評価ベンチマークの採択

評価面では MTEB (Muennighoff et al., 2022) が包括的評価を、MMTEB (Enevoldsen et al., 2025) が500超タスク・250超言語へ拡張した。本研究の評価は、(1) 中心的な主張は著者が作成したものに依存しない独立のベンチマークで確認できること、(2) 多言語の検索性能を広くカバーすること、(3) 多数のモデル・設定を統一条件で比較できる軽量性、の3点を満たすように選ぶ。検索の軽量評価としては、BEIR (Thakur et al., 2021) のうち 13 データセットを各 50 クエリ・小型コーパスに縮約した NanoBEIR (Zeta Alpha; Câmara, 2024) があり、その機械翻訳版がコミュニティから公開されている（LightOn 8言語、Liquid AI 日・韓、Serbian AI Society セルビア語、sionic-ai タイ語・sionic-ai ベトナム語）——NanoBEIR とその翻訳版はいずれも本研究の著者が作成したものではない。本研究は、公式 MMTEB Multilingual v2 と、英語オリジナルの NanoBEIR にこれら第三者作成の翻訳版 13 言語を加えた Multilingual NanoBEIR (14言語) を主評価に用い、HAKARI-Bench (Tateno, 2026) を補助比較・開発時 proxy として用いる。HAKARI-Bench は本研究の著者自身が作成・公開した情報検索用ベンチマークで、MMTEB の検索タスクや BEIR を含む多言語・多様なタスクを小型の Nano サブセットに縮約した 500 以上のタスクから成り、多数のモデルを軽量に統一条件で比較できる。Nano 順位が公式順位（MTEB, MMTEB, BEIR の共通モデル・共通タスク上での比較）を Spearman > 0.97 で再現すると報告されている。

3. Bekko の設計 — 3つの原理

Bekko の設計は、§1.4 で述べたアプローチを3つの原理に整理して具体化したものである。原理①: 事前学習済みの多言語知識を保持したまま AP を削る（構造的プルーニング, §3.2）。原理②: 小型モデルの知識不足を、広範なデータと二系統の合成データセットで補う (§3.3)。原理③: 現代的な対照学習レシピで品質を引き出す (§3.4)。これらすべてを単一GPU で実行する (§3.5)。全体像を図2に示す。

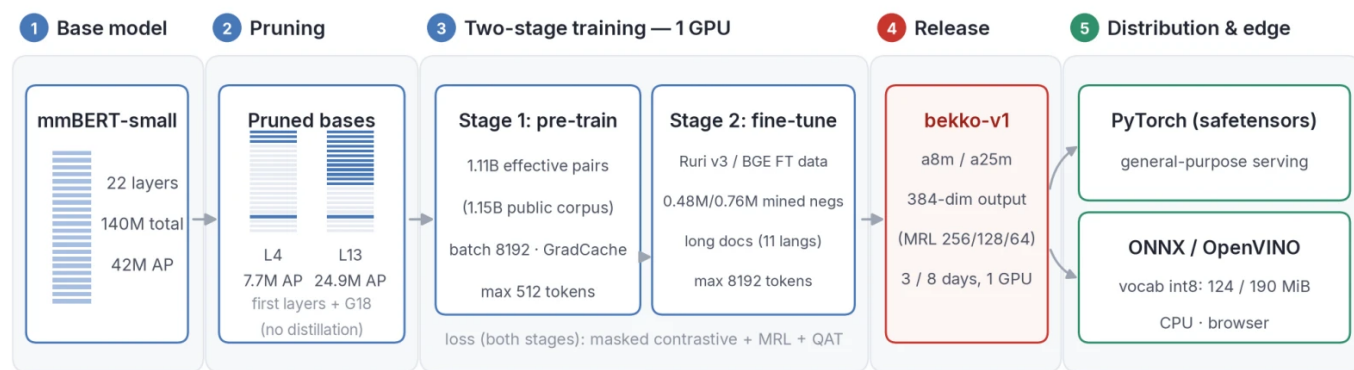


図2: Bekko の構築パイプライン全体像。蒸留を用いず、プルーニング済みモデルを base として二段階対照学習を単一GPU で完了し（a8m 約3日/a25m 約8日）、ONNX / OpenVINO 配布物では語彙埋め込みの int8 化で 124/190 MiB を実現する (§3.2–3.5, §7.1)。

3.1 基盤モデル: mmBERT-small

基盤として jhu-clsp/mmBERT-small (Marone et al., 2025) を採用する。ModernBERT アーキテクチャに基づく多言語エンコーダで、主な仕様は次の通り。

項目	値
アーキテクチャ	ModernBERT (encoder-only)
Total Parameters	140M
Active Parameters (AP)	42M
総レイヤー数	22
隠れ層次元 (hidden_size)	384
アテンションヘッド数	6
中間層次元 (intermediate_size)	1152
語彙サイズ	256,000 (Gemma 2 tokenizer)
最大位置埋め込み	8,192
Global Attention 間隔	3層ごと
Local Attention 窓幅	128
RoPE theta	Global / Local とも 160,000

ModernBERT の重要な設計特性として、Global-Local-Local (G-L-L) の繰り返しパターンがある。22層のうちレイヤー 0, 3, 6, 9, 12, 15, 18, 21 が全系列への Global Attention (G) を持ち、残りは窓幅128の Local Attention (L) で動作する (図 3)。このリズムの維持はプルーニングの設計制約となる (§3.2)。RoPE theta には ModernBERT と mmBERT で違いがある——ModernBERT (Warner et al., 2024) は Global 層 160,000 / Local 層 10,000 と使い分けるのに対し、mmBERT は Global / Local の両方を 160,000 に設定しており、より長文脈向けの構成である。これが第一段階からの長文耐性に寄与する可能性がある (§5.3)。mmBERT-small を選ぶ理由は、3兆トークン・1800以上の言語の事前学習による広範な言語知識、hidden_size=384 の小型設計、ModernBERT の最新最適化 (flash attention, RoPE (Su et al., 2021)) にある。

3.2 原理①: 知識を保持する構造的レイヤープルーニング

mmBERT-small の22層から検索に有効なレイヤーのサブセットを選び、純粋な構造的レイヤー除去でプルーニングモデルを構築する。保持レイヤーの重みは変更しない (蒸留・追加事前学習を要さない)。

レイヤー選択は3つの原則による：(1) 低レベル表現を担う初期連続レイヤーを保持、(2) できる限り G-L-L リズムを保つ、(3) 長距離依存のため深層 Global レイヤーを少なくとも1つ含める。どのパターンを残すべきかは検索性能に強く効くため、§6.1 で独立に分析する。結論を先取りすると、最終層 (21) ではなく遠めの深層 Global 層 (18) を残す構成が良好である (層数が多いほど差は明瞭; §6.1)。リリースモデルの保持パターンとパラメータは次の通り。

モデル	基盤 (pruned)	保持レイヤー	層数	AP	Total
bekko-embedding-v1-a8m	mmBERT-L4H384-pruned	[0, 1, 2, 18]	4	7,671,168	105,975,168
bekko-embedding-v1-a25m	mmBERT-L13H384-pruned	[0,1,2,...,11, 18]	13	24,930,432	123,234,432

§1.2 の通り、総パラメータは語彙埋め込み行列 (≈98M) が支配的だが、推論コストを決めるのは AP (a8m で約7.7M) である。プルーニング済みモデルは、そのまま埋め込みモデルとして使うためのものではなく、対照学習の base model として用いる。

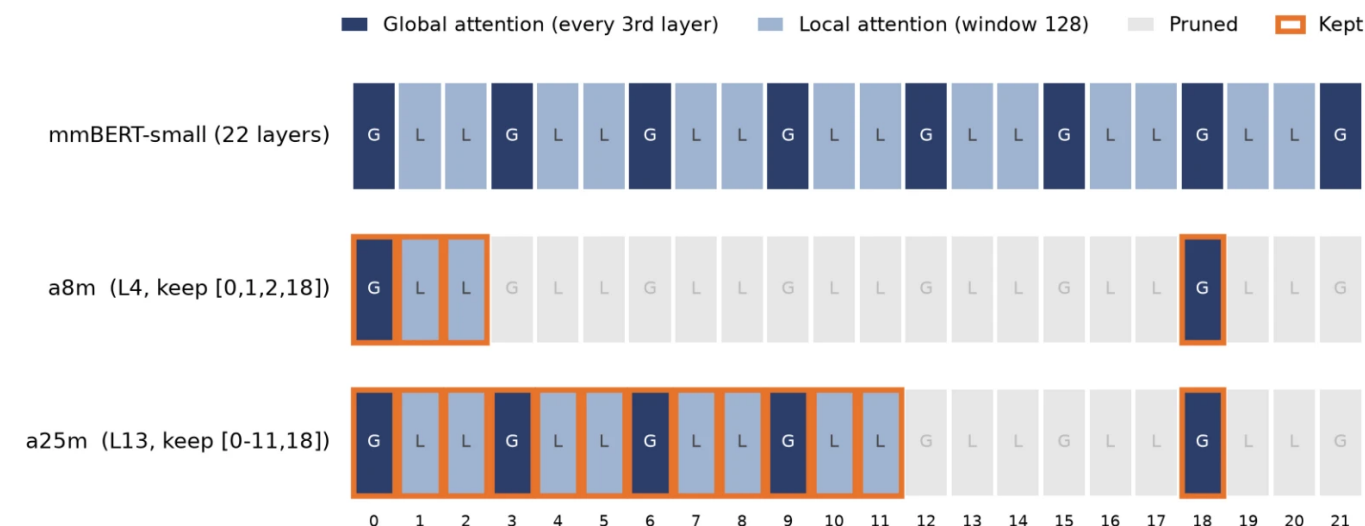


図3: 構造的レイヤープルーニング。最終レイヤー（21）ではなく深層 Global レイヤー（18）を加える設計を採用する（層選択の比較は §6.1）。

3.3 原理②: 知識を補うデータ

Bekko のデータ設計は、Ruri (Tsukagoshi & Sasano, 2026) のペア設計を多言語へ一般化したものである。Ruri のデータ設計の要点は3つある。(1) LLM による合成データセットの作成——日本語 Wikipedia の文章から LLM で質問・検索クエリを生成した AutoWikiQA（約250万件）等、Ruri は 1億件超の合成データを対照事前学習に投入する。(2) 教師なしの query-doc 大規模構築——Wikipedia・ニュース・学術データ等から「（節）タイトル→段落・本文」ペアを機械的に抽出し、人手アノテーションなしで非対称ペアを得る。(3) 教師なしの doc-doc 大規模構築——同一記事内の別の文・段落を関連テキストとみなすペア抽出。これらを統合した約2.2億ペアの対照事前学習に、BM25 hard negatives・reranker フィルタ・二段階学習を組み合わせ、小型でも高い汎用性を達成した（二段階学習・LLM 合成自体の先行研究は §2.2 の E5・GTE・InPars・Promptagator・SWIM-IR を参照）。本研究はこの (1)–(3) のペア設計を多言語へ拡張する。

第一段階データ（公開: bekko-embedding-v1-unsupervised）

第一段階データは合計約11.5億ペアで、2,452 サブセットから構成され、Hugging Face 上に [bekko-embedding-v1-unsupervised](#) として公開している（統計は公開データからの実測；付録 B）。リリースモデルの学習では、後述の対訳品質フィルタとブロック整形の結果、このうち 1,863 サブセット・実効約11.1億行（公開コーパスの約 97%）を実際に使用した。ペアは2タイプに大別される：doc_doc（文書↔文書の対称ペア——対訳・関連段落など；1,975 サブセット・約4.8億行）と query_doc（対象文書を指す短いテキスト——検索クエリ・記事タイトルや見出し・長文の質問など——と文書の非対称ペア；477 サブセット・約6.6億行）。うち 34 サブセットは明示的な hard negative を持つ triplet 形式だが、役割上はそれぞれのペアタイプに含まれる。系統は次の通り。

系統	主なデータソース	タイプ	行数 (概数)	概要
対訳 bitext	NLLB, CCMatrix / Opus-100 / WikiMatrix / WikiTitles	doc_doc	3.3億	1,600 超の言語ペア（非英語間を含む）。英語対訳の一部は reranker + 埋め込みモデルでフィルタした高品質サブセット
ニュース	Multilingual CC-News (47言語) / 関連段落ペア (134言語)	query_doc / doc_doc	1.5億	見出し→本文 (query_doc) ・記事内/記事間の段落ペア (doc_doc)
Wikipedia / FineWiki	finewiki (314言語) / wikipedia-multilingual の IR・lead・段落ペア	query_doc / doc_doc	1.9億	title→本文 (query_doc) 、関連段落・lead 段落の意味的近接ペア (doc_doc)
高速 LLM 合成	FineWeb2-IR (21言語) / wikipedia-multilingual synthetic query (11言語)	query_doc	1.2億	query-crafter-multilingual で合成（後述）
高品質 LLM 合成	fineweb-edu / arXiv abstract / PubMed abstract / CC-News / 英語 Wikipedia	query_doc	0.4億	Qwen3.5-35B-A3B で合成（後述）
既存 IR ペア集合	nomic unsupervised / lighton / swim-ir / all-nli / miracl 等	query_doc / doc_doc	3.1億	既存の弱教師ありペア・QA・NLI

本稿で Bekko を 100 言語超対応と述べる根拠は、この第一段階データの言語カバレッジである。対訳 bitext は NLLB・CCMatrix 由来の多対多対訳を正規化 1,622 言語ペア（約3.3億行）含み、Wikipedia 系 (finewiki) は 314 言語、ニュース系

は 47～134 言語をカバーする（付録 B）。言語横断の整合性は MMTEB BitextMining の結果（§5.1）でも裏付けられる。ただし行数は英語・主要言語に偏っており、低資源言語の性能は限定的である（§8）。

学習時には2つの前処理を適用する（詳細は付録 B）。(1) 対訳品質フィルタ: NLLB・CCMatrix 由来の多対多対訳は機械マイニングによる低品質な翻訳ペアを含むため、言語ペア（サブセット）ごとにサンプル抽出した対訳を監査して品質を採点し、あからさまに低品質な言語ペアを除外した——結果、多対多サブセットの 1,575 言語ペアのうち約 1,300 を使用する（前述の正規化 1,622 ペアは全対訳系統を跨いだユニーク数）。なお英語対訳の高品質サブセットは、これとは別に reranker と埋め込みモデルによる行単位のフィルタで構築している（付録 B）。(2) 重複排除ブロック: in-batch negatives に同一・重複テキストが混入すると偽陰性となるため、各サブセット内でランダムサンプルした 8,192 行のブロック（ブロック内に query / doc の重複なし）を事前構成し、学習時は 1 ブロックを 1 バッチとして用いる。なお第一・第二段階とも、1 つのバッチは常に単一のサブセットだけから構成し、複数のサブセットを 1 バッチに混ぜない——バッチ単位でサブセットを行数比例で選ぶ。これにより in-batch negatives が同一ソース・同一分布のテキストで揃い、負例の難度が安定する。

二系統の合成データ。文書から LLM でクエリを合成する手法は InPars (Bonifacio et al., 2022)・Promptagator (Dai et al., 2023) などが先駆けとなり SWIM-IR (Thakur et al., 2024) が多言語化した。Ruri の AutoWikiQA も同系統である。Bekko は合成クエリ生成自体を新提案するのではなく、品質重視と速度重視の相補的な2つの合成器を使い分け、多言語・大規模へ一般化する点に特徴がある（合計約1.6億ペア）：

- 1. 高品質合成 (Qwen3.5-35B-A3B) : fineweb-edu / arXiv abstract / PubMed abstract / CC-News / 英語 Wikipedia の文書に対し、Qwen3.5-35B-A3B で IR クエリを生成する。クエリの形式・観点・具体度（直接的／部分的／タイポ含み／キーワードのみ／曖昧など）や想定質問者を行ごとに制御し、文脈を読んだ多様で現実的なクエリを作る。計約3,900万ペア（英語 Wikipedia 約2,940万、arXiv 約288万、PubMed 約236万 等；生成条件の詳細は付録 B）。
- 2. 高速合成 (query-crafter-multilingual) : 著者開発の多言語クエリ生成モデル (Qwen3-1.7B ベース) で FineWeb2-IR (21 言語) 等を大量合成する。キーワード・自然文クエリ・タイトル・FAQ・要約など7種のクエリタイプを重み付きで生成する。軽量で大量生成できる一方、文脈を深く読む複雑なクエリは作りにくい（付録 D）。計約1.2億ペア。

第二段階データ（公開 hard negatives: bekko-embedding-v1-hard-negatives）

第二段階データは query / positive (1) / hard negatives 形式の高品質データで構成し、独自マイニング分を Hugging Face 上に bekko-embedding-v1-hard-negatives として公開している。

系統	データ	タイプ	概要
日本語	Ruri v3 FT (Tsukagoshi & Sasano, 2026)	query_doc	Ruri の全 FT サブセット（最大512トークン）
BGE サブセット	dureader, hotpotqa, miracl, mr-tydi, squad, mmarco (≦20万行), mmarco 中国語 (≦8万行), pubmed_qa_labeled (query_doc) / ATEC, BQ, LCQMC, PAWSX (doc_doc)	query_doc / doc_doc	BGE-M3 FT データのうちライセンスが比較的明確なサブセット
独自 hard negatives	bekko-embedding-v1-hard-negatives (総 759,587 行・22 サブセット)	query_doc / doc_doc	下記

独自 hard negatives は、英語汎用 IR (wikipedia_hard_negatives_english 50万行, agnews, gooaq, natural_questions, trivia_qa)、doc_doc (all_nli, coco_captions)、ドメイン (arxiv, codesearch, fineweb, pubmed)、長文版 (wikipedia_hard_negatives_long_docs_*, 11言語) から成る。各行は query + positive + 最大15本の hard negatives (FT では最大7本使用)。マイニングは、短～中文では sparse BM25 + dense vector のハイブリッド検索を用い、dense モデルには自モデルの開発過程の先行チェックポイント (256次元 truncate) を使う（自己ブートストラップ）。positive 除外・near-duplicate 除外などの品質フィルタを適用する。長文版（最大8192トークン）は sparse BM25 のみで採掘し、クエリは LLM (DeepSeek-V4, deepseek-v4-flash) で生成する (MLDR (Chen et al., 2024) と同枠組み)。長い文書の一部だけが正解を含むよう文書を分割サンプリングしてハードな負例を構成し、第二段階で8192トークンの長文脈検索能力を得る。

第一段階データの系統別の内訳・実例・傾向（対訳 bitext、doc_doc の broad positive、二系統の合成 query_doc 等）は付録 B に詳述する。第二段階の混合は a8m・a25m で同一であり（37 サブセット・実効 1,780,001 行。BGE-M3 FT 由来の MS MARCO ≦20万行・中国語 mMARCO ≦8万行を含む）、独自 hard negatives は公開 759,587 行から上限付きサンプリングした 482,000 行を用いる（全構成と件数は付録 C）。

3.4 原理③: 学習レシピ（二段階・損失・MRL）

二段階の対照学習を行う。第一段階は、大規模な弱教師ありペアの対照学習が高品質な埋め込みを生むという知見 (Wang et al., 2022 (E5); Neelakantan et al., 2022) に基づき、多言語の弱教師ありペアで汎用表現を獲得する (max_seq 512, バッチ

8192)。第二段階は GTE (Li et al., 2023)・BGE-M3 (Chen et al., 2024)・Ruri (Tsukagoshi & Sasano, 2026) の多段階学習に倣い、hard negatives を含む高品質データで検索性能を高め、長文脈 (max_seq 8192) に適応する。a8m / a25m の確定ハイパーパラメータを示す。

パラメータ	第一段階 a8m	第一段階 a25m	第二段階 a8m	第二段階 a25m
ベースモデル	mmBERT-L4H384-pruned	mmBERT-L13H384-pruned	第一段階モデル	第一段階モデル (チェックポイントマージ; 本文)
最大トークン長	512	512	512 / 8192 (データ群別)	512 / 8192
バッチサイズ	8192	8192	1152 / 192 (データ群別)	1152 / 192
学習率	9.0×10^{-4}	3.0×10^{-4}	1.0×10^{-5}	4.0×10^{-6}
ウォームアップ比率	0.1	0.1	0.3	0.3
エポック数	1	1	1	1
温度 τ / マージン m	0.03 / 0.1	0.03 / 0.1	0.03 / 0.01	0.03 / 0.01
MRL dims / weights	[384,256,128,64] / [1.0,0.3,0.15,0.1]	同左	同左	同左
勾配キャッシュのミニバッチ	2048	512	16-128	16-128

学習率はモデルが小さいほど高く設定する (a8m 第一段階 9.0×10^{-4} > a25m 3.0×10^{-4})。これは開発時の探索に基づく設定である。

第二段階の初期値: a8m は第一段階の最終チェックポイントをそのまま用いる。a25m は、第一段階の最終チェックポイントと、学習中の開発用評価が最良だった中間チェックポイントとを線形マージ (重み 0.5/0.5, bf16) したものを初期値とする——同一 run 内の2チェックポイントの重み平均であり、開発用評価での比較により最良の初期値として採択した。

Prefix: リリースモデルは query / document のいずれにも prefix を付与しない。E5 (Wang et al., 2022)・Ruri などの役割 prefix とは異なる選択であり、同じ API で対称・非対称タスクを扱える単純さを優先した。a8m 第一段階の統制比較では prefix 有無の差が開発用評価で 0.003 以内と小さく、役割 prefix の明確な優位は確認されなかった (§6.2)。

マスク付き対照損失。基盤損失には、Qwen3 Embedding (Zhang et al., 2025) で提唱されたマスク付き対照損失を用いる (後述のペアタイプ別の損失方向の選択と、明示的 hard negatives の扱いが本研究による適用)。バッチ内対照学習では、バッチ内の query-document ペア $\{(q_i, d_i)\}_{i=1}^N$ (N はバッチサイズ) に対し、 q_i の負例として他ペアの文書 $\{d_j\}_{j \neq i}$ (in-batch negatives) を使う。しかし大規模バッチでは、この「負例」の中に実際には q_i の正解となりうる文書 (偽陰性) が混ざる。マスク付き損失は、「正例のスコアをマージン m を超えて上回る負例候補は偽陰性の疑いが強いので、分母から除外する」という規則でこれを抑える。sim をコサイン類似度、 τ を温度として：

$$L_{q \rightarrow d} = -\frac{1}{N} \sum_{i=1}^N \log \frac{\exp(\text{sim}(q_i, d_i)/\tau)}{\exp(\text{sim}(q_i, d_i)/\tau) + \sum_{j \neq i} M_{ij} \exp(\text{sim}(q_i, d_j)/\tau)}$$

マスク $M_{ij} \in \{0, 1\}$ は次で与えられ、除外対象 ($M_{ij} = 0$) の項は実装ではロジットを $-\infty$ ($\exp(-\infty) = 0$) に置くことで分母から消す：

$$M_{ij} = \mathbf{1} [\text{sim}(q_i, d_j) \leq \text{sim}(q_i, d_i) + m]$$

分配関数 (分母) の構成は Qwen3 Embedding の拡張正規化項に従う——GTE (Li et al., 2023) の改良対照損失の系譜で、他ペアの文書 d_j に加えて、他のクエリ q_j ($q_i - q_j$) と文書間 $d_i - d_j$ の類似度も含め、いずれの項にも同じマージン規則を適用する (式は q-d 項のみ示した)。ただし明示的 hard negatives の扱いは、hard negatives にもマスクを適用する Qwen3 Embedding の定義と異なる——各クエリに割り当てられた明示的 hard negatives (triplet の負例列) は検証済みの負例であるため、本研究ではそのクエリの損失で除外規則の対象外とし常に分母に残す (他のクエリに割り当てられた負例が in-batch 候補として現れる場合は通常どおりマスクされうる)。ペアタイプ別に方向を分岐し、対称な doc_doc ペア (対訳・NLI・Wikipedia 等) には q と d の役割を入れ替えた損失も平均する双方向損失 $L_{\text{bidir}} = (L_{q \rightarrow d} + L_{d \rightarrow q})/2$ を、非対称な query_doc ペア (IR 合成・検索) には単方向損失を適用する。温度は $\tau=0.03$ 、マージンは第一段階 $m=0.1$ ・第二段階 $m=0.01$ 。偽陰性の混入は hard negatives を含む第二段階で特に問題になりうるため、マスクの役割は第二段階で大きいと考えられる (段階別のマスク発動率の統計は取得していない)。

大規模バッチと GradCache。大規模バッチの対照学習は in-batch negatives に hard negative が含まれる確率を高め、検索性能を大きく押し上げる (Neelakantan et al., 2022)。単一GPU でこれを実現するため GradCache (Gao et al., 2021) を用いる——埋め込み計算と逆伝播を分離し、バッチを小さなミニバッチ (a8m 第一段階 2048) に分割して勾配をキャッシュしながら、バッチ全体の in-batch negatives を近似なく計算する手法で、Sentence Transformers (Reimers & Gurevych, 2019) ライブラリの Cached 系損失 (CachedMultipleNegativesRankingLoss 系; 実装ドキュメント) として実用的な実装が提供されている。本研究のマスク付き損失もこの Cached 実装と同じ勾配キャッシュ構造で実装し、単一GPU でバッチ 8192 を扱う (バッチは単一サブセットから構成する; §3.3)。バッチサイズは、GTE (Li et al., 2023) が対照事前学習の性能はバッチ約1万で飽和すると報告していることから、8192 を採用した。

MRL。MatryoshkaLoss (Kusupati et al., 2022) で対照損失を複数次元に同時適用し、運用時に次元を切り出せる多粒度埋め込みを学習する (dims [384,256,128,64], weights [1.0,0.3,0.15,0.1])。低次元ほど重みを弱め全次元品質を保つ。MRL は超小型モデルで特に有用で、低リソース環境では出力ベクトル自体を小さくしたいケースが多い (リリースモデルは 256 次元へ truncate しても overall -1.3~1.8%; §7.4)。リリースと同一 dims での重み比較 (a8m 相当構成・20% データの proxy アプレーション; 付録 F.3) では、設定間の差は HAKARI 全体平均で 0.001 程度と小さく、低次元の重みをやや強めた他の2設定がリリース採用値を僅かに上回った (リリース採用値は 3 設定中最下位だが差は僅少) ——重み選択の感度は低い。

総損失には、さらに量子化対応学習 (Quantization-Aware Training; QAT) の補助項を加える。QAT とは、学習中に埋め込みを int8 / ubinary へ疑似的に量子化した上で同じ対照損失を計算し、量子化しても性能が保たれる表現をあらかじめ学習させる仕組みである。重み・活性を量子化する一般的な QAT とは異なり、本稿での適用対象は出力埋め込み空間であることに注意する (モデル重みの量子化は学習では行わず、§7.3 で事後量子化として別に扱う。実装と観察は §6.3)。実装では MRL の各次元の内側で float の損失と量子化埋め込みの損失を重み付きで足す：

$$L = \sum_{k \in \{384, 256, 128, 64\}} w_k^{\text{MRL}} \left(L_{[:,k]}^{\text{mask}} + \sum_{p \in \{\text{int8}, \text{ubinary}\}} w_p^{\text{QAT}} L_{p,[:,k]}^{\text{mask}} \right)$$

ここで $L_{[:,k]}^{\text{mask}}$ は先頭 k 次元、 $L_{p,[:,k]}^{\text{mask}}$ は同じ k 次元に量子化 p を疑似適用した埋め込みで計算した同損失で、 $w_k^{\text{MRL}} = [1.0, 0.3, 0.15, 0.1]$ 、 $w_p^{\text{QAT}} = [0.1, 0.1]$ (QAT 項の実効重みは積 $w_k^{\text{MRL}} \cdot w_p^{\text{QAT}}$ となる)。QAT 項は補助的で、その観察と解釈は §6.3 で述べる。

3.5 単一GPU 学習とトークン長グループ

全学習を単一GPU (NVIDIA RTX PRO 6000 Blackwell Max-Q, GPU メモリ 96GB) で実施した (a8m 約3日、a25m 約8日)。これを可能にするのは、AP の小型化そのものに加え、GradCache による大規模バッチのメモリ効率化 (§3.4)、トークン長グループ別のバッチ設定によるパディング削減、ModernBERT の flash attention の組み合わせである。第一段階は最大 512 トークン・固定バッチ 8192 で学習する。第二段階は、BGE-M3 (Chen et al., 2024) のトークン長グループ学習に倣い、データをトークン長で2群に分け、群ごとにバッチサイズを設定する——短〜中文データ (最大 512 トークン) はバッチ 1152、長文データ (最大 8192 トークン) はバッチ 192 である。第二段階の各訓練例 (行) は query 1 件・positive 1 件・hard negatives 最大 7 件から成る。大規模埋め込みモデルの学習には数十GPU規模の分散環境が用いられることが多い——例えば BGE-M3 は最大 96 GPU (A800)、mGTE は 32 GPU (A100)、E5 は最大 64 GPU (V100) を報告する (Chen et al., 2024; Zhang et al., 2024; Wang et al., 2022)——のに対し、Bekko は一般的なワークステーションに載る単一GPU で完了する。コンシューマ向け単一GPU (RTX 5090) でも、勾配キャッシュのミニバッチサイズをより小さくすることで必要な GPU メモリ量を抑えられ、所要時間はやや増えるが同じ学習が可能な見込みである (RTX 5090 での完走実測は行っていない)。

4. 評価設定

4.1 評価ベンチマーク

主評価 — 著者が作成したデータセットに依存しない2系統を主評価とする。(1) MMTEB Multilingual v2 (Enevoldsen et al., 2025): 131 タスク。a8m・a25m の公式評価 (§5.1) の一次ソースで、Retrieval の nDCG@10 を主指標とする。(2) Multilingual NanoBEIR (14言語): 英語オリジナルの NanoBEIR (Zeta Alpha 作成; BEIR (Thakur et al., 2021) の縮約) にコミュニティ翻訳版 13 言語を加えたもの (§2.4; データセットはいずれも著作権ではない)。計測は HAKARI-Bench ハーネスの統一条件で行う (§5.2)。なお NanoBEIR 系は各タスク 50 クエリ規模の軽量近似評価であり、フル規模の評価を置き換えるものではない——このため公式 MMTEB を主評価①として必ず併置する。

補助比較・proxy — HAKARI-Bench (Tateno, 2026): MTEB/MMTEB/BEIR を Nano 化した軽量多言語 IR ベンチ。NanoMMTEB-v2・NanoRTEB・NanoMLDR (長文書)・NanoLongEmbed (長文)・NanoCoIR (コード)・さまざまな言

語別の MTEB (例: 日本語の NanoJMTEB) 等を、統一条件で測る。リリースモデルの補助比較 (§5.2, 効率フロンティアの可視化) と、開発時の proxy 評価・設計アブレーション (§6) に用いる。なお HAKARI の dense 評価は、各タスクで cosine と dot の両スコアリングを試し高い方を採用する仕様 (タスク別 best-of-similarity の上界; HAKARI 論文に明記) である——Bekko 自身のこの選択による上振れは固定 cosine 比 +0.0002 未満と極小だが、本稿の HAKARI 系の数値・順位はこのプロトコルの下での値である。ただし HAKARI-Bench は本研究の著者が公開したベンチマーク (self-citation) であり、独立性を担保するため、中心的な検索性能の主張は独立した公式 MMTEB Multilingual v2 (§5.1) で必ず併せて確認する。

推論効率・量子化の評価 — モデル重みの量子化 (語彙のみ int8 / 全重み int8; §7.3) の精度保持は NanoBEIR-en + NanoMMTEB-v2 を基本に、入力長依存の影響を NanoLongEmbed・NanoMLDR・NanoCoIR で確認し、CPU / Raspberry Pi / Apple Silicon / CUDA GPU での推論速度は Natural Questions の文書エンコードで計測し (§7.2)、ブラウザ (WebGPU / WASM) でも同じ Natural Questions の文書エンコードで実測する (§7.5)。

4.2 比較モデル

§1.2 の通り、推論計算量を支配するのは AP (非埋め込みパラメータ) である。比較モデルは、「商用利用可能なライセンスで重みが公開されている、多言語対応の dense 埋め込みモデルで AP 350M 以下」+ 語彙ベースラインの BM25 という基準で統一する。この基準により、非商用ライセンスのモデル (例: jina-embeddings-v3)、AP が基準を超える decoder 系 LLM 転用の埋め込みモデル (例: harrier-oss-v1-0.6b, AP 440M)、文脈を持たない静的埋め込み (§2.3) は統一集合に含めない。なお harrier-oss-v1-270m (Microsoft, 2026) は decoder-only アーキテクチャ (last-token pooling) の埋め込みモデルだが、商用可・重み公開・多言語 dense・AP ≤ 350M の基準をすべて満たすため統一集合に含める。AP 300M 級のモデル (mE5-large・bge-m3・arctic-embed-l-v2.0 (Yu et al., 2024)) は、AP が 1~2 桁小さい Bekko がどこまで迫れるかを測る上限側の比較対象である。BM25 は MMTEB にスコアが存在しないため、HAKARI 系の比較 (§5.2–5.3) でのみ用いる。BM25 のスコアは、各 Nano セットの構築時に保存された BM25 候補ランキングを評価したものである——候補構築パイプラインは bm25s ライブラリの Okapi BM25 に基づき、トークナイズは言語別に切り替える (日本語・中国語・タイ語・韓国語・ベトナム語は単語分割器、ステマーが利用可能な言語はステマー、それ以外は正規表現)。以下、multilingual-e5 系は mE5-small のように、gte-multilingual-base は gte-m-base と略記する。

モデル	AP	総パラメータ 次元		最大入力長
BM25 (lexical; §5.2–5.3 のみ)	—	—	—	—
bekko-embedding-v1-a8m	7.67M	106M	384	8192
mE5-small	21.6M	118M	384	512
bekko-embedding-v1-a25m	24.93M	123M	384	8192
granite-97m-r2	28.3M	97M	384	32768
mE5-base	86.0M	278M	768	512
harrier-oss-v1-270m	100.3M	268M	640	32768
embeddinggemma-300m	106.3M	308M	768	2048
granite-311m-r2	110.3M	312M	768	32768
gte-m-base	113.3M	305M	768	8192
mE5-large	303.9M	560M	1024	512
arctic-embed-l-v2.0	311.8M	568M	1024	8192
bge-m3	311.8M	568M	1024	8192

AP の定義・集計は、MMTEB の `n_active_parameters_override` があればそれを、なければ `n_parameters - n_embedding_parameters` を用いる。mE5-small が総 118M でも AP 21.6M に過ぎないことは、この軸の重要性を端的に示す。

比較の公平性: 公式 MMTEB (§5.1) では、競合モデルは全て同一スナップショット (2026-06-28) で 131 タスクを完走しており、各モデルは MTEB の `ModelMeta` に登録されたそのモデル自身の推奨 `prompt・dtype` で評価される。競合モデルのスコアは公式 MMTEB cache 由来であり、Bekko はリリースモデル (prefix なし; §3.4) を同一のタスク集合 (131 タスク) ・同一の集計規則で評価した。AP だけに依存しないよう、総パラメータ・配布サイズ・スループット (§7.2) も併記する。

5. 結果: 小さな Active Parameters での検索品質

検索品質は、公式 MMTEB Multilingual v2 (§5.1) と、第三者作成データセットからなる Multilingual NanoBEIR (14言語, §5.2) を主評価とし、著者が公開した軽量ベンチ HAKARI-Bench (§5.2) を効率フロンティアの可視化と多角的確認のための

補助比較として用いる。なお a8m と a25m の第二段階データは同一だが (§3.3)、層数のほかに学習率・勾配キャッシュ設定・a25m のチェックポイントマージ (§3.4) も異なるため、本稿の a8m/a25m 間の比較はリリースされた2構成の比較であり、AP (容量) だけを変えた統制されたスケーリング比較ではない点に注意する。

5.1 公式 MMTEB Multilingual v2 による評価 (主評価)

a8m・a25m を公式 MMTEB Multilingual v2 (131 タスク) で評価した。競合モデルは公式 2026-06-28 スナップショット (mteb 2.16.1, 全 131 タスクを完走した 170 モデル) の値、Bekko はリリースモデルを同一のタスク集合・同一の集計規則 (タスクスコア = 全 split/subset の平均、タイプ別 = タイプ内タスク平均) で評価した値である (§4.2)。表2 に統一比較集合のタスクタイプ別スコアを示す。本研究の中心は検索であるため Retrieval (表では Ret) を最重要指標とし、総合のタスク平均 (Mean) と他の全タスクタイプも併記する。参考として、第一段階 (大規模対照学習) 終了時点の pretrain model (a8m-pt / a25m-pt) の値も併記する。

表2: 公式 MMTEB Multilingual v2 のタスクタイプ別スコア (すべて ×100; AP 昇順)。Mean = 131 タスクの平均 (Mean(Task))、Ret = Retrieval (task-macro, nDCG@10)、Rerank = Reranking、Bitext = BitextMining、STS = Semantic Textual Similarity、PairCls = PairClassification、Class = Classification、Clust = Clustering、MultiLbl = MultilabelClassification。InstructionReranking は全モデルで 0 付近 (中心化スコア) のため省略。-pt 行は第一段階終了時点の参考値。太字は Bekko の強みを示すスコアで、列内最良の意味ではない (根拠は下の要点と本文を参照)。

モデル	AP	dims	Mean	Ret	Rerank	Bitext	STS	PairCls	Class	Clust	MultiLbl
bekko-a8m-pt	7.7M	384	55.0	55.1	42.3	72.1	68.6	75.6	54.4	41.4	18.1
bekko-a8m	7.7M	384	56.7	56.2	60.6	73.1	71.6	76.1	54.8	42.4	18.6
mE5-small	21.6M	384	56.4	50.9	60.4	69.4	71.7	77.3	57.0	41.5	19.3
bekko-a25m-pt	24.9M	384	56.7	56.1	44.4	74.1	70.6	76.7	56.9	42.5	18.6
bekko-a25m	24.9M	384	58.3	57.5	61.6	75.4	73.4	77.4	57.2	43.0	19.2
granite-97m-r2	28.3M	384	51.9	60.3	59.4	44.2	65.6	74.5	49.6	43.6	17.9
mE5-base	86.0M	768	57.0	52.7	60.2	69.5	71.4	77.2	58.2	41.8	20.2
harrier-270m	100.3M	640	66.6	66.4	61.9	81.5	75.4	80.1	70.8	52.5	24.0
embgemma-300m	106.3M	768	61.2	62.5	63.3	64.4	74.7	81.4	60.9	51.2	24.8
granite-311m-r2	110.3M	768	56.0	65.2	62.0	57.9	69.0	76.1	53.3	44.4	18.5
gte-m-base	113.3M	768	58.3	57.2	60.7	71.8	72.9	80.5	57.2	44.2	19.8
mE5-large	303.9M	1024	58.6	53.7	62.9	73.8	73.3	78.8	59.4	42.7	21.1
arctic-l-v2.0	311.8M	1024	57.0	58.4	63.7	64.1	70.1	76.8	57.4	42.8	18.9
bge-m3	311.8M	1024	59.6	54.6	62.8	79.1	74.1	80.8	60.3	40.9	20.1

- 要点は次の通り。
- Retrieval (task-macro) : AP が1~2桁大きいモデルに匹敵・凌駕する。a8m 56.2 (AP 7.67M) が、AP 3~40倍の mE5-small (50.9)・mE5-base (52.7)・mE5-large (53.7)・bge-m3 (54.6) を上回る。a25m 57.5 はさらに gte-m-base (57.2) と同水準で、arctic-embed-l-v2.0 (58.4, AP 12倍) に 1 ポイント未満の差まで迫る。統一集合で Bekko より上位に立つのは、Retrieval 特化の傾向が強い granite R2 系 (97m 60.3 / 311m 65.2)・embeddinggemma-300m (62.5)・harrier-270m (66.4) と arctic である。Bekko の強みは Retrieval に集中する。
 - 総合 (Mean) : a25m (58.3) は gte-m-base (58.3) と同水準で、AP 12倍の arctic-l-v2.0 (57.0)・mE5-base (57.0)・granite 系を上回り、mE5-large (58.6) に 0.3 差まで迫るが、bge-m3 (59.6)・embeddinggemma (61.2)・harrier-270m (66.6) には及ばない。a8m (56.7) は mE5-small (56.4) を上回る。したがって「a25m は bge-m3 と同等以上」という関係は MMTEB Retrieval・MNanoBEIR・HAKARI Overall では成立する (後者2つは 0.01 差以内) が、MMTEB 総合では成立しない。
 - タスクタイプ別の傾向 (表2) : 検索系 (Ret・Rerank) と言語横断対訳 (Bitext)・STS に強く、PairCls・Clust・MultiLbl は比較群の中位、Class はやや下位である。
 - 言語横断整合の裏付け (Bitext) : BitextMining (task-macro) で a25m 75.4 と、統一集合では harrier-270m (81.5)・bge-m3 (79.1) に次ぐ 3 位に位置し、mE5-large (73.8)・gte-m-base (71.8)・arctic (64.1)・granite R2 系 (311m 57.9 / 97m 44.2) を上回る。a8m も 73.1 と AP 40 倍の mE5-large に 0.7 差の水準にある。これは第一段階データの対訳 bitext (1,600 超の言語ペア・約3.3億行; §3.3) と整合する強みである。なお 100 言語超対応の主根拠はデータの言語カバ

レッジ (§3.3) であり、BitextMining が裏付けるのは同ベンチマークに含まれる言語ペアの範囲での言語横断整合性である。

なお本稿の中心的主張範囲は一次検索 (dense retrieval) としての品質であり、Clustering 等を含む汎用文埋め込み全体での最先端を主張するものではない。

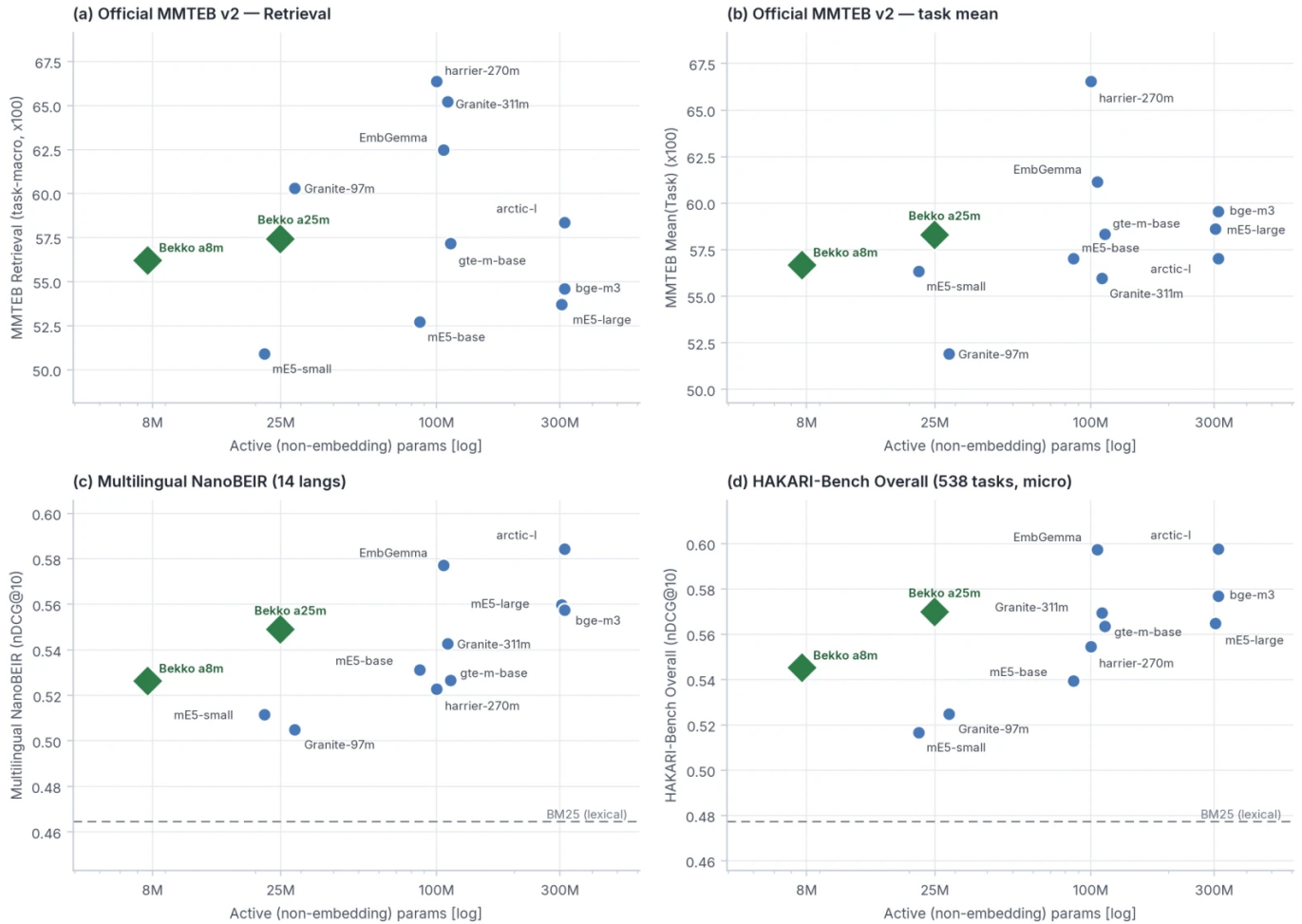


図4: 検索品質対 AP——AP 軸 (§1.2) の中心的実証。(a) MMTEB Retrieval では a8m (AP 7.7M) が bge-m3・mE5 系を上回り、a25m は AP 12 倍の arctic に迫る。(b) MMTEB 総合 (Mean(Task)) でも a25m は arctic を上回り、gte-m-base と同水準である (表2)。(c) Multilingual NanoBEIR (§5.2) でも a25m は 300M 級の bge-m3・mE5-large と 0.02 以内に並び、全 dense モデルが BM25 (破線) を上回る。(d) HAKARI-Bench Overall でも概ね同じ構図である。

言語別の傾向は、主評価②である Multilingual NanoBEIR の言語別スコアで確認する (§5.2, 付録 G.1, 図9)。a25m は 14 言語すべてで mE5-small と BM25 を上回り、英語・ポルトガル語・アラビア語では bge-m3 をも上回る一方、他の言語では bge-m3・mE5-large を僅差 (bge-m3 との差は最大 0.026) で下回る。すなわち AP 約12倍の 300M 級に全言語で並走し言語間の落ち込みが小さいのが特徴で、タイ語・アラビア語・セルビア語などスコア下位の言語に伸び代を残す。言語別・長文タスクの全比較表は付録 G に示す。

5.2 Multilingual NanoBEIR (主評価②) と HAKARI-Bench による補助比較

もう一つの主評価である Multilingual NanoBEIR (14言語; データセットは著者作ではない, §2.4) と、著者公開の軽量ベンチ HAKARI-Bench (§4.1; self-citation) の各タスクセットで、統一比較集合 (§4.2; BM25 を含む 13 モデル) を比較する。数値はいずれも HAKARI-Bench の公開リーダーボード集計からの切り出しである (nDCG@10, 0-1)。-pt の2行のみ公開リーダーボードに含まれない参考値で、同一プロトコルのローカル評価による。表3 の各列は次を測る: Overall = HAKARI-Bench 公開リーダーボードの総合値 (micro: 重複除去後の全 538 raw タスクの nDCG@10 単純平均。MNanoBEIR の 182 セルを個別に含むため同ベンチの重みが約 34% を占める。ベンチマーク単位で均す macro 集計では僅差の順位が入れ替わる)、MNanoBEIR = BEIR 縮約版の14言語セット (多言語の一般 Web 検索)、NanoMMTEB-v2 = MMTEB Multilingual v2 の Retrieval タスクのみを sampling した subset (18タスク)、NanoRTEB = 実運用ドメイン寄りの検索ベンチ RTEB (Liu et

al., 2025) の公開英語部の Nano 化 (14タスク)、NanoCoIR = コード検索ベンチ CoIR (Li et al., 2024) の Nano 化 (10タスク)。

表3: Multilingual NanoBEIR (主評価②) と HAKARI-Bench の各タスクセット (nDCG@10; AP 昇順)。MNanoBEIR 列が主評価②であり、他の列 (Overall・NanoMMTEB-v2・NanoRTEB・NanoCoIR) は著者公開の HAKARI-Bench による補助比較である。-pt 行は第一段階 (大規模対照学習) 終了時点の pretrain model の参考値。視認性のため bekko-a8m・bekko-a25m のスコアを太字で示す (列内最良の意味ではない)。長文系タスクセットは §5.3 (図5) と付録 G.2 に示す。

モデル	AP	dims	Overall	MNanoBEIR	NanoMMTEB-v2	NanoRTEB	NanoCoIR
BM25	—	—	0.477	0.465	0.455	0.355	0.544
bekko-a8m-pt	7.7M	384	0.523	0.505	0.494	0.527	0.727
bekko-a8m	7.7M	384	0.545	0.526	0.502	0.550	0.747
mE5-small	21.6M	384	0.517	0.512	0.445	0.471	0.692
bekko-a25m-pt	24.9M	384	0.549	0.533	0.488	0.571	0.769
bekko-a25m	24.9M	384	0.570	0.549	0.494	0.594	0.786
granite-97m-r2	28.3M	384	0.525	0.505	0.531	0.567	0.780
mE5-base	86.0M	768	0.539	0.531	0.469	0.514	0.700
harrier-270m	100.3M	640	0.555	0.523	0.522	0.550	0.789
embgemma-300m	106.3M	768	0.597	0.577	0.517	0.670	0.847
granite-311m-r2	110.3M	768	0.569	0.543	0.577	0.606	0.814
gte-m-base	113.3M	768	0.563	0.527	0.486	0.558	0.753
mE5-large	303.9M	1024	0.565	0.560	0.484	0.556	0.747
arctic-l-v2.0	311.8M	1024	0.598	0.584	0.502	0.549	0.707
bge-m3	311.8M	1024	0.577	0.557	0.485	0.536	0.692

要点は次の通り。

- Multilingual NanoBEIR (主評価②) : 最上位は arctic-l-v2.0 (0.584) と embeddinggemma (0.577) である。a25m (0.549) は AP が約12倍の bge-m3 (0.557)・mE5-large (0.560) と 0.02 以内で、granite-311m (0.543)・mE5-base (0.531)・gte-m-base (0.527)・harrier-270m (0.523) を上回る。a8m (0.526) は AP 7.7M で mE5-small (0.512)・granite-97m (0.505) を上回り、gte-m-base・harrier-270m と同水準である。BM25 (0.465) は全 dense モデルが上回る。
- HAKARI Overall (538 raw タスク micro) : arctic (0.598) $\approx$ embeddinggemma (0.597) > bge-m3 (0.577) > a25m (0.570) $\approx$ granite-311m (0.569) の順で、a25m は bge-m3・granite-311m (AP 4~12 倍) と 0.01 差以内に位置する。a8m 0.545 は mE5-small (0.517)・granite-97m (0.525)・mE5-base (0.539) を上回り、harrier-270m (0.555) に 0.01 差である。なおベンチマーク単位で均す macro 集計では gte-m-base $\approx$ bge-m3 $\approx$ a25m となるなど僅差の順位は集計方式に依存する。
- MMTEB 系検索 (NanoMMTEB-v2)・実運用寄り検索 (NanoRTEB) : NanoMMTEB-v2 では granite R2 系 (311m 0.577 / 97m 0.531) が強く、a8m (0.502) は arctic (0.502) と並び、a25m (0.494) も mE5 系 (0.445–0.484) を僅かに上回る。なお単純平均では a8m が a25m を上回るが、これは a25m が一部の外れタスク (8K 文脈の合言葉検索など 3 タスク) で大きく低いことによるもので、18 タスク中 14 タスクでは a25m が a8m を上回り、公開リーダーボードのタスク順位集計 (Borda) でも a25m が上位である。NanoRTEB では embeddinggemma (0.670)・granite-311m (0.606) に次いで a25m (0.594) が 3 位に入り、a8m (0.550) も mE5-large (0.556) に迫る。
- コード検索 (NanoCoIR) : embeddinggemma (0.847)・granite-311m (0.814)・harrier-270m (0.789) に次いで a25m (0.786) が granite-97m (0.780) と並ぶ水準であり、a8m (0.747) は mE5-large (0.747) と同等である。

第一段階終了時点の pretrain model (表3 の -pt 行) を同条件で測ると、a8m-pt $\rightarrow$ a8m は Overall 0.523 $\rightarrow$ 0.545、MNanoBEIR 0.505 $\rightarrow$ 0.526、NanoMLDR 0.481 $\rightarrow$ 0.545、NanoLongEmbed 0.656 $\rightarrow$ 0.682 へ上昇した。a25m-pt $\rightarrow$ a25m も順に 0.549 $\rightarrow$ 0.570、0.533 $\rightarrow$ 0.549、0.530 $\rightarrow$ 0.571、0.675 $\rightarrow$ 0.706 である。第二段階は全体を約0.02押し上げ、特に長文書内検索の改善が大きい。MMTEB でも Mean(Task) が a8m 55.0 $\rightarrow$ 56.7、a25m 56.7 $\rightarrow$ 58.3 へ上昇し、とりわけ Reranking (42.3 $\rightarrow$ 60.6 / 44.4 $\rightarrow$ 61.6) の改善が大きい (表2)。ただし設計要因を切り分けたアブレーションではないため、長文 hard negatives だけの因果効果とはみなさない。

最も近い競合である granite-97m-r2 (AP 28.3M; §2.3) との直接比較をまとめる。granite-97m が a25m を上回るのは MMTEB Retrieval (60.3 vs 57.5) ・ NanoMMTEB-v2 (0.531 vs 0.494) ・ MMTEB Clustering (43.6 vs 43.0) である。一方 a25m は、MMTEB 総合 Mean (58.3 vs 51.9) ・ BitextMining (75.4 vs 44.2) ・ STS (73.4 vs 65.6) ・ MNanoBEIR (0.549 vs 0.505) ・ HAKARI Overall (0.570 vs 0.525) ・ 長文2ベンチ (§5.3) で上回り、さらに小さい a8m (AP 7.7M) も MNanoBEIR (0.526) と Overall (0.545) で granite-97m を上回る (表2・表3)。すなわち granite-97m は MMTEB 系の検索に強く特化しているのに対し、Bekko は多言語 Web 検索・長文・コード・言語横断を含む検索の広い側面と総合のバランスで優位であり、加えて蒸留不要・学習データ公開・CPU 速度 (§7.2: a8m 364 vs granite-97m 125 docs/s) という設計面の差がある。

本節の HAKARI 系の数値は著者公開のベンチマークに基づくため、解釈には常に独立評価 (§5.1) を併置する。両者を突き合わせると、検索系タスクでは AP の大きなモデルに匹敵・凌駕する一方、MMTEB 総合では中位であり、HAKARI Overall が示す姿よりも MMTEB 総合の評価は厳しい。この差は §5.1 に明記した通りである。MNanoBEIR・HAKARI Overall 対 AP の効率フロンティアは図4(c)(d) に示した。

5.3 長文検索・長文書内検索: 結果と考察

長い入力を扱う 2 つのタスクセットで統一集合を比較する。NanoLongEmbed (6タスク; LongEmbed (Zhu et al., 2024) の Nano 化。needle-in-haystack 系の合成タスクと要約・QA 系を含む) は長文コーパスからの検索を、NanoMLDR (13言語; MLDR の Nano 化) は長文書の中から目的の箇所を含む文書を当てる多言語検索を測る (per-task / 言語別は付録 G.2)。

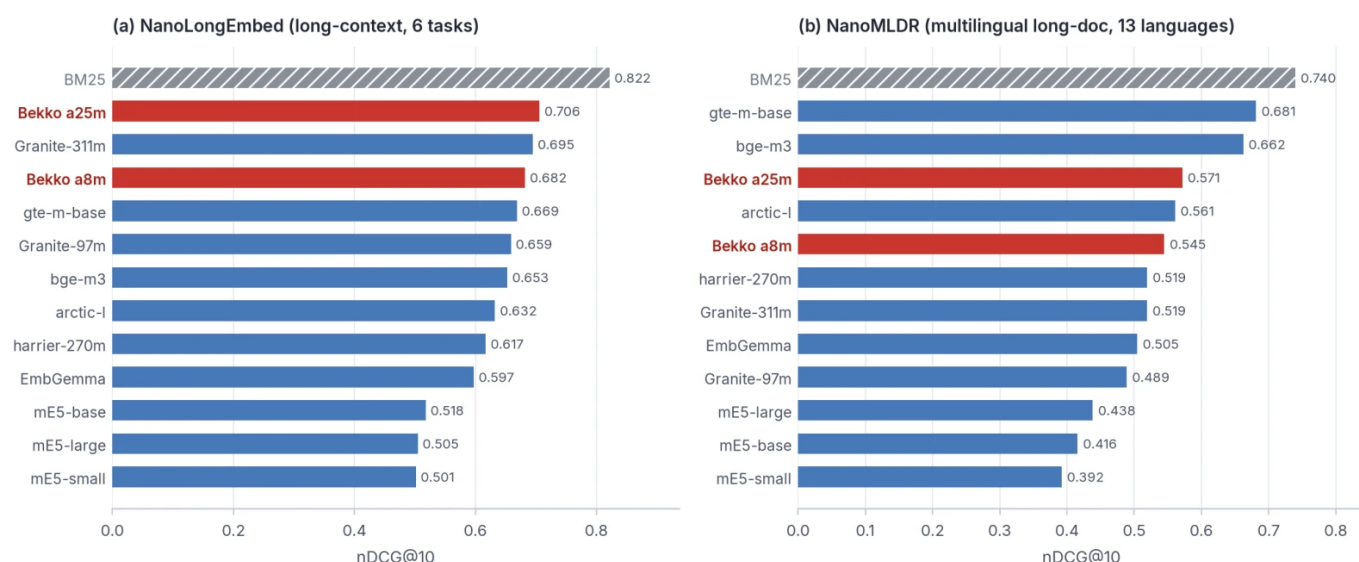


図5: 長文検索 (NanoLongEmbed) と長文書内検索 (NanoMLDR)。BM25 (斜線) が両ベンチで全 dense モデルを上回り、dense 内では a25m が NanoLongEmbed 1 位・NanoMLDR 3 位に位置する。per-task・言語別の全数値は付録 G.2 に示す。

結果は次の通り (図5)。

- BM25: 短文中心の MNanoBEIR では全 dense モデルが BM25 を上回った (§5.2) のとは対照的に、長文の両ベンチでは BM25 が全 dense モデルを上回る (NanoLongEmbed 0.822 / NanoMLDR 0.740)。長い文書の中の一節を当てる課題では、クエリと該当箇所の語彙一致が強い手がかりとなるため、語彙ベースの BM25 が構造的に有利である。BM25 が MLDR で dense 系ベースラインを大きく上回することは BGE-M3 の原論文でも報告されており (同論文 Table 3)、dense モデルの長文書検索が未解決の課題であることを示す。実運用では BM25 と dense のハイブリッド検索が有効な領域であり、Bekko はその dense 成分として低コストに組み込める。
- NanoLongEmbed: a25m 0.706 が dense モデル中 1 位 (a8m 0.682 も granite-311m 0.695 に次ぐ 3 位)。per-task では、a25m は Nano2WikiMultihopQA 0.873、NanoNarrativeQA 0.480、NanoNeedle 0.688、NanoPasskey 0.752、NanoQMSum 0.474、NanoSummScreenFD 0.969 である (付録 G.2)。なお mE5 系は最大入力長が 512 トークンであり、長文タスク全般で大きく低い (mE5-large でも平均 0.505)。
- NanoMLDR: gte-m-base (0.681) ・ bge-m3 (0.662) に次いで a25m (0.571) が dense 3 位。ただし MLDR は bge-m3 の論文で構築されたベンチマークであり、同モデルの fine-tuning には長文書データが含まれる (Chen et al., 2024)。Bekko は MLDR 系の学習データを意図的に避けており (独自の長文 hard negatives は MLDR の構築手法のみを借用; §3.3)、そ

の条件で arctic (0.561) ・ harrier-270m (0.519) ・ granite R2 系 (97m 0.489 / 311m 0.519) ・ embeddinggemma (0.505) ・ mE5 系 (0.392-0.438) を上回る。a8m (0.545) も AP 40 倍の mE5-large (0.438) を +0.11 上回る。

考察・仮説: Bekko の長文耐性には、(1) 基盤 mmBERT が Global / Local いずれの RoPE theta も 160,000 と長文脈向けに設定していること (§3.1)、(2) 第二段階で最大 8192 トークンの長文 hard negatives (11言語) に適応していること、の両方が寄与していると考えられる。ただしこれらの要因 (RoPE 設定・長文データ・タスク構成・最大長) は本稿では切り分けられておらず、寄与の分離は今後の課題である (§8)。

6. 分析: 保持レイヤーの選択と学習レシピの検証

本章では設計選択の根拠を述べる。§6.1 では、リリースモデルの基盤となる保持レイヤーの選択 (a8m = L4 [0,1,2,18]、a25m = L13 [0-11,18]) の根拠となった候補パターンの比較を示す。§6.2 では学習レシピの主要素を、リリースモデルの学習と同一条件の統制比較で検証し、§6.3 では量子化対応学習 (QAT) の有無の比較を述べる。

6.1 どの層を残すか — プルーニングパターンの比較

どの層を残すかで検索性能は大きく変わるため、リリースモデルの保持パターンは候補間の簡易な比較実験で選定した。mmBERT-small の22層から候補となる保持パターンを構成し (L4 5種・L7 11種・L13 10種; 全候補と全数値は付録 E)、全候補に共通の probe——プルーニング後に MS MARCO で検索向けファインチューニングし、NanoBEIR-en (13タスク) の nDCG@10 平均で比較——を適用した (公開 pruned モデルのモデルカードに掲載の実測値)。なお比較対象には、リリースモデルには使っていない中間サイズの L7 (7層) も含む。主な結果を次に示す。

深さ	保持レイヤー	NanoBEIR-en mean	備考
22層 (プルーニングなし)	全層	0.5151	ベースライン
L13	[0-11, 18]	0.4964	採択 (a25m 基盤)。22層の 96.4% を維持
L13	[0-11, 21]	0.4800	末尾を最終層に変更
L13	[0-12] (連続)	0.4553	深層 Global なし
L7	[0-5, 18]	0.4693	採択 (開発時の中間サイズ構成)
L7	[0-5, 21]	0.4629	末尾を最終層に変更
L7	[0-6] (連続)	0.4291	深層 Global なし
L4	[0, 1, 2, 18]	0.4530	採択 (a8m 基盤)
L4	[0, 1, 2, 21]	0.4558	末尾を最終層に変更
L4	[0, 1, 2, 3] (連続)	0.3329	深層 Global なし
L4	[18, 19, 20, 21] (深層のみ)	0.4130	初期層なし

観察は3点にまとまる。第一に、各深さの上位候補はいずれも「初期レイヤー群+深層 Global レイヤー」を保持する構成であり、初期層を捨てた構成 (L4 [18-21] 0.4130、L13 [9-21] 0.4307) は大きく劣化する。採択した front-heavy 構成 (初期の連続レイヤー+深層 Global 1層) はこの上位群に属する単純な近最良解で、厳密な最良は L13 (13C) のみである——L4 では 4C が僅差 (+0.003)、L7 では非連続の 7H が僅差で上回る (付録 E)。第二に、深層 Global を全く含まない連続構成は L7 で -0.04、L4 で -0.12 と大きく劣る——本 probe の候補集合内では、長距離依存を担う深層 Global 層を1層残す構成が一貫して高スコアである。第三に、末尾に残す Global 層は最終層 (21) より遠めの深層 (18) が良く、その差は層数が多いほど明瞭である (L13 +0.016、L7 +0.006)。ただし最小構成の L4 では差は誤差程度で逆転しており (-0.003)、「18 が 21 より常に良い」とまでは言えない。a8m の基盤に [0,1,2,18] を採択したのは、L4 単体では [0,1,2,21] が僅かに上回るものの差が誤差程度であり、L7・L13 で一貫して優位な G18 構成との設計の統一性を優先したためである (付録 E)。

参考として、採択パターンのプルーニング直後 (追加学習前) の NanoMIRACL / NanoFineWeb2IR (開発用の非公開データセット) スコアは、L4 [0,1,2,18] で 0.318/0.431、L7 [0,1,2,3,4,5,18] で 0.395/0.531、L13 [0-11,18] で 0.498/0.667 と、層数とともに単調に高い。プルーニングしただけの状態でも検索に有効な信号が残ることを示唆している。

最終レイヤー (21) は MLM 事前学習の出力に特化しており、検索表現としての汎用性が低い可能性がある。したがって遠方の深層 Global 層 (18) を残すことは有効な設計ヒューリスティックだと考えるが、検索知識がそこに因果的に局在すると主張するものではない。また本 probe は英語 (MS MARCO / NanoBEIR-en) に基づく層選択であり、リリースの多言語二段階学習とは条件が異なる。この観察は「深層の多くは冗長である」という ShortGPT (Men et al., 2024) ・ Gromov et al. (2024) の知見 (§2.3) とも整合する。

6.2 学習レシピの検証 — リリース系譜での統制比較

学習レシピの主要素のうち prefix と QAT は、第一段階の全データを用いたリリース系譜の統制比較で確認した（MRL 重みのみ 20% データの proxy 比較; 付録 F.3）。prefix 比較はリリースで採用した「なし」を含む3条件、QAT 比較は prefix（query: / passage:）を付与する構成の中で QAT 以外を揃えた比較であり、後者は prefix 以外のデータ・MRL 設定をリリースモデルと共有する。指標は学習中に定期実行される開発用評価（複数の Nano IR 評価の複合スコア, 0-1）で、1 エポック終端の値（final）と学習過程全体での最大値（best）を読む。各設定 1 run であり、シードを変えた再現は行っていない（一次データは付録 F）。

- Prefix: a8m の第一段階について、prefix を「query + passage (final 0.578 / best 0.581)」「なし (リリースの設定; 0.577 / 0.582)」「query のみ (0.575 / 0.583)」に変えて比較した（付録 F.1）。差はいずれも 0.003 程度以内と小さい。第二段階まで含めた MMTEB Multilingual v2 の統制比較（同一の第二段階データで prefix のみ変更）でも Mean(Task) の差は 0.003 未満（prefix なしが僅かに上）で、prefix 有無で結果に大幅な違いは見られない。参考として、Clustering など対称タスクでは prefix なしの方が高い傾向がある（付録 F.1）。性能がほぼ同等である以上、リリースモデルは query/document の区別を呼出側へ要求しない単純で使いやすい prefix なしを採択した（§3.4）。
- MRL 重み: リリースと同一の dims での重み比較（a8m 相当構成・20% データの proxy; 付録 F.3）でも設定間の差は 0.001-0.007 と小さく、感度は低い。
- QAT: 有無の統制比較は §6.3 で述べる（レシピ要素の中で最も差が大きい）。
- 温度・学習率: $\tau=0.03$ 、および小型モデルほど高い学習率（a8m 第一段階 $9.0 \times 10^{-4} > \text{a25m } 3.0 \times 10^{-4}$ ）は開発初期の探索で選定した確定値である（§3.4 の表）。この探索はリリースモデルと構成の異なる開発初期の設定で行ったため、本稿では統制比較を提示しない。

6.3 量子化対応学習 (QAT) の有無の比較

学習段階から量子化耐性を持たせる狙いで、float32 に加えて int8・ubinary に量子化した埋め込みにも対照損失を適用する補助損失（以下 QAT 損失）を導入した（重み [float32, int8, ubinary] = [1.0, 0.1, 0.1]）。実装は Jacob et al. (2018) の simulated quantization を文埋め込み空間に適用したもので、次元ごとに値域を 256 階調へ割り当てる非対称量子化を straight-through estimator で通す（ubinary はゼロを閾値とする {0,1} の二値化）。

QAT の有無は、prefix（query: / passage:）を付与する構成の中で QAT 以外を揃えた統制比較（prefix 以外のデータ・MRL 設定はリリースモデルと同一）で確認した（§6.2 と同じ開発用評価; 各設定 1 run。一次データは付録 F.2）。a8m (L4) の第一段階では QAT あり (final 0.578 / best 0.581) が QAT なし (0.560 / 0.561) を明確に上回る。a25m (L13) の第一段階ではほぼ同等 (0.590 / 0.594 vs 0.587 / 0.596) である。さらに L13 では、QAT あり／なしのそれぞれを同一の FT データ（開発時の FT 構成; 付録 F.2）で第二段階まで進めた比較でも、QAT あり (final 0.604) が QAT なし (0.595) をやや上回る。すなわち QAT の併用は、報告した final 値では性能を損なわず（L13 第一段階の best 値のみ QAT なしが僅かに上回り、読み方によって混在する）、最小構成の L4 では開発用評価がむしろ明確に高い——int8/ubinary 量子化に伴う正規化が正則化として作用した可能性があるが、各設定 1 run の比較であるため断定はしない。

一方、量子化耐性そのものへの寄与は切り分けられていない。リリースモデルは int8/binary 量子化 (rescore 併用) でほぼ無損失だが（§7.4）、それが QAT の寄与か、埋め込みが元々情報密度の高い分布を持つためかは判別できない。また、学習時の疑似量子化が実運用の検索エンジンにおける量子化スコアリングと一致する保証はないため、量子化耐性は出力ベクトルの実測（§7.4）で確認している。

7. モデル配布サイズの小型化・推論速度

本章では、§1.2 で述べた AP 軸の帰結を実機で示す——モデル配布サイズの小型化（§7.1）と推論速度（§7.2）である。Bekko の配布物は2系統ある——汎用に使える PyTorch 重み (safetensors) と、低スペック環境・エッジ向けの ONNX / OpenVINO である。後者では、配布サイズの大半とロード時メモリを占める静的な語彙埋め込みテーブルを int8 圧縮で大きく削減し、モデルファイルを小型化する（§7.1）。推論速度は、AP が小さいため CPU・エッジデバイスから GPU まで一貫して高速である（§7.2）。§7.3 と §7.4 では、「量子化」という同じ言葉で呼ばれがちだが対象も影響も全く異なる2つの操作を分けて評価する——§7.3 はモデルの重み (Transformer 層を含む演算パラメータ) を int8 化する量子化、§7.4 は推論後に得られる出力ベクトルを int8/binary 化する量子化である。これらの帰結として、ブラウザ上でも動作する（§7.5）。本章の実用上の結論を表4 にまとめる。

表4: 環境別の配布物・構成のまとめ（根拠は各節）。

環境 / 用途	構成	理由	留意点
サーバ / エッジ CPU	OpenVINO (語彙埋め込み int8)	CPU 最速のバックエンド (§7.2)	最適化はバックエンド依存
Apple Silicon	PyTorch (MPS)	Mac では MPS が最速 (§7.2)	CPU のみの場合は OpenVINO
ブラウザ	ONNX (語彙埋め込み int8) + Transformers.js	124 / 190 MiB で配布可・WebGPU/WASM 動作 (§7.5)	初回ダウンロード時間・ピークメモリは未計測
GPU	PyTorch (bf16 / fp16) + FlashAttention-2	最高スループット (§7.2 の実測は fp16)	CUDA / FlashAttention-2 が必要
ベクトル DB (インデックス圧縮)	256 次元 + int8 + rescore	ほぼ無損失の圧縮 (§7.4)	rescore 用に fp32 ベクトルの保持が必要

7.1 メモリ: 静的トークン埋め込み行列の int8 化

Bekko の AP は小さいが、256,000 語彙 × 384 次元の多言語語彙埋め込み行列が総パラメータ・配布サイズの大半を占める (fp32 で 256000×384×4 バイト ≈ 375 MiB)。多言語モデルではパラメータの大半が埋め込み層に集中することは [Abdaoui et al. \(2020\)](#) も指摘しており、語彙側の圧縮が配布・ロード時メモリ削減に直結する。この行列は Transformer の行列積演算に寄与しない lookup テーブルであるため (参照・逆量子化・メモリ帯域のコストは残る; §1.2)、精度への影響を抑えながら積極的に圧縮できる。ONNX / OpenVINO 配布物は語彙埋め込み int8 版を既定とし、比較用に fp32 / fp16 版も併せて配布する (汎用の PyTorch 配布は safetensors のままとする)。

int8 化は row-wise symmetric (行ごと対称量子化) を用いる (配布サイズの内訳と削減効果は図6)。各トークン行 $\mathbf{w}_r \in \mathbb{R}^{384}$ について、スケール $s_r = \max_j |w_{r,j}| / 127$ を求め、 $\hat{w}_{r,j} = \text{round}(w_{r,j} / s_r) \in [-127, 127]$ を int8 で保存する。復元は $w_{r,j} \approx s_r \cdot \hat{w}_{r,j}$ 。保存物は int8 テーブル (256000×384×1) + fp32 行スケール (256000×4) で、静的埋め込みテーブルは fp32 比 約74.7% 削減される。ONNX グラフ上では `Gather(int8 table) → Cast(float) → Gather(row_scale) → Mul → 下流の fp32 Transformer` として実装する (行ごとスケールで量子化誤差が小さい)。以下、この構成を「語彙埋め込み int8 版」と呼ぶ (ONNX / OpenVINO の既定配布)。

モデル	fp32 ONNX	既定配布 (語彙埋め込み int8)	削減率	tokenizer.json
a8m	404.3 MiB	124.1 MiB	69.3%	32.8 MiB
a25m	470.3 MiB	190.1 MiB	59.6%	32.8 MiB

なお本稿で 124 / 190 MiB と書くときは ONNX モデルファイル単体のサイズを指す。ローカル配置・ダウンロードの合計は tokenizer.json (32.8 MiB) 等を加え、a8m 約 157 MiB / a25m 約 223 MiB である。

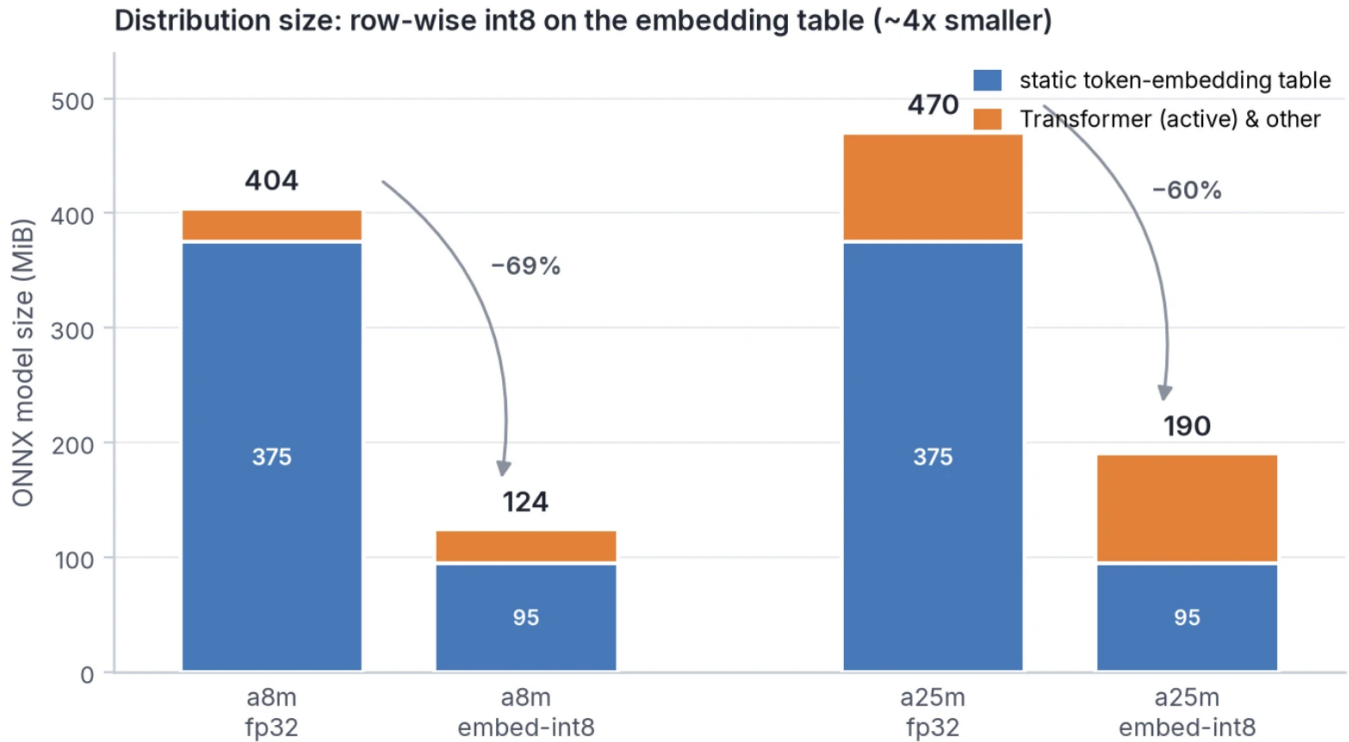


図6: 配布サイズの内訳と埋め込み int8 化の効果。圧縮対象は行列積演算に寄与しない静的埋め込みテーブルであり、AP (Transformer) 部分は変えない (出典: 公開モデルカードのファイル構成)。

精度の劣化はごく小さい——リリースモデル a8m での実測 (31 タスク; §7.3) では、語彙のみを int8 化したこの構成は量子化なしの基準と平均 -0.0001 の差にとどまり、リリース配布物では PyTorch とのベクトル一貫性 ($\min \cos \geq 0.9994$; a8m は x86・Raspberry Pi 5・Apple Silicon、a25m は x86・Raspberry Pi 5 で確認) も検証している。ロード時メモリも小さく、低リソース環境に適する。a8m は AP が一桁 M 台と小さいぶん、a25m よりさらに軽量である。

さらに踏み込んだ実験として (リリース前の prefix あり旧系譜・最大長 512 での予備実験; §8)、語彙そのものを 256,000 → 219,836 (FineWiki で観測した30言語のトークンのみ) へ事後的にプルーニング (語彙トリミング; Ushio et al., 2023) してから int8 化すると、配布サイズは a8m 110.7 MiB / a25m 176.7 MiB までさらに縮む (フル Nano-set 231 タスク平均で a8m -0.32% / a25m -0.14% とほぼ無損失)。ただし低資源・非ラテン文字言語 (例: テルグ語・スワヒリ語・ベンガル語の NanoMIRACL) で局所的な劣化が生じるため、既定の配布物は全語彙を保つ語彙埋め込み int8 版とし、この語彙プルーニング版はリリースには含めず実験結果の報告にとどめる (§8)。

7.2 推論速度 — CPU / Raspberry Pi / Apple Silicon / CUDA GPU

本研究の動機は GPU 非搭載・低リソース環境での実用にある。したがって最も重要なのは CPU 上での速度であり、まずこれを比較する。AP が小さいほど Transformer の演算量が減り、その差は CPU スループットに直接現れる (§1.2)。

CPU / Apple Silicon / CUDA GPU での比較。同一の Natural Questions 文書エンコード (バッチ 64, 最大 512 トークン) を、各環境で最速のバックエンド——x86 / Raspberry Pi 5 は OpenVINO (比較モデルは既定の OpenVINO / INT8 アーティファクト、Bekko は語彙埋め込み int8 版)、Apple Silicon は PyTorch MPS、CUDA GPU は PyTorch fp16 (SDPA / FlashAttention-2 の2実装を併記) ——で計測した。表中のモデルは統一集合 (§4.2) のうち全環境で実測を行ったサブセットであり、mE5-base・gte-m-base・embeddinggemma-300m・harrier-270m・arctic-l-v2.0・bge-m3 の x86 / Raspberry Pi 実測は含まない。したがって本節の CPU / Raspberry Pi 速度主張は、同一条件で実測した表中のモデルに対する比較である。

モデル	AP	x86 CPU	Raspberry Pi 5	Apple M4 Max (MPS)	CUDA (SDPA)	CUDA (FA2)
bekko-a8m	7.7M	364	33	592	4,697	5,561
mE5-small	21.6M	226	19	370	3,315	3,746
bekko-a25m	24.9M	134	10.5	351	3,243	4,006
granite-97m-r2	28.3M	125	10.0	286	3,221	3,917
granite-311m-r2	110.3M	38	2.9	106	1,497	2,159
mE5-large	303.9M	21	1.5	67	1,092	1,318

単位はすべて docs/s (Natural Questions 文書のエンコード数/秒; 高いほど速い)。x86 CPU = Ryzen 9 7950X (OpenVINO)、Raspberry Pi 5 = OpenVINO、Apple M4 Max = PyTorch MPS、CUDA = RTX 5090 (transformers 5.12.1・fp16, SDPA / FlashAttention-2 [FA2] ; NQ 100k で計測)。計測環境の詳細は付録 A。

表中の実測集合では a8m が CPU でも最速であり (図7)、x86 で mE5-small の約1.6倍・granite-97m の約2.9倍・mE5-large の約17倍、Raspberry Pi 5 でも最速 (mE5-small の約1.8倍、mE5-large の約22倍) である。a25m (x86 134 / Pi 10.5) は granite-97m とほぼ同等で、AP が4~12倍大きい granite-311m・mE5-large を大きく上回る。

一方で表は、AP が同程度ならアーキテクチャと推論実装も速度を左右することも示している。従来型 BERT (XLM-R 系) アーキテクチャの mE5-small (AP 21.6M, 226 docs/s) は、ModernBERT 系の a25m (AP 24.9M, 134)・granite-97m (AP 28.3M, 125) より CPU では速い。ModernBERT の効率化——unpadding、FlashAttention の可変長カーネル、交互 Global/Local attention——は GPU カーネルを前提としたハードウェア指向設計であり (Warner et al., 2024)、これらの利得が得られない CPU 実行では、シンプルな従来型 BERT の方が最適化された実装 (OpenVINO 等) に乗りやすい。すなわち AP は FLOPs (演算量) の主要因だが、実スループットには AP 以外の要因——アーキテクチャ・カーネル実装・バックエンド——も効く (§1.2 の注意と整合)。その上で、a8m は AP を一桁 M 台まで小さくすることでこのアーキテクチャ差を吸収してなお最速である。

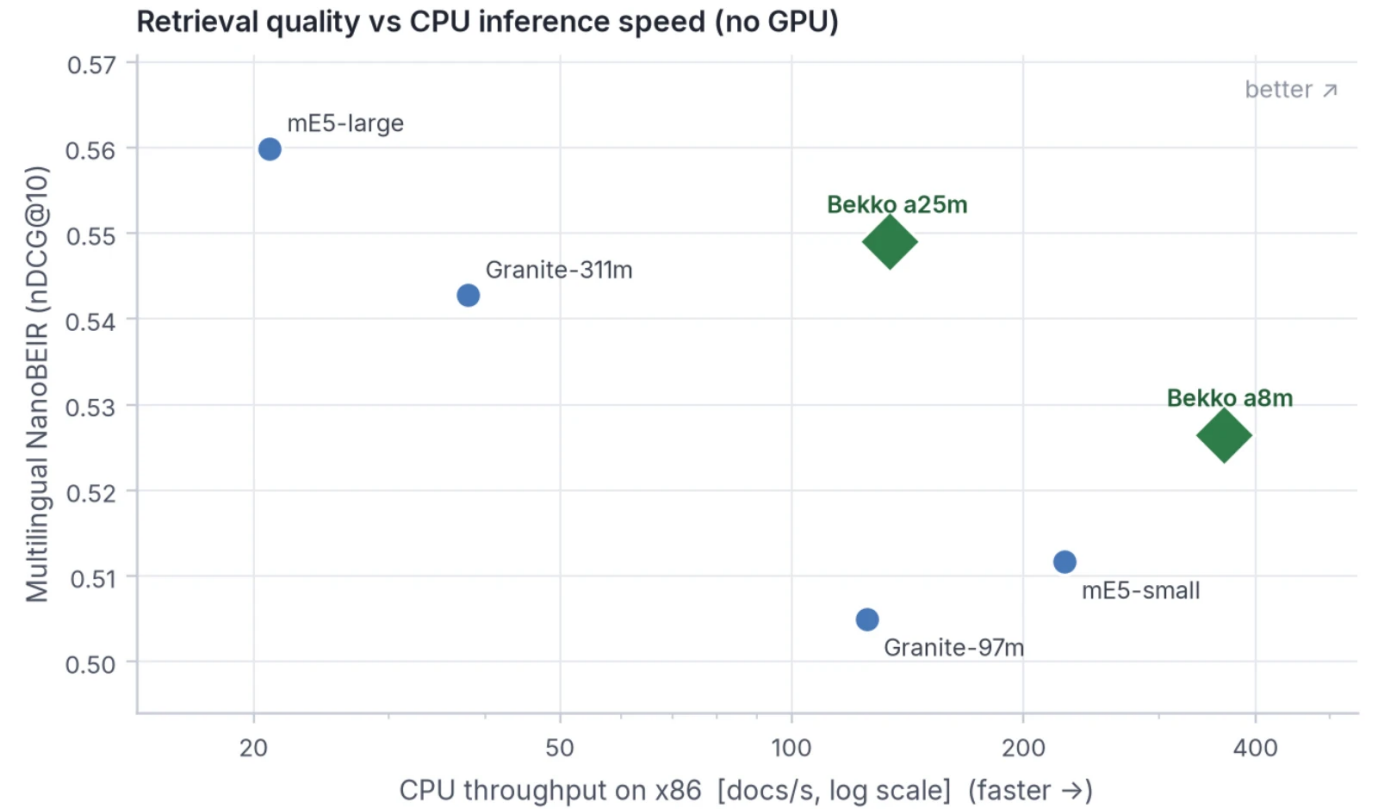


図7: 検索品質対 CPU 推論速度 (GPU 非搭載環境; 統一集合のうち CPU 実測のあるモデル)。a8m (364 docs/s) は mE5-small (226) より速くかつ高品質、a25m (134) は mE5-large (21) 級の品質を約6倍の速度で達成し、両者が品質・速度のフロンティアを形成する——これが本研究の主要な効率主張である。

Apple Silicon (表の MPS 列; 計測環境は付録 A) でも a8m が最速で、表外の bge-m3 (78 docs/s) の約7.6倍に達する。CPU のみでも OpenVINO で a8m が最速 (297 docs/s) である。ただし留意すべき点として、Mac の素の PyTorch CPU

(torch 2.12.1) では a8m (76 docs/s) が mE5-small (200) より遅い。ModernBERT 系のカーネルが Mac の PyTorch CPU で十分に最適化されていないためと考えられ (前述のアーキテクチャ・実装要因)、MPS / OpenVINO では逆転する。PyTorch のバージョンやビルドによって結果が変わりうる点にも注意されたい。

CPU バックエンドの選択。OpenVINO が最良の CPU バックエンドであり、その中で語彙埋め込み int8 版 (§7.1) を既定とする (OpenVINO 内の fp32/fp16/語彙 int8 の速度差は小さく、int8 版は配布サイズ・ロード時メモリの優位で選ぶ)。PyTorch CPU と比べ、a8m は x86 で 2.82 倍 (129→364 docs/s) ・Raspberry Pi 5 で 1.69 倍 (19.6→33 docs/s) 、a25m は x86 で 3.23 倍・Pi で 1.74 倍に高速化し、モデルロード時の RSS も約 210–280 MiB 削減する。語彙埋め込み int8 版の ONNX は、ブラウザでの実行が事実上 ONNX Runtime (Transformers.js) を前提とするため、ブラウザ配布に用いる (§7.5) 。サーバやエッジの CPU では OpenVINO の方が速い。

GPU でも最速。GPU 環境でも傾向は同じである (表の CUDA 列) 。a8m は FlashAttention-2 で 5,561 docs/s と計測した範囲で最速 (mE5-small の約1.5倍、bge-m3・mE5-large の約4.2倍; 表外の bge-m3 は 1,324・embeddinggemma-300m は 1,678 docs/s) であり、a25m (4,006) も a8m 以外の全モデルを上回る。FlashAttention-2 は SDPA 比で a8m +18% / a25m +24% 高速である。

7.3 モデル重みの量子化 — 語彙のみ int8 と全重み int8 の対比

本節はモデルの重みを int8 化する2つの構成を比較する (推論後の出力ベクトルの量子化は §7.4 で別に扱う) 。一つは §7.1 で述べた既定配布の構成——配布サイズの大半を占める語彙埋め込みテーブルだけを int8 化し、Transformer 層の重みは fp32 のまま残す (以下、語彙のみ int8) 。もう一つは、Transformer 層を含む全演算パラメータを int8 化する事後量子化 (OpenVINO qint8 / ONNX qint8; 以下、全重み int8) である。前者は行列積演算に手を触れないのに対し、後者は演算そのものを量子化する——この違いが精度に直結することを、同一条件の実測で示す。リリースモデル `bekko-embedding-v1-a8m` (エッジ配布の主対象である最小構成) を、bf16 の基準と同一チェックポイントで CPU 推論向けに構成し、NanoBEIR-en + NanoMMTEB-v2 (31 タスク) で評価した。

構成	mean nDCG@10	絶対変化	相対保持率
基準 (量子化なし; SentenceTransformers / CUDA / bf16)	0.5448	—	—
語彙のみ int8 (既定配布; OpenVINO CPU)	0.5447	−0.0001	100.0%
全重み int8: OpenVINO qint8 (CPU)	0.5193	−0.0255	95.3%
全重み int8: ONNX qint8 AVX512 (CPU)	0.5113	−0.0335	93.8%

語彙のみ int8 (既定配布) の精度劣化はほぼない——31 タスク平均で −0.0001 (保持率 100.0%) 、この 31 タスク内ではタスク単位でも最大 −0.007 にとどまり、8,192 トークン級の長文中の合言葉検索タスク (後述の全重み int8 で崩壊するタスク) も 0.876 → 0.873 と維持する。長文系のタスクセットでも同様で、NanoLongEmbed は平均 ±0.000、長文書内検索の NanoMLDR (13言語) は +0.008、コード検索の NanoCoIR も −0.001 と差がない (これらを含めたタスク単位の最大低下は −0.008) 。これは配布物のベクトル一貫性検証 (x86・Raspberry Pi 5・Apple Silicon で min cosine ≥ 0.9994; §7.1) とも整合する。

一方、全重み int8 は平均で 4.7~6.2% の精度低下を伴い、しかも劣化は一様ではなく、計測した長文系のタスクセットで特に大きい (入力長のほかタスク・言語・バックエンドも同時に変わるため、長さ単独の因果とは断定しない) 。31 タスク中の劣化の約半分~6割は上記の合言葉検索タスク 1 件の崩壊 (0.876 → 0.384) に由来し、これを除く 30 タスクの平均低下は −0.010~−0.017 にとどまる。長文タスクセットで測ると劣化は顕著で、長文検索 (NanoLongEmbed) で −8.6%、長文書内検索 (NanoMLDR, 13言語) では平均 −23.9% (特に非ラテン文字言語で崩壊級) に達する一方、中程度長のコード検索 (NanoCoIR) は −4.8% にとどまる。すなわち全重み int8 は、リリースモデルが第二段階で獲得した長文耐性 (§5.3) を最も損なう。

さらに、全重み int8 にはこの精度低下に見合う速度の利得も乏しい。高性能な x86 CPU (Ryzen 9 7950X, バッチ64, 最大 512トークン) では速度の利得が無く、OpenVINO qint8 (全重み int8, 397 docs/s) はむしろ語彙のみ int8 の OpenVINO (420 docs/s) より約 5% 遅い (CPU の高速化は OpenVINO ランタイム自体に由来し、Transformer 重みの int8 化からは得られない) 。一方、非力な ARM (Raspberry Pi 5, Cortex-A76 4コア) では ONNX の int8 カーネルが ONNX の語彙のみ int8 の 1.6 倍 (14.9 → 23.7 docs/s) に高速化するが、それでも OpenVINO の語彙のみ int8 (28.7 docs/s) には及ばず、前述の精度低下と一致性の問題を伴う。なお本節の速度は、量子化バックエンド間の相対比較を目的に同一マシン・同一スクリプトで測ったものであり、計測系・ライブラリ版が異なる §7.2 の表とは絶対値を直接比較しない。加えて全重み int8 は実行環境依存性も強く、a25m の OpenVINO qint8 は配布時のベクトル一貫性検査 (cosine 0.98 閾値) を通らず配布自体を見送ってお

り、a8m の OpenVINO qint8 は ARM 環境 (Raspberry Pi 5・Apple Silicon) で出力が全要素 NaN となり、ONNX qint8 も ARM ではベクトル一貫性が cosine 0.77 前後まで低下する。

以上より、計測した構成の範囲では、速い CPU でも遅い CPU でも、精度をほぼ落とさずに最速だったのは語彙のみ int8 の既定配布 (Transformer 層は fp32 のまま) であり、Transformer 層まで含めた全重み int8 の量子化には精度低下に見合う利得がなかった。

7.4 出力ベクトルの圧縮: 次元削減と量子化

§7.3 までは扱ったのはモデル側 (重み) の圧縮である。本節は、推論で得られた出力ベクトル (埋め込み) 自体の圧縮を扱う。出力ベクトルの圧縮はベクトルデータベースのインデックスサイズと検索時のメモリ・帯域の削減に直結する一方、モデルの重み・推論速度には影響しない。圧縮の手段は2つあり、掛け合わせて使える。

- ・次元削減: MRL (§3.4) により、384 次元の出力を先頭 256 / 128 / 64 次元に切り詰めて使う。
- ・出力ベクトルの量子化: fp32 の各次元を int8 (1/4 サイズ) または binary (1/32 サイズ) へ量子化する。

出力ベクトルの量子化では、rescore (再採点) の併用でどれだけ検索順位を復元できるかも併せて確認する。rescore とは、量子化ベクトルで候補を粗く検索し、上位候補 (本評価では上位 100 件) だけを fp32 ベクトルで再採点する二段階検索 (全コーパスの再探索ではない) であり、BPR (Yamada et al., 2021) が binary code+連続ベクトル再ランキングで精度を保ったままメモリを削減した方式に対応する。リリースモデル両方の全組合せ (4次元 × {fp32, int8, int8+rescore, binary, binary+rescore}) を HAKARI-Bench Overall で評価した (図8)。

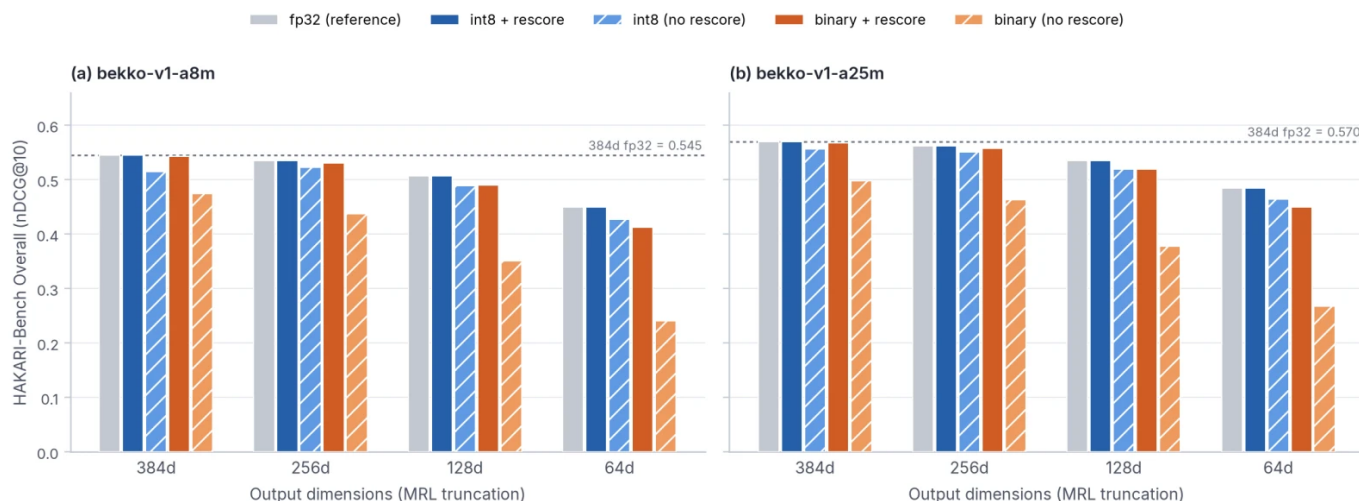


図8: 出力ベクトルの圧縮 (次元削減 × 量子化, HAKARI-Bench Overall)。各次元グループ内で int8+rescore の棒は fp32 とほぼ同じ高さであり、劣化は次元削減由来分に収束する。rescore なしの量子化は損失が大きく、binary は低次元との併用で劣化が増幅する。

結果は4点にまとまる (基準 = 384次元 float; a8m 0.5453 / a25m 0.5700。Overall は表3 と同じ 538 raw タスク micro)。

1. 次元削減は 256 次元までなら損失が小さい: -1.8% (a8m) / -1.3% (a25m)。128 次元で -7.0% / -6.2%、64 次元では -17.5% / -15.0% と大きく落ちる。
2. rescore なしの量子化は損失が大きい: 384 次元でも int8 単体で -5.5% / -2.4%、binary 単体で -12.9% / -12.6% 低下する (a8m の方が int8 量子化に敏感である)。
3. rescore を併用すればほぼ無損失になる: int8+rescore は -0.04% / -0.03%、binary+rescore も -0.44% / -0.38% にとどまる。
4. 次元削減と量子化を掛け合わせても、int8+rescore なら劣化は次元削減分だけに収束する (例: a25m の 128 次元で float 0.5348 に対し int8+rescore 0.5348)。一方 binary は低次元との併用で劣化が増幅し (a25m 64次元 binary+rescore -21.1%; float 単体の -15.0% より悪い)、rescore なしの binary 64次元 (-53.0~-55.9%) は実用外である。

実用上の指針としては、「256次元 + int8 + rescore」が、次元削減分 (-1.3~-1.8%) を除けば量子化による追加損失がほぼゼロの実用域であり、binary は rescore を前提とした候補生成専用として使うべきである。ただし rescore を用いる構成では、再採点のために fp32 (または高精度) ベクトルを別途保持する必要があり、メモリ削減の効果は候補生成用インデックス側に限られる。rescore でほぼ完全に復元できる理由が QAT の寄与か埋め込み分布の性質そのものかは、切り分けられていな

い (§6.3)。なお、次元削減と量子化の組合せの最適点がタスク依存であることは [Kisako et al. \(2026\)](#) も報告しており、Bekko は MRL truncate と出力量子化の双方を提供することで運用側の選択肢を確保している。

7.5 ブラウザ上での動作

Bekko は巨大な多言語トークナイザを持ちながら、ONNX + 語彙埋め込み int8 で a8m 124 MiB / a25m 190 MiB と小さく、リモートサーバを介さずにブラウザ上で直接動作する。実装は Transformers.js ベースで、語彙埋め込み int8 版の `onnx/model.onnx` をロードして、WebGPU が使える環境では WebGPU、なければ WASM (CPU) にフォールバックする。MRL の 384/256/128/64 次元切り替えにも対応する。利点は、(1) 入力や文書を外部へ送信せず端末内で処理が完結しプライバシーに優れること、(2) サーバや推論 API のコストとレイテンシが不要なこと、(3) オフライン／エッジでの検索・RAG が可能なこと、である。多言語検索モデルをブラウザにロードできるこの小ささは、超小型 AP と語彙埋め込み int8 化の直接の帰結である。

ブラウザ実機での速度を計測した。計測環境は §7.2 の Apple Silicon 計測と同一マシン (Apple M4 Max, macOS 26.5) 上の Chrome 149 である (付録 A)。Natural Questions の文書 1,000 件のエンコード (バッチ 64) を、同一マシンのネイティブ実行と比較する。

実行形態	a8m	a25m
ネイティブ PyTorch MPS (Apple GPU; §7.2)	592	351
ネイティブ OpenVINO (CPU; §7.2)	297	97
ブラウザ WebGPU	187	89
ブラウザ WASM (CPU)	51	16

単位はすべて docs/s (Natural Questions 文書のエンコード数/秒)。

ブラウザ WebGPU はネイティブ MPS の約 1/3 (a8m) ~1/4 (a25m)、ブラウザ WASM はネイティブ OpenVINO CPU の約 1/6 のスループットで動作する。WebGPU は WASM の 3.7~5.7 倍高速で、AP の小さい a8m は WebGPU で a25m の約 2.1 倍・WASM で約 3.3 倍高速——AP 極小化の利得はブラウザ上でも一貫して現れる。この計測は単一環境・エンコードのみの簡易なものであり、詳細なブラウザベンチマークを意図したものではない——多言語埋め込みモデルがブラウザ上でも実用的な速度で動作することの実証を目的とする。

8. 限界と今後の課題

- 言語カバレッジの偏り: 学習データは英語と主要言語に偏っており、低資源言語での性能は限定的である。小型モデルほど、対象とするドメイン・言語を学習データで直接広くカバーしておく必要があると考えられ、未対応のドメイン・言語のデータを増やす余地が大きい。低資源言語向けデータ合成の強化が今後の方向性である。
- 情報検索性能以外のタスク: Classification は統一比較集合の下位に位置し、Clustering・MultilabelClassification も比較群の中位にとどまる (prefix を付与して学習した構成との比較では大きく改善しているが、prefix 以外の要因も含むモデルレベルの比較である; §6.2・付録 F.1)。検索に焦点を置いた学習データ・損失設計の帰結と考えられ、検索性能を保ったままこれらのタスクタイプを改善することが今後の課題である。
- トークナイザの縮小: 256,000 語彙の埋め込み行列が配布サイズの大半を占めており、語彙側にはさらなる縮小余地がある。旧構成でのローカル予備実験では、観測されたトークンのみへ語彙を再学習なしで事後プルーニングする構成 (§7.1; 未リリース) が配布サイズをさらに削減し平均ではほぼ無損失だったが、低資源・非ラテン文字言語に局所的な劣化が残った。保持語彙リストの改善や語彙削減を伴う再学習により、この劣化リスクを抑えつつさらなる小型化を狙う。
- 基盤モデルの構築方法: 本研究ではプルーニングで基盤を抽出したが、超小型構成をフルスクラッチで事前学習すれば、プルーニングで失われる表現力を取り戻せる可能性がある。また本研究は蒸留を用いない構成の到達点を示したものであり、大型モデルを教師とした蒸留を併用すればさらなる改善の余地がある。
- アブレーションの統計的限界: §6 のレシピア比較 (prefix・MRL 重み・QAT) はいずれも各設定 1 run に基づく観察であり、シード分散の推定や有意性検定を行っていない。0.01 未満の差はこの制約の下で読む必要がある。QAT の正則化仮説の因果的検証には至っておらず、§5.3 の長文耐性の由来 (RoPE 設定か長文データか) もアブレーション未実施である。
- 評価データの独立性と規模: Multilingual NanoBEIR は同一の英語 13 タスクの機械翻訳直積 (各タスク約 50 クエリの軽量評価) であり、182 タスクは独立な課題ではない (§4.1, 付録 G.1)。また学習混合には評価と同系列の IR ファミリー (MS MARCO・mMARCO・MIRACL・HotpotQA 等; §3.3, 付録 B-C) が含まれ、評価タスクとの split 単位の重複監査は体系的には実施していない。公式 MMTEB を主評価①に置くことでリスクを緩和しているが、正式な train/eval 重複監査は今後の課題である。

7. a8m/a25m 比較の交絡: 両モデルは第二段階データこそ同一だが、学習率・勾配キャッシュ設定・a25m のチェックポイントマージなど学習設定が異なるため (§3.4–3.5)、両者の差を AP (容量) のみの効果として解釈できない (§5 冒頭)。
8. 速度評価の範囲: §7.2 はバッチ 64 の文書エンコード・スループットの計測であり、対話的検索で重要なバッチ 1 のクエリレイテンシ (p50/p95) や、固定トークン長での長さスケールリングは本稿では扱っていない。スループットの優位から対話レイテンシの優位を直接は導けない。

9. 結論

本研究は、推論の演算量を主に決める Active Parameters (AP) を効率の軸として多言語埋め込みモデルへ一貫適用し (Kaplan et al., 2020; Lan et al., 2019 の非埋め込みパラメータの見方に依拠)、この軸で超小型・多言語埋め込みモデル Bekko Embedding を構築した。手法の要点は3つである。第一に、蒸留を用いず、mmBERT-small の22層を Global-Local リズムを保つ構造的レイヤーブルーニングで4層 (a8m, AP 8M 弱) / 13層 (a25m, AP 25M 弱) へ削り、ブルーニング済みモデルを base model として学習する。第二に、小型モデルの知識不足を、公開コーパス約11.5億ペア (実効学習約11.1億ペア; 二系統の LLM 合成データセットを含む) の多言語データで補う。第三に、ペアタイプ別に損失方向を切り替えるマスク付き対照損失・MRL を統合した二段階対照学習を、単一GPU のみで完了する。

評価では、公式 MMTEB Multilingual v2 の Retrieval で a8m (56.2) が mE5 系・bge-m3 を上回り、a25m (57.5) はさらに gte-m-base (57.2) と同水準に達し、AP が1~2桁大きいモデルに検索性能で匹敵・凌駕することを示した (§5.1)。この傾向は Multilingual NanoBEIR (14言語) でも一貫し、長文検索・長文書内検索・コード検索にも強い (§5.2–5.3)。一方で MMTEB 総合では中位にとどまり、Classification 系タスクを中心に総合の伸び代を残す。設計探索からは、初期の連続レイヤーに加え、最終レイヤーではなく遠方の深層 Global 層を残すブルーニングが有効という設計ヒューリスティックを観察した (§6.1; 因果的局在を主張するものではない)。また学習レシピの統制比較では、量子化対応学習 (QAT) の併用が性能を損なわず、最小構成の a8m ではむしろ開発用評価を明確に高めるという観察を得た (各設定 1 run; §6.3)。効率面では、同一ワークロード・同一ハードウェアで計測した比較モデルの中で a8m は CPU・GPU 双方で最速であり (§7.2)、行列積演算に寄与しない語彙埋め込みを row-wise int8 化した ONNX / OpenVINO 配布物 (モデルファイル a8m 124 MiB / a25m 190 MiB) は Raspberry Pi 5 やブラウザでも動作する (§7)。モデル重みと学習データをオープンに公開しており、本研究が低リソース環境・オンデバイスでの多言語検索の実現と、小型多言語埋め込みモデル研究の再現・発展の基盤となることを期待する。

謝辞 (Acknowledgements)

本研究の学習・損失実装は sentence-transformers (Reimers & Gurevych, 2019) をベースに構築した。GradCache 系のキャッシュ付き対照損失・MatryoshkaLoss・各種バッチサンプラーの枠組みを活用し、その上にマスク付き双方向損失・QAT・ペアタイプ別損失・トークン長グループ別学習を実装した。sentence-transformers の成熟した基盤があったからこそ、単一GPU 規模の研究でも学習実装と評価を容易に行うことができた。基盤を提供する mmBERT (Marone et al., 2025) / ModernBERT (Warner et al., 2024)、データ設計の着想を与えた Ruri (Tsukagoshi & Sasano, 2026) にも謝意を表する。

References

- Aarsen, T. (2025). Train 400x faster Static Embedding Models with Sentence Transformers. Hugging Face blog. <https://huggingface.co/blog/static-embeddings>
- Abdaoui, A., Pradel, C., & Sigel, G. (2020). Load What You Need: Smaller Versions of Multilingual BERT. [arXiv:2010.05609](https://arxiv.org/abs/2010.05609)
- Bonifacio, L., Abonizio, H., Fadaee, M., & Nogueira, R. (2022). InPars: Data Augmentation for Information Retrieval using Large Language Models. [arXiv:2202.05144](https://arxiv.org/abs/2202.05144)
- Câmara, A. (Zeta Alpha). (2024). Fine-tuning LLMs for State-of-the-Art retrieval (NanoBEIR を公開したブログ記事) . [Zeta Alpha blog / HF collection zeta-alpha-ai/nanobeir](https://zeta-alpha.ai/nanobeir)
- Chen, J., Xiao, S., Zhang, P., Luo, K., Lian, D., & Liu, Z. (2024). M3-Embedding: Multi-Linguality, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation. [arXiv:2402.03216](https://arxiv.org/abs/2402.03216)
- Conneau, A., et al. (2019). Unsupervised Cross-lingual Representation Learning at Scale (XLM-R). [arXiv:1911.02116](https://arxiv.org/abs/1911.02116)
- Dai, Z., Zhao, V. Y., Ma, J., et al. (2023). Promptagator: Few-shot Dense Retrieval From 8 Examples. [arXiv:2209.11755](https://arxiv.org/abs/2209.11755)
- Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. [arXiv:1810.04805](https://arxiv.org/abs/1810.04805)

- Enevoldsen, K. C., et al. (2025). MMTEB: Massive Multilingual Text Embedding Benchmark. [arXiv:2502.13595](#)
- Feng, F., Yang, Y., Cer, D., Arivazhagan, N., & Wang, W. (2022). Language-agnostic BERT Sentence Embedding (LaBSE). [arXiv:2007.01852](#)
- Gao, L., Zhang, Y., Han, J., & Callan, J. (2021). Scaling Deep Contrastive Learning Batch Size under Memory Limited Setup (GradCache). [arXiv:2101.06983](#)
- Granite Team. (2025a). Granite Embedding Models. [arXiv:2502.20204](#)
- Granite Team. (2025b). Granite Embedding R2 Models. [arXiv:2508.21085](#)
- Granite Embedding Team, IBM Research. (2026). Granite Embedding Multilingual R2 Models. [arXiv:2605.13521](#)
- Gromov, A., Tirumala, K., Shapourian, H., Gloriosi, P., & Roberts, D. A. (2024). The Unreasonable Ineffectiveness of the Deeper Layers. [arXiv:2403.17887](#)
- Jacob, B., et al. (2018). Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. [arXiv:1712.05877](#)
- Kaplan, J., McCandlish, S., Henighan, T., et al. (2020). Scaling Laws for Neural Language Models. [arXiv:2001.08361](#)
- Karpukhin, V., et al. (2020). Dense Passage Retrieval for Open-Domain Question Answering. [arXiv:2004.04906](#)
- Kisako, R., Tsukagoshi, H., & Sasano, R. (2026). When Is 0.1% Enough? Analyzing the Combined Effects of Dimensionality Reduction and Quantization on Text Embedding Compression. [arXiv:2606.01074](#)
- Kusupati, A., et al. (2022). Matryoshka Representation Learning. [arXiv:2205.13147](#)
- Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2019). ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. [arXiv:1909.11942](#)
- Lewis, P., et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. [arXiv:2005.11401](#)
- Li, X., Dong, K., Lee, Y. Q., Xia, W., Zhang, H., Dai, X., Wang, Y., & Tang, R. (2024). CoIR: A Comprehensive Benchmark for Code Information Retrieval Models. [arXiv:2407.02883](#)
- Li, Z., Zhang, X., Zhang, Y., Long, D., Xie, P., & Zhang, M. (2023). Towards General Text Embeddings with Multi-stage Contrastive Learning (GTE). [arXiv:2308.03281](#)
- Liquid AI. (2025). NanoBEIR Multilingual Extended (Japanese/Korean translations). [HF dataset](#)
- Liu, F., Enevoldsen, K. C., Solomatin, R., Chung, I., Aarsen, T., & Fődi, Z. (2025). Introducing RTEB: A New Standard for Retrieval Evaluation. Hugging Face blog. <https://huggingface.co/blog/rteb>
- Marone, M., Weller, O., Fleshman, W., Yang, E., Lawrie, D., & Van Durme, B. (2025). mmBERT: A Modern Multilingual Encoder with Annealed Language Learning. [arXiv:2509.06888](#)
- Men, X., Xu, M., Zhang, Q., Wang, B., et al. (2024). ShortGPT: Layers in Large Language Models are More Redundant Than You Expect. [arXiv:2403.03853](#)
- Messmer, B., Sabolčec, V., & Jaggi, M. (2025). Enhancing Multilingual LLM Pretraining with Model-Based Data Selection. [arXiv:2502.10361](#)
- Microsoft. (2026). harrier-oss-v1: Multilingual text embedding models. Hugging Face model card. <https://huggingface.co/microsoft/harrier-oss-v1-270m>
- Muennighoff, N., Tazi, N., Magne, L., & Reimers, N. (2022). MTEB: Massive Text Embedding Benchmark. [arXiv:2210.07316](#)
- Neelakantan, A., Xu, T., Puri, R., et al. (2022). Text and Code Embeddings by Contrastive Pre-Training. [arXiv:2201.10005](#)
- Penedo, G., et al. (2024). The FineWeb Datasets: Decanting the Web for the Finest Text Data at Scale. [arXiv:2406.17557](#)
- Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks (sentence-transformers). [arXiv:1908.10084](#)
- Reimers, N., & Gurevych, I. (2020). Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation. [arXiv:2004.09813](#)
- Sajjad, H., Dalvi, F., Durrani, N., & Nakov, P. (2023). On the Effect of Dropping Layers of Pre-trained Transformer Models (Poor Man's BERT). Computer Speech & Language, 77, 101429. [arXiv:2004.03844](#)
- Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. [arXiv:1910.01108](#)

- Schechter Vera, H., et al. (2025). EmbeddingGemma: Powerful and Lightweight Text Representations. [arXiv:2509.20354](#)
- Serbian AI Society. (2025). NanoBEIR-sr: Serbian translation of NanoBEIR. [HF dataset](#)
- Sionic AI. (2025). NanoBEIR-th / NanoBEIR-vi: Thai and Vietnamese translations of NanoBEIR. [HF dataset \(th\)](#) / [HF dataset \(vi\)](#)
- Sourty, R. (LightOn). (2025). NanoBEIR-Multilingual: Multilingual version of NanoBEIR (machine-translated, 8 languages). [HF dataset](#)
- Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., & Liu, Y. (2021). RoFormer: Enhanced Transformer with Rotary Position Embedding (RoPE). [arXiv:2104.09864](#)
- Tatenno, Y. (2026). HAKARI-Bench: A Lightweight Benchmark for Comparing Retrieval Architectures and Efficiency Settings under Unified Conditions. [arXiv:2606.22778](#)
- Thakur, N., Reimers, N., Rücklé, A., Srivastava, A., & Gurevych, I. (2021). BEIR: A Heterogenous Benchmark for Zero-shot Evaluation of Information Retrieval Models. [arXiv:2104.08663](#)
- Thakur, N., Ni, J., Hernández Ábrego, G., et al. (2024). Leveraging LLMs for Synthesizing Training Data Across Many Languages in Multilingual Dense Retrieval (SWIM-IR). [arXiv:2311.05800](#)
- Tsukagoshi, H., & Sasano, R. (2026). Ruri: 日本語に特化した汎用テキスト埋め込みモデル. 自然言語処理 (Journal of Natural Language Processing) 33(2), 591–629. DOI:10.5715/jnlp.33.591 / [J-Stage PDF](#) (preprint: [arXiv:2409.07737](#))
- Tulkens, S., & van Dongen, T. (2024). Model2Vec: Fast State-of-the-Art Static Embeddings. Zenodo. DOI:10.5281/zenodo.17270888 / [GitHub](#)
- Ushio, A., Zhou, Y., & Camacho-Collados, J. (2023). An Efficient Multilingual Language Model Compression through Vocabulary Trimming. [arXiv:2305.15020](#)
- Wang, L., et al. (2022). Text Embeddings by Weakly-Supervised Contrastive Pre-training (E5). [arXiv:2212.03533](#)
- Wang, L., et al. (2024). Multilingual E5 Text Embeddings: A Technical Report. [arXiv:2402.05672](#)
- Warner, B., et al. (2024). Smarter, Better, Faster, Longer: A Modern Bidirectional Encoder for Fast, Memory Efficient, and Long Context Finetuning and Inference (ModernBERT). [arXiv:2412.13663](#)
- Weller, O., Ricci, K., Marone, M., Chaffin, A., Lawrie, D., & Van Durme, B. (2026). Seq vs Seq: An Open Suite of Paired Encoders and Decoders (Ettin). In ICLR 2026. [arXiv:2507.11412](#)
- Yamada, I., Asai, A., & Hajishirzi, H. (2021). Efficient Passage Retrieval with Hashing for Open-domain QA (Binary Passage Retriever, BPR). [arXiv:2106.00882](#)
- Yu, P., Merrick, L., Nuti, G., & Campos, D. (2024). Arctic-Embed 2.0: Multilingual Retrieval Without Compromise. [arXiv:2412.04506](#)
- Zhang, X., et al. (2024). mGTE: Generalized Long-Context Text Representation and Reranking Models for Multilingual Text Retrieval. [arXiv:2407.19669](#)
- Zhang, Y., et al. (2025). Qwen3 Embedding: Advancing Text Embedding and Reranking Through Foundation Models. [arXiv:2506.05176](#)
- Zhu, D., Wang, L., Yang, N., Song, Y., Wu, W., Wei, F., & Li, S. (2024). LongEmbed: Extending Embedding Models for Long Context Retrieval. [arXiv:2404.12096](#)

Appendix

A. 学習・推論環境

学習環境:

項目	値
GPU (学習)	NVIDIA RTX PRO 6000 Blackwell Max-Q × 1
学習時間	第一段階: a8m 約2.9日 / a25m 約7.7日。第二段階: a8m 約1〜2時間 / a25m 約3時間

項目	値
学習ソフトウェア	Python 3.11 / PyTorch 2.8.0 (CUDA 12.9 ビルド; ドライバは CUDA 13.1) / transformers 4.57.6 / sentence-transformers 5.3 系 (独自拡張) / flash-attn 2.8.3 / accelerate 1.12.0 / datasets 4.5.0
フレームワーク	sentence-transformers ベースの独自 Trainer

推論速度の計測環境 (§7.2 の一次データに対応) :

環境	ハードウェア	主要ソフトウェア
x86 CPU	AMD Ryzen 9 7950X (16コア/32スレッド, AVX-512 対応)	Python 3.11 / PyTorch 2.8.0 / sentence-transformers 5.3 系 / ONNX Runtime 1.26.0 / OpenVINO 2026.2.0
Raspberry Pi 5	ARM Cortex-A76 4コア (aarch64)	Python 3.11 / PyTorch 2.12.1 / sentence-transformers 5.6.0 / ONNX Runtime 1.27.0 / OpenVINO 2026.2.1
Apple Silicon	M4 Max (macOS 26.5)	Python 3.12 / PyTorch 2.12.1 / sentence-transformers 5.6.0 / ONNX Runtime 1.27.0 / OpenVINO 2026.2.1
GPU (CUDA)	NVIDIA RTX 5090	Python 3.12 / PyTorch 2.10.0 (CUDA 13.0) / transformers 5.12.1 / sentence-transformers 5.6.0 / FlashAttention 2.8.3 (fp16)
ブラウザ	Apple M4 Max (macOS 26.5; Apple Silicon 行と同一マシン) 上の Chrome 149	Transformers.js 4.2.0 (WebGPU / WASM) + ONNX 語彙埋め込み int8 版

ブラウザ計測は単一環境・エンコードのみの簡易計測である (§7.5)。

B. 第一段階データの詳細 (bekko-embedding-v1-unsupervised)

第一段階データ (公開データセット `bekko-embedding-v1-unsupervised`) の中身を系統別に示す。データセットは 2,452 サブセット (それぞれ Hugging Face の dataset config として配布) に分かれ、各サブセットは名前から「どのソース由来か」「ペアの種類 (dd=doc_doc / qd=query_doc / 明示的負例付きの triplet)」がわかる命名規則を持つ。学習時の前処理は、(1) 512 トークンでの truncate、(2) バッチ内重複を避けるための 8,192 行ブロック ID の付与 (§3.3 の重複排除ブロック) を全サブセット共通で行い、対訳 bitext 系 (NLLB・CCMatrix 由来) にはさらに (3) 対訳品質スコアが最低の言語ペアのロード時除外 (§3.3 の品質フィルタ) を適用する。学習時のバッチは常に単一サブセットから構成され、サブセットを跨いで混ざることはない (§3.3)。

サブセット数をペアタイプ別に見ると、doc_doc 1,975 (約80%) / query_doc 477 であり、両タイプ合わせて 34 サブセットが明示的負例を持つ triplet 形式である (§3.3; triplet は内数)。doc_doc のサブセットが多いのは対訳 bitext を言語ペアごとに別サブセットへ分割しているためで (nllb-sampled だけで 1,575 サブセット)、行数では query_doc (約6.6億行) の方が多い (§3.3)。

source	サブセット数	主タイプ	内容
nllb-sampled-500k	1,575	doc_doc	NLLB・CCMatrix 由来の言語ペア総当たり対訳 (多対多・低資源含む、各≤500k 行)
finewiki	314	query_doc	Wikipedia の title→本文 (314言語)
multilingual-cc-news-...related-paragraph-pairs	134	doc_doc	同一ニュース記事内の段落↔段落 (134言語)
parallel-sentences-opus-100	99	doc_doc	OPUS-100 の en↔xx 対訳文
nllb-english-bitext-hq	95	doc_doc	英↔95言語の高品質対訳 (NLLB・CCMatrix 由来; bge-reranker-v2-m3 + bge-m3 でフィルタ)
multilingual_cc_news	47	query_doc	ニュース見出し→本文 (47言語)
parallel-sentences-wikimatrix	39	doc_doc	WikiMatrix マイニング対訳
parallel-sentences-wikititles	1	doc_doc	WikiTitles の対訳記事タイトル (全言語ペアを 1 サブセットに集約)
nomie-embed-unsupervised-data	25	qd17 / dd8	StackExchange/Quora/PAQ/gooaq 等の既存弱教師ペア
FineWeb2-IR	21	qd_triplet	query-crafter 合成クエリ+BGE-M3 hard negative (21言語×約3M)
swim-ir-{mono,cross}-lingual	10/17	query_doc	SWIM-IR 合成 IR (cross は query≠doc 言語)
wikipedia-multilingual-ir-{pairs,related-lead,related-paragraph}	各11	qd/dd	title+section→段落 / lead↔関連lead / 同記事段落↔段落 (11言語)

source	サブセット数	主タイプ	内容
wikipedia-multilingual-synthetic-ir-query-short_doc	11	query_doc	query-crafter 合成クエリ→短文書
mmarco-hard-negatives-reranker-filtered	10	qd_triplet	MS MARCO 多言語+reranker フィルタ hard negative
Nemotron-Post-Training-Dataset-v2	9	query_doc	指示→応答 (chat/code/math/stem + 5言語)
OpenCodeReasoning	1	query_doc	プログラミング問題→解答のペア
lightonai/embeddings-pre-training	3	query_doc	hermes / beir_dbpedia / webfaq
{wikipedia-english,fineweb,ccnews,arxiv,pubmed}-ir-simulated-search-queries	各1	query_doc	Qwen3.5-35B-A3B 合成クエリ (専門家の検索行動を模す)
{all-nli, wikipedia-synthetic-nli}	各1	dd_triplet	NLI / 事実改変による合成 hard negative
miracl	1	qd_triplet	MIRACL 多言語

系統別の中身と実例 (実データより) :

- 対訳 bitext (doc_doc) : 2言語の同義文。例 en 「When food is gone you are my daily meal」↔ ja 「食べ物が無くなった時、あなたは…」 (nllb-english-bitext-hq)、en↔ja 「All right. Just sign here?」↔ 「よし ここにサインだな？」 (opus-100)。
nllb-sampled は ace_Latn-ban_Latn のような言語ペア単位のサブセットで多対多。nllb 系サブセットの上流は NLLB プロジェクトと CCMatrix のマイニング対訳である。
- doc_doc (非対訳, broad positive) : 同記事の段落↔段落 (Wikipedia/ニュース)、記事 lead↔ハイパーリンク先 lead (関連話題)。パラフレーズではなく「同トピックの広い正例」 (title→本文は非対称ペアとして query_doc に分類する)。NLI triplet (all-nli、および日付などを微妙に改変した合成 hard negative の wikipedia-synthetic-nli)。
- query_doc (クエリ→文書の非対称ペア) : 既存の IR データセット (nomic / lighton / swim-ir / miracl / mmarco) に加え、二系統の合成クエリを用いる。(1) 高速合成 (query-crafter-multilingual; 付録 D) は、FineWeb2 由来の各文書に対してキーワード・自然文の質問・タイトルなど複数種類の短いクエリを生成する。例: ja 「セブ留学のマンツーマン教育」→ 該当 Web 文書。(2) 高品質合成 (Qwen3.5-35B-A3B, NVFP4 量子化) は、arXiv・PubMed・CC-News・fineweb-edu・英語 Wikipedia の文書に対して文脈を読んだ検索クエリを生成する。例: arXiv 「sparse PCA algorithm selecting coordinates of largest variance…」→ 該当論文 abstract。いずれも、直接的な表現と間接的な表現、流暢な質問文とキーワード列を意図的に混在させる。
- triplet (34, 1.4%) : 明示的 hard negative を持つ (FineWeb2-IR / mmarco / miracl / NLI 系)。

全体としては次の性格を持つ。言語カバレッジが非常に広く (finewiki 314 言語・CC-News 134 言語・対訳 1,600 超の言語ペア)、検索の中核信号は LLM 合成クエリ (query-crafter / Qwen3.5-35B-A3B) が高い比率を占める。正例の「近さ」は厳密な対訳から同トピックの広い正例までを意図的に混在させており、第二段階で高品質な IR データに絞り込むことを前提とした「広く浅い」設計である。品質フィルタは reranker・埋め込みモデルに依存するためそのバイアスが残る。また合成・既存 IR データは英語・主要言語中心で、対訳 bitext と Wikipedia・ニュース系が低資源言語を補う二層構造になっている。

総行数・系統別行数・対訳言語ペア数は、公開データの実測で 1,146,929,152 行 (≈11.5億) に確定した (自作 LLM 合成 164,478,976 行、対訳 bitext 330,678,272 行・正規化 1,622 言語ペア)。言語・言語ペア単位の内訳は、公開データセットのサブセット (言語ペア・言語ごとに分割されている) から算出できる。

C. 第二段階データソース一覧

第二段階 (hard negatives ファインチューニング) の実効混合を示す。a8m・a25m と同一の 37 サブセット・計 1,780,001 行である (リリースモデルの学習 manifest 実測。なお manifest は下表「追加 query_doc / doc_doc / ドメイン」の 10 データセットを 3 グループのサブセットとして数え、表では内訳を展開して示す)。

系統	サブセット	実効件数	備考
Ruri v3 FT (10 サブセット)	auto-wiki-qa-nemotron 154,956 / nli 222,429 / jsquad 61,659 / quiz-no-mori 37,437 / jaquad 31,253 / quiz-works 30,631 / mkqa 30,310 / miracl 25,905 / jqara 19,515 / mr-tydi 18,200	632,295	最大 512 トークン
BGE 系 query_doc	squad 87,599 / hotpotqa 84,516 / dureader 80,416 / mr-tydi 48,729 / miracl 40,203 / msmarco 200,000 / mmarco 中国語 80,000 / other 500	621,963	BGE-M3 FT データのうちライセンスが比較的明確なもの。MS MARCO・中国語 mMARCO は上限付き
	BQ 12,518 / ATEC 11,325 / LCQMC 10,000 / PAWSX 9,900	43,743	同上

系統	サブセット	実効件数	備考
BGE 系			
doc_doc			
独自 hard negatives (英語汎用)	wikipedia_hard_negatives_english	250,000	公開 500,000 行から上限サンプリング
独自 hard negatives (追加 query_doc / doc_doc / ドメイン)	agnews / gooqa / natural_questions / trivia_qa (計 80,000) 、 all_nli / coco_captions (計 40,000) 、 arxiv / codesearch / fineweb / pubmed (計 80,000)	200,000	各データセット 20,000
独自 hard negatives (長文, 11 言語)	wikipedia_hard_negatives_long_docs_{en,fr,es,it,de,pt,ar,ko,ja,ru,zh}	en 12,000 ・ 他言語 各 2,000 (計 32,000)	最大 8192 トークン。クエリは DeepSeek-V4 (deepseek-v4-flash) で生成

独自 hard negatives は公開データセット [bekko-embedding-v1-hard-negatives](#) (総 759,587 行・22 サブセット) から、上限付きサンプリングで計 482,000 行を用いた。バッチ設定は、短～中文 (最大 512 トークン) がバッチ 1152、長文 (最大 8192 トークン) がバッチ 192 (§3.5)。

D. query-crafter-multilingual

FineWeb2-IR 等の高速合成に用いた [query-crafter-multilingual](#) は、著者が開発・公開した多言語クエリ生成モデルである (Apache-2.0; 21 言語対応)。Qwen3-1.7B を基盤に、与えられた passage から「その passage を検索し得る短い検索指向テキスト」——自然文の質問・キーワードクエリ・FAQ・短い要約など——を高速に大量生成することを目的とする。

構築手順は3段階からなる。(1) 英語は fineweb-edu、非英語 20 言語は FineWeb2-HQ を上流コーパスとし、各 passage を最大1万字に切り詰めた上で 70-700 トークンの範囲 (トークン化が膨らむギリシャ語のみ上限を拡大) に絞る。(2) 教師 LLM (DeepSeek-V3 Chat) で、根拠性ルール——passage に根拠のない実体・日付・事実を発明しない、代名詞だけで対象を指さない、根拠が不十分なら **NONE** を返す等——を明文化した英語プロンプトにより 7 種の instruction type の参照出力を生成し、約 52.5 万行の SFT データセットを構築する (公開: [query-crafter-multilingual-sft-synth-500k](#), ODC-By)。(3) このデータで Qwen3-1.7B / Qwen3-4B を LoRA ファインチューニング (1.7B: rank 16 / alpha 32 ・ lr 1.5e-4 ・ 1 epoch ・ bf16) し、アダプタをマージして公開する。

7 種の instruction type: query (自然文の質問) / alt_query (同義の言い換え質問) / keywords (3語のキーワード) / synonym_keywords (同義語に言い換えた3語キーワード) / title (短いタイトル) / faq (答えが passage になる FAQ 形式の質問) / summary (中核事実のみの短い要約)。

モデル選定と品質: held-out テスト (約6,300行) で生成クエリと元 passage の意味的整合を BGE-M3 の cosine 類似度で測ると、1.7B 版 0.7808 / 4B 版 0.7820 / 教師 (DeepSeek-V3) 0.7811 と差はごく小さく、推論コストの低い 1.7B 版を公開版として採択した。この cosine 評価は検索指標 (recall / nDCG) そのものではなく意味的整合の offline proxy である点には注意が要る。既知の制約として、生成クエリは passage の表層に近い浅いクエリに偏りやすく、推論や背景知識を要する複雑なクエリは作りにくい——これが、文脈を読む高品質合成 (Qwen3.5-35B-A3B) と組み合わせる二系統設計 (§3.3) の根拠である。生成されたクエリには一定のノイズが混ざる——例えば、指定した語数を守らないキーワードや、元の文書に答えが書かれていない質問である。こうした傾向は出力の手動監査で特定し、学習データに入れる前にルールのフィルタ (生成失敗を示す出力や極端に長いクエリの除去など) で取り除いた。学習データ FineWeb2-IR の構築では、7 種のクエリタイプのうち基本の 5 タイプ (質問・キーワード・タイトル・FAQ・要約) を高い比率、言い換え系の 2 タイプを低い比率で混合して生成し、各クエリの hard negatives は BGE-M3 による近傍検索の上位候補から採掘した。

E. プルーニングパターン候補一覧

全候補 (L4 5種・L7 11種・L13 10種) とスコアを示す。スコアは §6.1 の probe——各候補をプルーニング後に MS MARCO で検索向けファインチューニングし、NanoBEIR-en (13タスク) の nDCG@10 平均で比較——による。参考として、同一 protocol でのプルーニングなし mmBERT-small (22層) は 0.5151。

L4 候補:

パターン	保持レイヤー	設計意図	NanoBEIR-en mean
4A	[0,1,2,3]	連続4層（深層 Global なし）	0.3329
4B (採用)	[0,1,2,18]	連続3層 + 遠方G18	0.4530
4C	[0,1,2,21]	連続3層 + 最終G21	0.4558
4D	[18,19,20,21]	深層のみ	0.4130
4E	[0,19,20,21]	先頭1層 + 深層3層	0.4408

L7 候補:

パターン	保持レイヤー	設計意図	NanoBEIR-en mean
7A	[0,1,2,3,4,5,6]	連続7層（G6 まで）	0.4291
7B	[0,1,2,3,4,5,9]	連続6層 + G9	0.4282
7C	[0,1,2,3,4,5,12]	連続6層 + G12	0.4204
7D (採用)	[0,1,2,3,4,5,18]	連続6層 + 遠方G18	0.4693
7E	[0,1,2,3,6,7,8]	初期4層 + 中間ブロック	0.4236
7F	[0,1,2,3,6,7,12]	初期4層 + 中間 + G12	0.4149
7G	[0,1,2,3,6,8,12]	初期4層 + 間引き + G12	0.4171
7H	[0,1,2,3,6,8,18]	初期4層 + 間引き + G18	0.4722
7I	[0,1,2,3,4,5,21]	連続6層 + 最終G21	0.4629
7J	[15,16,17,18,19,20,21]	深層のみ	0.4299
7K	[0,16,17,18,19,20,21]	先頭1層 + 深層6層	0.4589

L13 候補:

パターン	保持レイヤー	設計意図	NanoBEIR-en mean
13A	[0-12]	連続13層（G12 まで）	0.4553
13B	[0-11, 15]	連続12層 + G15	0.4576
13C (採用)	[0-11, 18]	連続12層 + 遠方G18	0.4964
13D	[0-11, 21]	連続12層 + 最終G21	0.4800
13E	[0-10, 12, 18]	連続11層 + G12 + G18	0.4904
13F	[0-9, 12, 15, 18]	連続10層 + G12/15/18	0.4791
13G	[0-8, 10, 12, 15, 18]	初期9層 + 間引き	0.4877
13H	[0-7, 9, 12, 15, 18, 21]	初期8層 + 間引き + G21	0.4810
13I	[9-21]	深層のみ	0.4307
13J	[0, 10-21]	先頭1層 + 深層12層	0.4806

なお採択パターンは各深さの厳密な最良ではない場合がある——L7 では非連続の 7H (0.4722) が 7D (0.4693) をわずかに上回るが、連続層の単純さを優先して 7D を採択した。L4 では 4C (G21) が 4B (G18) を誤差程度 (+0.003) 上回るが、L7・L13 で一貫する G18 優位と設計の統一性から 4B を採択した (§6.1)。

また、probe の 13 タスクには MS MARCO ファインチューニングと同ドメインの NanoMSMARCO が含まれる。感度分析として、これを除いた 12 タスク平均でも順位を確認した——4C 0.4530 > 4B 0.4486、13C 0.4950 (引き続き明確な最良) は不変で、7H と 7D は $0.4697 \approx 0.4698$ のほぼ同率 (順位反転) となる。すなわち L13 の結論は安定し、L4/L7 の僅差の順位は in-domain 成分に敏感である。

F. アブレーション実験の全結果

F.1 (prefix) はリリースで採用した「なし」を含む3条件の比較である。F.2 (QAT 有無) は prefix (`query: / passage:`) を付与する構成内の統制比較で、prefix 以外の第一段階データ・MRL 設定をリリースモデルと共有する。指標は学習中の開発用評価 (複数の Nano IR 評価の複合スコア, 0-1) で、final = 1 エポック終端の値、best = 学習全体での最大値。各設定 1 run。F.1-F.2 は §6.2-6.3 の、F.3 は §3.4 (MRL 重み) の一次データである。

F.1 prefix (a8m 第一段階フルデータ)

設定	final	best
query + passage	0.5778	0.5807
なし (リリースの設定)	0.5768	0.5820
query のみ	0.5752	0.5826

第二段階まで含めた prefix の影響は MMTEB Multilingual v2 でも確認した (§6.2)。この比較は、リリースモデルとは別に比較のためだけに prefix (query + passage) を付与して学習した未リリースの対照モデルを用い、第二段階データ・設定を揃えた統制比較である。Mean(Task) の差は 0.003 未満 (prefix なしが僅かに上) で、公式 2026-06-28 スナップショット (170 モデル) に対する one-add 比較の総合順位も両構成で同順位だった。またタスクタイプ別では、prefix を付与した構成に対し prefix なしは Retrieval の差が 1 ポイント未満にとどまる一方、Clustering で +6~+11 ポイント、MultilabelClassification で +6 ポイント程度高い傾向がある (こちらは prefix 以外の要素も含むモデルレベルの参考比較)。採択理由は「prefix 有無で結果に大幅な違いが見られず、prefix なしの方が単純で使いやすい」ことにある。

F.2 QAT 有無 (それ以外同一; MRL は両方に含む)

比較	QAT あり final / best	QAT なし final / best
a8m 第一段階 (L4)	0.5778 / 0.5807	0.5605 / 0.5608
a25m 第一段階 (L13)	0.5900 / 0.5937	0.5870 / 0.5955
L13: 第二段階まで通した比較 (両系譜とも開発時の同一 FT 構成)	0.6043 / 0.6054	0.5954 / 0.5992

なお F.2 の「第二段階まで通した比較」に用いた FT データは開発時の構成であり、リリースの FT レシピ (付録 C) とは異なる (両系譜には同一のものをを用いた)。

F.3 MRL 重み比較 (a8m 相当構成・20% データ proxy)

リリースと同一の dims [384,256,128,64] における MRL 重みの比較。リリース a8m の第一段階を模した L4 構成・第一段階データの 20% サンプル・QAT なしの proxy アブレーションで、最終チェックポイントを HAKARI-Bench dense 全体 (551 タスク, nDCG@10) で評価した。551 は評価時点の HAKARI-Bench dense タスク数 (同論文の構成) であり、リリース比較 (§5.2) に用いた公開リーダーボード pin (538 タスク) とはスナップショットが異なる——本表は proxy 内の相対比較のみに用い、§5.2 の数値と直接比較しない。

weights (w384/w256/w128/w64)	HAKARI all (551)	MNanoBEIR (182)	NanoMMTEB-v2 (18)
1.0 / 0.6 / 0.3 / 0.15	0.4968	0.4901	0.4602
1.0 / 0.5 / 0.2 / 0.1	0.4967	0.4903	0.4622
1.0 / 0.3 / 0.15 / 0.1 (リリース採用)	0.4955	0.4890	0.4555

設定間の差は 0.001–0.007 と小さく、重み選択の感度は低い。リリース採用値は 3 指標いずれでも 3 設定中最下位だが差は 0.001–0.007 と僅少であり、低次元の重みをやや強める設定が僅かに良い (proxy のため絶対値ではなく傾向を読む)。

G. 統一比較集合の詳細 — MNanoBEIR 言語別・長文タスク

§5 のサマリーの根拠となる、統一比較集合 (§4.2) の詳細スコア。MMTEB のタスクタイプ別スコアは本文の表2 に示した。MNanoBEIR・長文系は HAKARI-Bench の公開リーダーボード集計に基づく (量子化・次元 truncate を行わない標準構成; Bekko は a8m rev 953408de, a25m rev 8acf4b74, prefix なし)。

G.1 MNanoBEIR 言語別 (nDCG@10, 0–1; 14言語, リリースモデル・標準構成。太字 = bekko-a8m / bekko-a25m の行 (視認性のため; 列内最良の意味ではない))

MNanoBEIR の各 NanoBEIR-{lang} は NanoBEIR (Zeta Alpha) のコミュニティ機械翻訳版 (LightOn 8言語 / Liquid AI ja・ko / Serbian-AI-Society sr / sionic-ai th・vi) 由来で、本研究の著作作ではない (§2.4)。よって著者公開の HAKARI-Bench (self-citation) よりデータセットとしての独立性が高い。構成は、英語が sentence-transformers/NanoBEIR-en (zeta-alpha-ai 原本 13 セットの集約 repack)、翻訳 13 言語が上記各 org の NanoBEIR-{lang} で、全 14 言語が同一の 13 タスクを持つ 14×13 = 182 タスクの完全直積である。このため言語平均とタスク平均は一致する。

モデル	AP(M)	dims	avg	en	fr	de	it	es	pt	vi	sr	sv	no	ja	ko	ar	th
BM25	—	—	0.465	0.572	0.514	0.450	0.486	0.506	0.485	0.462	0.396	0.422	0.423	0.468	0.450	0.436	0.437
bekko-a8m	7.7	384	0.526	0.603	0.538	0.538	0.536	0.552	0.542	0.537	0.484	0.520	0.511	0.537	0.506	0.498	0.468

モデル	AP(M)	dims	avg	en	fr	de	it	es	pt	vi	sr	sv	no	ja	ko	ar	th
mE5-small	21.6	384	0.512	0.584	0.510	0.539	0.537	0.534	0.482	0.523	0.438	0.513	0.499	0.523	0.506	0.464	0.512
bekko-a25m	24.9	384	0.549	0.617	0.549	0.558	0.563	0.560	0.558	0.556	0.529	0.541	0.533	0.550	0.530	0.523	0.520
granite-97m-r2	28.3	384	0.505	0.592	0.535	0.535	0.522	0.532	0.522	0.517	0.441	0.504	0.484	0.497	0.495	0.461	0.434
mE5-base	86.0	768	0.531	0.603	0.541	0.542	0.545	0.537	0.524	0.540	0.482	0.536	0.538	0.522	0.517	0.491	0.520
harrier-270m	100.3	640	0.523	0.598	0.553	0.531	0.532	0.532	0.529	0.529	0.504	0.512	0.502	0.507	0.495	0.489	0.507
embgemma-300m	106.3	768	0.577	0.658	0.600	0.595	0.592	0.596	0.599	0.580	0.514	0.544	0.542	0.588	0.565	0.552	0.557
granite-311m-r2	110.3	768	0.543	0.613	0.562	0.554	0.555	0.560	0.561	0.548	0.508	0.543	0.522	0.551	0.531	0.499	0.492
gte-m-base	113.3	768	0.527	0.625	0.545	0.524	0.531	0.540	0.537	0.540	0.508	0.529	0.517	0.513	0.495	0.477	0.491
mE5-large	303.9	1024	0.560	0.628	0.556	0.578	0.565	0.562	0.539	0.570	0.522	0.559	0.556	0.557	0.557	0.522	0.567
arctic-l-v2.0	311.8	1024	0.584	0.652	0.590	0.603	0.596	0.590	0.589	0.583	0.567	0.585	0.586	0.566	0.571	0.536	0.567
bge-m3	311.8	1024	0.557	0.607	0.557	0.564	0.566	0.561	0.552	0.570	0.555	0.564	0.548	0.559	0.546	0.511	0.545

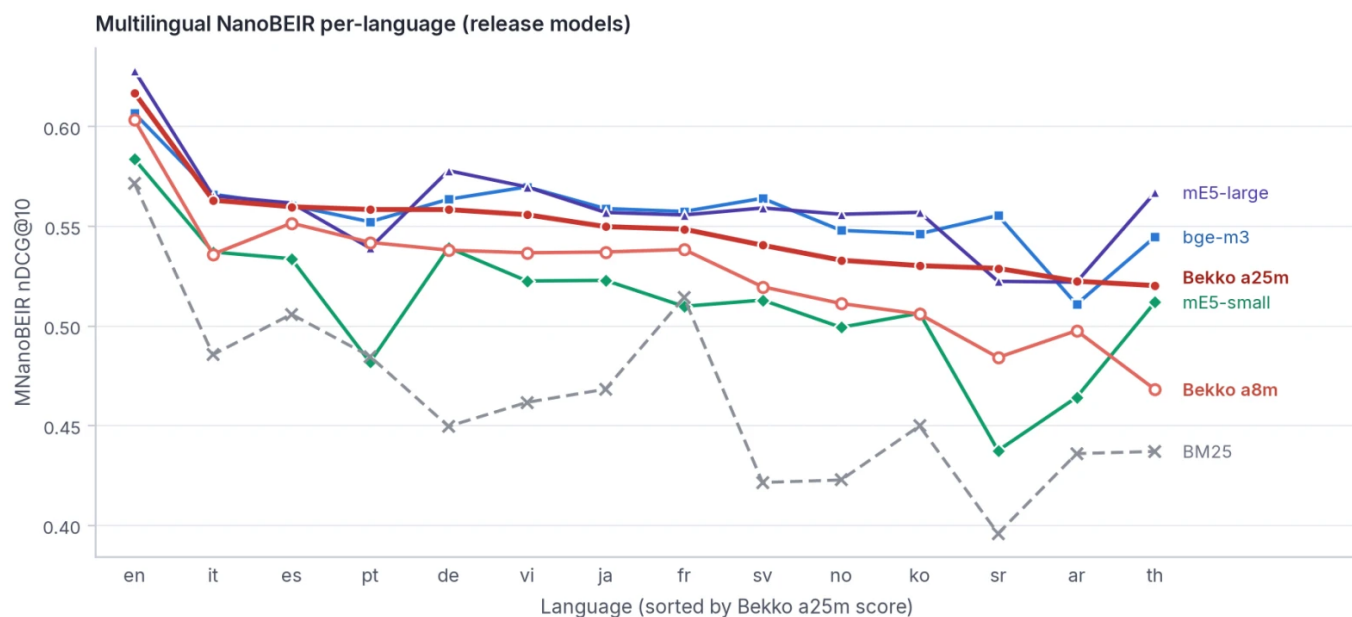


図9: MNanoBEIR 言語別スコア（抜粋 6 モデル; 全モデルは上表）。a25m（太い赤線）は言語間の落ち込みが小さく、AP 約12倍の bge-m3・mE5-large に概ね並走する（bge-m3 との最大差 0.026）。

要点: (1) a25m は 14 言語すべてで mE5-small・BM25・granite-97m を上回り、en・pt・ar では bge-m3 をも上回る。(2) bge-m3 との差は最大でも 0.026 (sr) で、a25m は言語間の落ち込みが小さい。(3) a8m は AP 7.7M ながら全言語で BM25 を上回り、de・it・th を除いて mE5-small 以上 (ko は三桁丸めで同値) である。(4) Bekko 両モデルの下位3言語は th・ar と sr (セルビア語) で、僅差で ko が続く——非ラテン文字言語が中心だがセルビア語も並ぶ。最弱言語はモデルごとに異なるが、Bekko にとってはこれら下位言語の改善が今後の課題である (§8)。

G.2 長文タスクの詳細 — NanoMLDR 言語別・NanoLongEmbed per-task (nDCG@10, 0-1。太字 = bekko-a8m / bekko-a25m の行 (視認性のため; 列内最良の意味ではない))

NanoMLDR 言語別 (13言語) :

モデル	AP(M)	dims	avg	en	de	es	fr	it	pt	ru	ar	hi	ja	ko	th	zh
BM25	—	—	0.740	0.725	0.714	0.944	0.912	0.888	0.950	0.866	0.760	0.318	0.759	0.687	0.387	0.703
bekko-a8m	7.7	384	0.545	0.480	0.487	0.781	0.806	0.689	0.806	0.653	0.485	0.296	0.555	0.412	0.251	0.382
mE5-small	21.6	384	0.392	0.395	0.276	0.540	0.567	0.558	0.608	0.478	0.357	0.241	0.368	0.265	0.186	0.258
bekko-a25m	24.9	384	0.571	0.517	0.516	0.818	0.826	0.719	0.840	0.658	0.492	0.348	0.565	0.429	0.319	0.380
granite-97m-r2	28.3	384	0.489	0.428	0.392	0.732	0.752	0.663	0.740	0.550	0.422	0.281	0.426	0.362	0.228	0.377
mE5-base	86.0	768	0.416	0.413	0.333	0.583	0.635	0.540	0.657	0.490	0.392	0.230	0.385	0.292	0.183	0.272
harrier-270m	100.3	640	0.519	0.457	0.419	0.781	0.772	0.684	0.766	0.598	0.443	0.319	0.502	0.409	0.262	0.341
embgemma-300m	106.3	768	0.505	0.448	0.463	0.707	0.746	0.643	0.737	0.587	0.415	0.301	0.519	0.357	0.288	0.350
granite-311m-r2	110.3	768	0.519	0.457	0.440	0.733	0.759	0.661	0.782	0.585	0.441	0.311	0.495	0.399	0.298	0.389
gte-m-base	113.3	768	0.681	0.548	0.600	0.908	0.890	0.842	0.906	0.775	0.660	0.531	0.714	0.640	0.413	0.430
mE5-large	303.9	1024	0.438	0.420	0.335	0.613	0.625	0.590	0.672	0.504	0.417	0.268	0.402	0.337	0.236	0.277
arctic-l-v2.0	311.8	1024	0.561	0.504	0.499	0.792	0.761	0.700	0.807	0.655	0.455	0.340	0.586	0.453	0.358	0.382

モデル	AP(M)	dims	avg	en	de	es	fr	it	pt	ru	ar	hi	ja	ko	th	zh
bge-m3	311.8	1024	0.662	0.511	0.592	0.891	0.880	0.814	0.883	0.746	0.602	0.497	0.698	0.586	0.480	0.428

NanoLongEmbed per-task (6タスク) :

モデル	AP(M)	dims	avg	2WikiMultihopQA	NarrativeQA	Needle	Passkey	QMSum	SummScreenFD
BM25	—	—	0.822	0.950	0.762	0.721	0.772	0.744	0.981
bekko-a8m	7.7	384	0.682	0.802	0.441	0.692	0.769	0.420	0.968
mE5-small	21.6	384	0.501	0.588	0.218	0.636	0.710	0.212	0.645
bekko-a25m	24.9	384	0.706	0.873	0.480	0.688	0.752	0.474	0.969
granite-97m-r2	28.3	384	0.659	0.823	0.456	0.629	0.720	0.416	0.908
mE5-base	86.0	768	0.518	0.575	0.229	0.633	0.710	0.252	0.708
harrier-270m	100.3	640	0.617	0.840	0.334	0.599	0.640	0.370	0.919
embgemma-300m	106.3	768	0.597	0.731	0.286	0.558	0.733	0.363	0.913
granite-311m-r2	110.3	768	0.695	0.872	0.521	0.612	0.726	0.473	0.965
gte-m-base	113.3	768	0.669	0.864	0.531	0.530	0.711	0.414	0.962
mE5-large	303.9	1024	0.505	0.586	0.253	0.545	0.675	0.232	0.737
arctic-l-v2.0	311.8	1024	0.632	0.770	0.416	0.518	0.734	0.392	0.965
bge-m3	311.8	1024	0.653	0.816	0.485	0.578	0.739	0.356	0.943

要点: (1) BM25 が両ベンチのほぼ全列で最上位——特に NanoMLDR の欧州語 (es 0.944 / pt 0.950) と NanoLongEmbed の 2WikiMultihopQA / SummScreenFD で顕著である。ただし NanoMLDR の hi (BM25 0.318) ・ th (0.387) では一部 dense が逆転する。(2) dense 内では、NanoMLDR は gte-m-base > bge-m3 > a25m の順 (bge-m3 の学習には長文書データが含まれる; [Chen et al., 2024](#))。NanoLongEmbed は a25m が平均で dense 1 位だが、タスク別では a8m が Needle / Passkey、a25m が 2WikiMultihopQA / QMSum / SummScreenFD で dense 最上位となり、NarrativeQA は gte-m-base が上回る。(3) mE5 系 (最大長 512) は長文タスク全般で大きく低く、長文対応はアーキテクチャ世代の差が最も現れる軸である。