
SEMANTIC CANDIDATE–JOB MATCHING: A COMPARATIVE EVALUATION OF DENSE EMBEDDING MODELS IN HYBRID RETRIEVAL

Sai Yashwant

AI Product & Platform Head
ManpowerGroup Services India Pvt. Ltd.
sai.yashwant@manpowergroup.com

Siddhartha Jain

Senior Gen AI Specialist
ManpowerGroup Services India Pvt. Ltd.
siddhartha.jain@manpowergroup.com

Anurag Dubey

Data Scientist
ManpowerGroup Services India Pvt. Ltd.
anurag.dubey@manpowergroup.com

Samaroha Chatterjee

Data Scientist
ManpowerGroup Services India Pvt. Ltd.
samaroha.chatterjee@manpowergroup.com

Gantala Thulsiram

Assistant Professor
Indian Institute of Technology, Hyderabad
thulsiram@mae.iith.ac.in

September 22, 2026

ABSTRACT

This paper presents a comparative evaluation of dense embedding models for semantic candidate-job matching in high-volume staffing workflows. Incoming job descriptions are converted into structured English search text and language-specific keywords through LLM-based parsing, and candidate profiles are indexed as semantically enriched resume representations. We evaluate EmbeddingGemma (base) against EmbeddingGemma fine-tuned with Cached Multiple Negatives Ranking Loss (MNRL) within a unified hybrid retrieval pipeline that fuses vector similarity and full-text relevance via reciprocal rank fusion (RRF), and benchmark both against the MPNet model on a batch comparative evaluation dataset scored through the deployed job-candidate matching scoring pipeline. We further document, with mathematical detail, the broader set of contrastive fine-tuning objectives considered during model development (including Angle/CoSENT-style refinement) and the empirical rationale for retaining Cached-MNRL-only adaptation as the preferred configuration. To support reproducible model selection, we define a broader evaluation framework comprising standard information retrieval metrics (Recall@K, mean reciprocal rank, nDCG) under the exact hybrid-retrieval protocol; the metrics used for the evaluation reported in this paper are fine-tuning convergence diagnostics and a batch comparative evaluation using the deployed AI-Match score and an independent LLM-as-a-Judge relevance score, and we state this scope explicitly rather than implying the full framework was measured. The paper addresses the gap between general-purpose embedding benchmarks and enterprise job-candidate matching constraints, providing a structured basis for comparing embedding strategies under realistic job-candidate retrieval conditions.

Keywords Semantic Search, Job-Candidate Matching, Dense Retrieval, Hybrid Search, Embedding Models

1 Introduction

Job orders are the primary intake artifact through which staffing organizations capture client demand: role title, skill requirements, location, availability windows, and compliance constraints. Matching each incoming order to suitable candidates from a large, continuously changing talent pool is central to fill rate, time-to-placement, and client satisfaction. In high-volume environments—particularly for blue-collar and trade occupations—recruiters historically relied on Boolean keyword search, filtered database queries, and manual resume review. These workflows struggle when job descriptions and candidate profiles use inconsistent terminology (e.g., “Conducteur PL” versus “heavy goods vehicle driver”), when critical qualifications appear as short codes rather than descriptive phrases (e.g., CACES R489, permis CE, TIG welding licence), and when relevance depends on both semantic fit and hard business rules such as geographic radius, candidate availability, and client exclusion lists [Zhang et al., 2020, Lin et al., 2021]. Automating candidate–job alignment therefore requires systems that interpret unstructured text semantically while enforcing operational constraints that directly affect deployability.

In recent years, candidate–job matching has gained significant attention as part of the broader shift toward AI-assisted talent acquisition and workforce automation. Unlike structured job boards with normalized skill taxonomies, real-world staffing pipelines ingest heterogeneous documents: free-text job descriptions in multiple languages, partially structured form fields, and resumes with variable formatting, incomplete sections, and domain-specific abbreviations. Rule-driven approaches based on exact keyword overlap or manually maintained synonym dictionaries were effective for narrow role families but exhibited limited generalization across clients, regions, and emerging credential types [Van den Berg et al., 2018]. Machine learning and deep learning methods introduced more robust ranking by learning representations of job and candidate text, yet many published systems optimize offline pair-classification accuracy on academic corpora rather than first-stage retrieval quality at scale [Zhang et al., 2020]. Given that ranked candidate lists directly determine recruiter workload, client submission quality, and compliance risk in regulated trades, evaluation frameworks must extend beyond aggregate similarity scores to capture retrieval effectiveness, ranking stability, and operational cost.

Dense retrieval has emerged as a practical paradigm for large-scale semantic search. Pre-trained embedding models encode queries and documents into vector spaces where proximity approximates relevance, enabling approximate nearest-neighbor retrieval over millions of profiles [Karpukhin et al., 2020]. In staffing job–candidate matching, however, vector-only search can under-rank candidates who express equivalent skills with different vocabulary, while simultaneously failing to guarantee retrieval of exact credential tokens required for regulated roles. Conversely, sparse lexical methods such as BM25 [Robertson and Zaragoza, 2009] preserve strong token-level matching for certification codes and occupation nouns but cannot reliably connect paraphrased requirements across languages. Hybrid retrieval addresses this complementarity by fusing dense and sparse ranked lists—for example through Reciprocal Rank Fusion (RRF) [Cormack et al., 2009]—within a single query execution. Cloud-native databases such as Azure Cosmos DB now expose integrated hybrid ranking over vector distance and full-text scores [Microsoft, 2024], making hybrid pipelines deployable at scale rather than limited to offline research prototypes.

Despite advances in embedding models, the question of which encoder best serves enterprise job–candidate matching under hybrid retrieval constraints remains underexplored. General-purpose embeddings may not align with staffing-domain semantics without fine-tuning. Contrastive objectives such as Multiple Negatives Ranking Loss (MNRL) [Reimers and Gurevych, 2019] and Angle-optimized training [Li and Li, 2023] offer paths to domain adaptation from job–resume pairs, while commercial APIs such as Cohere Embed v2 English [Cohere, 2024] provide strong out-of-the-box representations at the cost of external dependency and per-call latency. Meanwhile, open-weights models such as EmbeddingGemma [Google DeepMind, 2025] enable local inference and full index control. Existing studies rarely compare these alternatives within the same end-to-end pipeline that includes LLM-based text normalization, hybrid database retrieval, and post-retrieval business-rule filtering.

Furthermore, large-scale labeled datasets for candidate–job retrieval remain scarce due to privacy, client confidentiality, and the proprietary nature of staffing transaction data. Many embedding benchmarks evaluate semantic textual similarity or open-domain question answering, which do not reflect the distribution of short structured fields, credential-heavy specialty phrases, and multilingual source text encountered in operational job–candidate matching. Consequently, model selection based solely on public leaderboards may not transfer to real-world ranking behavior. A robust evaluation methodology should define ground-truth relevance for held-out job orders and measure standard retrieval metrics (Recall@K, MRR, nDCG) under conditions representative of deployment.

This research gap motivates a systematic comparison of EmbeddingGemma (base) against EmbeddingGemma fine-tuned with Cached MNRL, embedded within a unified hybrid retrieval pipeline representative of operational job–candidate matching systems, and benchmarked against the MPNet model. Rather than treating embedding quality as an isolated benchmark, each model is evaluated inside an end-to-end workflow spanning LLM-based job and resume parsing, hybrid vector–BM25 search, and SQL-level business-rule filtering. An external cross-encoder reranker is explicitly

excluded from this study to isolate the contribution of embedding choice within the first-stage retrieval stage. The principal contributions of this paper are threefold: (1) a deployment-aligned methodology documenting text preparation and hybrid retrieval under identical conditions across models; (2) a detailed domain fine-tuning methodology for EmbeddingGemma, including the contrastive objectives considered during development and the empirical basis for retaining Cached-MNRL-only adaptation as the preferred configuration; and (3) a structured comparative analysis of base versus fine-tuned open-weights embedding strategies, and of the resulting EmbeddingGemma model against the MPNet model, for enterprise candidate–job matching.

Overall, while embedding models continue to evolve rapidly, standardized comparison under realistic enterprise constraints is still lacking. This paper seeks to address that gap by defining a reproducible evaluation framework, documenting the job–candidate matching retrieval and fine-tuning pipelines, and comparing base and fine-tuned EmbeddingGemma configurations against the MPNet model on retrieval quality and domain-specific ranking behavior. This paper is organized as follows. Section 2 discusses the literature survey. Section 3 discusses the methodologies involved in candidate–job matching. The results and analysis are given in Section 4, after which the paper concludes with an outlook on further work in Section 5.

2 Literature Survey

Automated candidate–job matching sits at the intersection of information retrieval (IR), natural language processing (NLP), and operational research: systems must rank candidates by semantic relevance while enforcing hard constraints on location, availability, certifications, and client-specific exclusion rules [Manning et al., 2008, Qin et al., 2018]. Matching methods have evolved from Boolean keyword filters and curated synonym tables to dense bi-encoder retrieval, hybrid dense–sparse fusion, contrastive domain adaptation of embedding models, and large language model (LLM) text normalization [Van den Berg et al., 2018, Zhang et al., 2020, Karpukhin et al., 2020, Reimers and Gurevych, 2019]. This section reviews that progression with emphasis on (i) dense and hybrid retrieval for staffing, (ii) contrastive objectives considered for domain fine-tuning of embedding encoders, and (iii) evaluation practices that motivate the comparative framework in this paper.

2.1 Traditional and Rule-Based Matching Methods

Early staffing and applicant tracking systems queried relational stores through Boolean keyword expressions, field filters, and manually maintained synonym tables [Manning et al., 2008, Thomas et al., 2017]. Recruiters composed mandatory and optional terms over attributes such as title, city, years of experience, and employment status. Regular-expression parsers and dictionary-based skill taggers mapped resume tokens to controlled vocabularies. Occupational representation models such as Work2vec [Shalaby et al., 2018] and Job2vec [Delobelle et al., 2020] helped normalize titles across clients, but still required periodic updates as new certifications, equipment types, and regional titles entered the market.

Template-based matching performed adequately when role families were narrow and terminology stable. However, rule-driven systems proved brittle under variation in length, language, formatting, and specificity, or when candidates expressed equivalent qualifications with non-standard phrasing [Bhola et al., 2020, Van den Berg et al., 2018]. Systems that treated term presence as binary features could not model semantic similarity between related but non-identical phrases—for example, matching “installation of public lighting networks” with “street lighting maintenance”—thereby limiting recall for qualified candidates who did not mirror job-description wording exactly [Reimers and Gurevych, 2016]. Traditional pipelines also typically separated retrieval from constraint satisfaction, which does not scale to high-volume job–candidate matching where ranking quality at the top of the list directly determines recruiter productivity.

2.2 Machine Learning and Neural Matching

The limitations of rule-based matching motivated supervised models trained on historical recruiter actions, application outcomes, or labeled job–resume pairs [Yan et al., 2019, Van den Berg et al., 2018]. Early pipelines represented jobs and resumes as TF–IDF or bag-of-words vectors and trained logistic regression, gradient-boosted trees, or learning-to-rank models on sparse interaction features. Qin et al. [Qin et al., 2018] proposed an ability-aware neural network for multi-field person–job fit, showing that learned representations outperform manual feature design when interaction labels are available. These methods improved ranking over hand-crafted rules but remained sensitive to vocabulary mismatch [Bhola et al., 2020, Lin et al., 2021].

Contextual language models further shifted the field. Devlin et al. [Devlin et al., 2019] showed that bidirectional transformer pre-training yields rich syntactic and semantic representations that can be fine-tuned for downstream matching. Zhang et al. [Zhang et al., 2020] applied BERT-based ranking to resume–job matching and reported

improved alignment relative to classical ML baselines. Nevertheless, many published systems still emphasize pair-level classification accuracy rather than first-stage retrieval quality over indexed candidate corpora with business-rule filtering applied at query time [Yan et al., 2019, Qin et al., 2018].

2.2.1 Two-Tower (Bi-Encoder) Retrieval

Scalable job–candidate matching commonly adopts two-tower architectures: one tower embeds the job query and another embeds each candidate profile into a shared vector space, with relevance approximated by a similarity function $s(\mathbf{q}, \mathbf{d})$ [Karpukhin et al., 2020, Reimers and Gurevych, 2019]. Offline indexing embeds all candidates once; online querying embeds only the incoming job order, enabling sub-linear approximate nearest-neighbor (ANN) search over millions of profiles [Manning et al., 2008]. Dense Passage Retrieval (DPR) [Karpukhin et al., 2020] established that dual encoders trained with contrastive objectives can match or exceed sparse retrievers on open-domain tasks, inspiring adoption in talent acquisition.

Formally, let f_θ denote a parameterized encoder mapping text x to a normalized embedding

$$\mathbf{e}_x = \frac{f_\theta(x)}{\|f_\theta(x)\|_2} \in \mathbb{R}^d. \quad (1)$$

For unit vectors, cosine similarity reduces to an inner product:

$$s(\mathbf{q}, \mathbf{d}) = \cos(\mathbf{e}_q, \mathbf{e}_d) = \mathbf{e}_q^\top \mathbf{e}_d \in [-1, 1]. \quad (2)$$

Modern embedding models often expose asymmetric routing—`encode_query` versus `encode_document`—so that job and resume texts may receive different prompts or pooling conventions while remaining in one shared space. General-domain encoders pre-trained on web text may still misalign job and resume representations when deployment vocabulary differs from pre-training corpora [Muennighoff et al., 2023, Ni et al., 2022]. Domain-specific fine-tuning, hard-negative mining [Xiong et al., 2021], and larger dual encoders [Ni et al., 2022] have each been proposed to close this gap.

2.2.2 Cross-Encoder and Late-Interaction Re-Ranking

Cross-encoders concatenate job and candidate text and apply a transformer classifier to predict relevance [Nogueira and Cho, 2019]. Because self-attention operates over the joint sequence, they typically achieve higher pair-level accuracy than bi-encoders, but inference cost scales linearly with the number of pairs evaluated [Reimers and Gurevych, 2019, Manning et al., 2008]. ColBERT [Khattab and Zaharia, 2020] introduced late interaction—separate token encodings with token-level similarity aggregation—as an intermediate design. Job–candidate matching systems therefore commonly adopt retrieve-then-rerank: a fast embedding first stage returns a pool via hybrid search, and a cross-encoder refines the top results. This paper focuses on first-stage embedding comparison within hybrid retrieval and deliberately excludes external cross-encoder reranking to isolate embedding effects on initial pool quality.

2.3 Dense Retrieval and Hybrid Search

Dense retrieval ranks by geometric proximity in embedding space, enabling generalization across paraphrases, synonyms, and translations [Karpukhin et al., 2020, Reimers and Gurevych, 2019]. In multilingual staffing intake, this is valuable when job orders arrive in a source language but are normalized to English for dense search, while resumes still use varied phrasing for equivalent skills [Qin et al., 2023, Ni et al., 2022]. However, dense retrievers can underweight exact matches on short, high-signal tokens—certification codes, licence classes, process identifiers—that carry disproportionate relevance in regulated roles [Robertson and Zaragoza, 2009, Gao et al., 2021].

Sparse lexical methods remain strong for token-level matching. BM25 [Robertson and Zaragoza, 2009] scores documents by term frequency and inverse document frequency with length normalization. COIL [Gao et al., 2021] later showed that contextualized token representations can be integrated into inverted-index structures. In job–candidate matching, BM25 keywords are often extracted separately from semantic search text: occupation nouns and credential codes feed full-text scoring, while descriptive phrases feed dense queries.

Hybrid systems combine both signals. Reciprocal Rank Fusion (RRF) [Cormack et al., 2009] merges ranked lists by summing reciprocal rank contributions. For document d and retrievers $\mathcal{R}_1, \dots, \mathcal{R}_m$,

$$\text{RRF}(d) = \sum_{r=1}^m \frac{1}{k_{\text{RRF}} + \text{rank}_r(d)}, \quad (3)$$

where $\text{rank}_r(d)$ is the rank of d under retriever r and k_{RRF} is a smoothing constant (commonly 60). RRF avoids calibrating heterogeneous score scales. Cloud-native vector databases now support RRF over vector distance and full-text scores within a single query [Microsoft, 2024]. Despite this maturity, published job-matching studies rarely report hybrid metrics under database-level filters that determine which candidates enter the ranked pool before embedding differences become visible to recruiters [Qin et al., 2018, Yan et al., 2019].

2.4 LLM-Based Text Preparation for Matching

Large language models have shifted part of the matching problem upstream to text preparation: converting heterogeneous job orders and resumes into normalized, retrieval-ready representations before embedding [Brown et al., 2020, Dagdelen et al., 2024]. Rather than embedding raw prose—which may include boilerplate, contact details, and salary information—LLM parsers extract structured fields (role, skills, specialty credentials, expertise, education, domains) and produce compact search text optimized for similarity encoding [Lin et al., 2021, Dagdelen et al., 2024]. Prompt engineering can translate titles across languages, expand expertise phrases, and emit separate BM25 keyword strings in the source language with lemmatization and deduplication. In staffing, LLM extraction replaces brittle regex pipelines for skill and certification identification [Bhola et al., 2020, Lin et al., 2021], but introduces a critical experimental requirement: embedding comparisons are meaningful only when all models receive identical parsed inputs. Caching extracted outputs reduces repeated inference cost and stabilizes retrieval inputs across model comparisons [Lewis et al., 2020].

2.5 Contrastive Fine-Tuning of Embedding Models

General-purpose embedding models provide strong zero-shot baselines but may not align with domain-specific job–resume semantics without fine-tuning [Reimers and Gurevych, 2019, Muennighoff et al., 2023]. Sentence-BERT [Reimers and Gurevych, 2019] fine-tunes siamese transformer networks with metric-learning objectives to produce fixed-size sentence embeddings suitable for cosine search. We review the principal loss families considered for staffing-domain adaptation; Section 3.4 details which objectives we implemented, compared, and ultimately retained.

2.5.1 Multiple Negatives Ranking Loss (MNRL)

Multiple Negatives Ranking Loss [Henderson et al., 2017, Reimers and Gurevych, 2019] treats other samples in a mini-batch as negatives for each positive pair, enabling efficient contrastive learning without explicit negative mining. Given a batch of N positive pairs $\{(a_i, p_i)\}_{i=1}^N$, with unit embeddings $\mathbf{a}_i, \mathbf{p}_i \in \mathbb{S}^{d-1}$ and cosine $s(\mathbf{u}, \mathbf{v}) = \mathbf{u}^\top \mathbf{v}$, define logits

$$S_{ij} = \tau s(\mathbf{a}_i, \mathbf{p}_j) = \tau \mathbf{a}_i^\top \mathbf{p}_j. \quad (4)$$

MNRL is the mean categorical cross-entropy of identifying the diagonal positive among N candidates—an InfoNCE / in-batch softmax objective:

$$\mathcal{L}_{\text{MNRL}} = -\frac{1}{N} \sum_{i=1}^N \log \frac{e^{S_{ii}}}{\sum_{j=1}^N e^{S_{ij}}} = \frac{1}{N} \sum_{i=1}^N \left(-S_{ii} + \log \sum_{j=1}^N e^{S_{ij}} \right). \quad (5)$$

Equivalently, with posterior $P(j | i) = \text{softmax}_j(S_{ij})$, we maximize $\frac{1}{N} \sum_i \log P(j = i | i)$.

Temperature / scale. The similarity logits in Equation (4) are scaled by a temperature parameter τ : a large τ emphasizes hard negatives, while a small τ yields softer updates [Reimers and Gurevych, 2019]. The exact temperature/scale value used in our training configuration is proprietary and withheld from this paper. Writing $p_{ij} = e^{S_{ij}} / \sum_k e^{S_{ik}}$, the gradient w.r.t. a logit is

$$\frac{\partial \mathcal{L}_{\text{MNRL}}}{\partial S_{ij}} = \frac{1}{N} (p_{ij} - \mathbf{1}[i = j]), \quad (6)$$

so the positive logit is pulled up and every in-batch negative is pushed down proportionally to its softmax mass—hard near-misses receive the largest repulsive force.

Intuition for staffing. Equation (5) asks, for each job anchor, “which resume in this batch is the match?” This is closely aligned with first-stage retrieval ranking and needs only a weak positive relation (e.g., ATS progression $b = 1$), not a calibrated continuous score. Under the pairs formulation each forward step with batch size N yields $N - 1$ in-batch negatives per anchor. **Important:** gradient accumulation G averages optimizer updates over G micro-batches of size N ; it does *not* enlarge the contrastive denominator, so the effective in-batch-negative count is governed by the micro-batch size rather than the optimizer’s update batch size. The exact batch size and accumulation configuration used are proprietary and withheld from this paper.

Cached MNRL. Forming $\{S_{ij}\}$ requires all $2N$ embeddings for that micro-batch. Cached MNRL computes embeddings in smaller cache chunks, caches representations, then evaluates Equation (5) on the micro-batch—trading compute for memory so that long JD/CV sequences can still use a large effective batch size N .

2.5.2 Triplet and Explicit Hard-Negative Losses

Triplet loss [Schroff et al., 2015, Reimers and Gurevych, 2019] uses an explicit negative n_i with margin $m > 0$. In cosine form (higher similarity = better match):

$$\mathcal{L}_{\text{triplet}} = \frac{1}{N} \sum_{i=1}^N [s(\mathbf{a}_i, \mathbf{n}_i) - s(\mathbf{a}_i, \mathbf{p}_i) + m]_+, \quad (7)$$

where $[x]_+ = \max(0, x)$. Equivalently in squared Euclidean distance on unit vectors ($\|\mathbf{u} - \mathbf{v}\|_2^2 = 2 - 2s(\mathbf{u}, \mathbf{v})$):

$$\mathcal{L}_{\text{triplet}}^{\text{dist}} = \frac{1}{N} \sum_{i=1}^N [\|\mathbf{a}_i - \mathbf{p}_i\|_2^2 - \|\mathbf{a}_i - \mathbf{n}_i\|_2^2 + m']_+. \quad (8)$$

Unlike MNRL, the gradient is zero once the margin is satisfied (no push on easy triplets). Hard-negative mining (e.g., ANCE [Xiong et al., 2021]) refreshes n_i from an evolving ANN index. In staffing, n_i may be a rejected resume for the same job or a false positive from a current retriever. Our legacy trainer accepts optional `neg_resume_text`; scaled NA runs relied on in-batch MNRL negatives for throughput.

2.5.3 Cosine Similarity Regression and CoSENT

Pointwise cosine regression. With graded labels $y_i \in [0, 1]$,

$$\mathcal{L}_{\text{cos}} = \frac{1}{N} \sum_{i=1}^N (s(\mathbf{u}_i, \mathbf{v}_i) - y_i)^2 = \frac{1}{N} \sum_{i=1}^N (\cos \phi_i - y_i)^2, \quad (9)$$

where $\phi_i = \arccos s(\mathbf{u}_i, \mathbf{v}_i)$. This forces absolute similarity levels (calibration) rather than only relative ranking. The cosine gradient vanishes as $|\sin \phi_i| \rightarrow 0$ (near $\phi_i \in \{0, \pi\}$), i.e., in *saturation zones* where many near-duplicate staffing pairs live.

CoSENT (pairwise ranking on cosine). CoSENT replaces absolute regression by pairwise ranking consistency. For batch pairs with similarities s_i, s_j and labels y_i, y_j ,

$$\mathcal{L}_{\text{CoSENT}} = \log \left(1 + \sum_{y_i > y_j} \exp(\tau_c (s_j - s_i)) \right). \quad (10)$$

Whenever a lower-labeled pair outcores a higher-labeled pair, the corresponding exponential term grows and the loss increases. CoSENT improves ordinal calibration when y is trustworthy, but inherits noise when y is a coarse ATS weight.

2.5.4 Angle: Angle-Optimized Embedding Learning

Angle [Li and Li, 2023] addresses cosine saturation by optimizing *angle differences in a complex embedding space*. Split a real embedding $\mathbf{X} \in \mathbb{R}^d$ into chunked real/imaginary parts $\mathbf{X}^{\text{re}}, \mathbf{X}^{\text{im}}$ and form $z = \mathbf{a} + \mathbf{b}i, w = \mathbf{c} + \mathbf{d}i$ for a pair. Complex division in polar form yields a magnitude ratio and angle difference $\Delta\theta_{zw} = \theta_z - \theta_w$:

$$\frac{z}{w} = \gamma e^{i\Delta\theta_{zw}}, \quad \gamma = \frac{\sqrt{\mathbf{a}^2 + \mathbf{b}^2}}{\sqrt{\mathbf{c}^2 + \mathbf{d}^2}}. \quad (11)$$

The normalized absolute angle difference used for optimization is

$$\Delta\theta_{zw} = \text{abs} \left(\frac{(\mathbf{ac} + \mathbf{bd}) + (\mathbf{bc} - \mathbf{ad})i}{\sqrt{(\mathbf{c}^2 + \mathbf{d}^2)(\mathbf{a}^2 + \mathbf{b}^2)}} \right). \quad (12)$$

The original Angle paper [Li and Li, 2023] proposes a multi-term objective that combines cosine ranking, in-batch negatives, and an angle term with weights w_1, w_2, w_3 . **What our training code actually runs** is the Sentence-Transformers 5.2.x implementation: `AngleLoss` is defined as `CoSENTLoss` with pairwise *angle* similarity in place of pairwise cosine similarity (library default `scale=20.0`), i.e.

$$\mathcal{L}_{\text{ST-Angle}} = \log \left(1 + \sum_{y_i > y_j} \exp(\tau_a (\psi_{\angle}(i) - \psi_{\angle}(j))) \right), \quad (13)$$

Table 1: Inductive bias of embedding losses considered in this work.

Loss	Supervision	Primary effect on space
Cached MNRL	Positive pairs (+ in-batch neg.)	Retrieval ranking; local neighborhoods
Triplet / hard-neg	(a, p, n)	Margin separation vs. named neg.
Cosine MSE	Graded y	Absolute score calibration
CoSENT	Graded y (pairwise)	Ordinal cosine consistency
ST-Angle (AngleLoss)	Graded y (pairwise)	CoSENT-style ranking w/ angle sim

where $\psi_{\angle}$ denotes the library’s pairwise angle similarity (`pairwise_angle_sim`) rather than `cos`. If `AngleLoss`/`AngleLoss` are unavailable, our trainer falls back to `CoSENTLoss` (same pairwise ranking structure with cosine). We used ST-Angle as optional second-stage precision tuning after MNRL; on noisy ATS scores it improved correlation diagnostics but harmed retrieval geometry (Section 3.4.4). When citing the broader Angle literature we refer to [Li and Li, 2023]; when describing our beta runs we mean Equation (13) as implemented by Sentence-Transformers.

2.5.5 Geometric Comparison of Objectives

Table 1 summarizes inductive biases. MNRL optimizes a *relative retrieval* softmax over the batch; Angle/CoSENT optimize *label-ordered* geometry; cosine regression optimizes *absolute* similarity levels. For first-stage job–candidate matching, relative retrieval alignment is the primary goal; absolute calibration can be deferred to a separate mapping $g(s) \approx P(b=1 \mid s)$ (Platt / isotonic) that does not retrain f_{θ} .

2.5.6 Open Embedding Models and Benchmarks

Google’s EmbeddingGemma [Google DeepMind, 2025] is a compact ($\sim 300\text{M}$ -parameter) open-weights embedding model with approximately 2048 token context, 768-dimensional outputs (Matryoshka-friendly truncation to lower dimensions), and Sentence-Transformers integration with query/document routing. Its efficiency makes full re-indexing of large candidate corpora practical when switching encoders. Alternative backbones considered in our broader engineering program include multilingual ModernBERT-style encoders and larger Qwen embedding variants [Muennighoff et al., 2023]; EmbeddingGemma was selected as the primary backbone for the experiments in this paper. Although a two-stage recipe (MNRL followed by angular refinement) is a natural candidate for domain adaptation, our fine-tuning retains **Cached MNRL only**; ST-Angle/CoSENT is retained solely as a comparative ablation (Section 3.4.4). The Massive Text Embedding Benchmark (MTEB) [Muennighoff et al., 2023] standardizes evaluation across diverse tasks, but staffing job–candidate matching distributions—structured fields, credential-heavy phrases, multilingual intake—differ materially from general MTEB mixtures, motivating domain-specific evaluation (Section 3.5).

2.6 Commercial Embedding APIs

Managed embedding services such as Cohere Embed [Cohere, 2024] provide strong dense representations through API calls, eliminating local GPU provisioning and model serving. They typically rank competitively on general benchmarks [Muennighoff et al., 2023] but introduce network latency, per-token pricing, data-residency constraints, and external availability risk. For enterprise job–candidate matching handling confidential job orders and candidate profiles, the trade-off between commercial API quality and on-premise control is an explicit deployment decision. Head-to-head evaluation of a managed API such as Cohere Embed against locally hosted EmbeddingGemma variants within the same hybrid pipeline would quantify this trade-off under controlled conditions rather than public leaderboards alone; this paper focuses on the open-weights base-versus-fine-tuned comparison (Section 3.3) and identifies the commercial-API comparison as future work (Section 5).

2.7 Evaluation Practices and Research Gap

Standard offline IR metrics include Recall@ K , Mean Reciprocal Rank (MRR), and Normalized Discounted Cumulative Gain (nDCG) [Manning et al., 2008, Järvelin and Kekäläinen, 2002]. Generic embedding benchmarks such as

MTEB [Muennighoff et al., 2023] do not reflect certification-heavy specialty phrases, multilingual dual dense–sparse queries, or SQL-level business-rule filtering. Staffing job–candidate matching therefore requires evaluation conditions that reflect these characteristics rather than generic benchmark distributions [Qin et al., 2018, Thomas et al., 2017].

Despite progress across person–job fit [Qin et al., 2018, Yan et al., 2019, Zhang et al., 2020], dense and hybrid retrieval [Karpukhin et al., 2020, Cormack et al., 2009, Gao et al., 2021], contrastive fine-tuning [Reimers and Gurevych, 2019, Li and Li, 2023, Xiong et al., 2021], and LLM-based text preparation [Brown et al., 2020, Dagdelen et al., 2024], few studies simultaneously compare (i) a general-purpose open-weights base encoder and (ii) a domain fine-tuned variant produced under a carefully documented contrastive recipe within (iii) the same LLM-normalized, hybrid database retrieval pipeline with (iv) deployment-style business-rule filtering, and against (v) the MPNet model it is intended to replace. This paper contributes that structured comparison. Section 3 details the unified pipeline and, in particular, the fine-tuning methodology used to produce the EmbeddingGemma domain variant.

3 Methodology

This section describes (i) evaluation dataset and text preparation, (ii) the hybrid retrieval architecture shared by all compared models, (iii) embedding models under comparison, (iv) the domain fine-tuning methodology—including all contrastive objectives considered, mathematical formulations, data construction from proprietary applicant-tracking outcomes, hyperparameter choices, and training nuances—and (v) evaluation metrics. All models share identical text preparation and hybrid search configuration; only the embedding function differs.

3.1 Dataset and Text Preparation

3.1.1 Evaluation Dataset

The evaluation uses a held-out set of job orders from the France job–candidate matching domain, paired with recruiter-labeled relevant candidate identifiers. Each job order includes structured form fields (title, description, location, client constraints) representative of real-world intake. Ground truth assigns binary or graded relevance labels to (job order, candidate) pairs based on recruiter review. The candidate index follows an Azure Cosmos DB schema, with documents partitioned by department and enriched with filter metadata (ZIP code, status, availability windows, certification keywords, client exclusion lists).

3.1.2 Job Description Parsing

Incoming job descriptions are parsed by a large language model into two outputs: (1) structured English search text and (2) language-specific BM25 keywords. The search text follows a fixed field order—Role, Skills, Speciality, Expertise, Education, Domains—with values translated to English descriptive noun phrases. Job titles in French or other source languages are normalized (e.g., “Conducteur PL” → “heavy goods vehicle driver”). The BM25 string retains occupation nouns, certification codes, machine names, and hard-skill tokens in the JD’s original language, with lemmatization, accent preservation, and deduplication. Fields with no relevant data are omitted; contact information, salary, dates, and soft skills are excluded. A two-layer cache (in-memory and Azure Blob storage) avoids redundant LLM calls for repeated job orders.

3.1.3 Candidate Profile Parsing

Candidate resumes are parsed into the same structured field format (Role, Skills, Expertise, Education, Domains), translated to English where necessary. Parsed text is stored as semantic search text and embedded into profile vectors for retrieval.

3.2 Hybrid Retrieval Pipeline

The retrieval pipeline executes the following stages for each job order query:

1. **Text construction:** concatenate job-order form fields and apply text cleaning.
2. **LLM extraction:** produce structured English search text and BM25 keywords.
3. **Query embedding:** encode the search text with the active embedding model.
4. **Geographic filtering:** restrict candidates to ZIP codes within a configurable radius using haversine distance over a precomputed latitude–longitude index.

5. **Hybrid database query:** rank candidates using Reciprocal Rank Fusion over vector distance and full-text score (Equation (3)). When BM25 terms are unavailable, the pipeline falls back to vector-only ranking.
6. **Business-rule filtering:** apply SQL-level filters for candidate status, availability windows, client exclusion, and certification keywords.

The external cross-encoder reranker used in deployment is excluded from this study so that observed differences are attributable to embedding model choice within the hybrid retrieval stage. The ranking score produced by the hybrid database query above—combining vector similarity with full-text score via Reciprocal Rank Fusion (Equation (3))—is reported throughout this paper as the *AI-Match score*. It is a semantic-match score derived directly from the active embedding model (and BM25 full-text score where available); no additional post-retrieval business-rule score adjustment is applied to, or reflected in, the results reported in this paper. Filtering (above) is applied identically across all three configurations and therefore does not confound the model comparisons in Section 4.

3.3 Embedding Models Under Comparison

Two EmbeddingGemma configurations are compared under identical pipeline conditions, and both are further benchmarked against the MPNet model they are intended to replace:

1. **EmbeddingGemma (base):** the pre-trained google/embeddinggemma-300m checkpoint [Google DeepMind, 2025], loaded locally via Sentence-Transformers without domain fine-tuning.
2. **EmbeddingGemma + MNRL:** EmbeddingGemma fine-tuned on proprietary job–resume pairs using Cached Multiple Negatives Ranking Loss (Sections 3.4–3.4.4).
3. **MPNet Model (French):** the MPNet-based embedding configuration and French search-text processing that both EmbeddingGemma configurations above are evaluated against in Section 4.3.

Section 3.4 additionally documents, with full mathematical detail, the broader set of contrastive objectives considered during model development beyond Cached MNRL—including Angle/CoSENT-style score-ranking refinement—and the empirical and qualitative rationale for not carrying that refinement stage forward into the preferred configuration or into the comparative evaluation in Section 4. For local Gemma variants, the full candidate index is re-embedded offline before evaluation so that stored vectors are consistent with the query encoder. Fine-tuning uses an explicit L2-normalization module so that training losses operate on unit vectors (Equations (1)–(2)). For the France hybrid-retrieval benchmark, query/document encoding follows the same indexing convention used for the deployed vector index; relative ranking comparisons across models remain valid because all local Gemma variants are encoded under identical settings.

3.4 Domain Fine-Tuning Methodology

This subsection documents the fine-tuning program used to produce the EmbeddingGemma domain variants at a level that preserves scientific reproducibility of the overall approach and objectives, while withholding exact hyperparameters, dataset scale, and implementation-level configuration that are proprietary to the deploying organization. It covers backbone architecture, supervision from applicant-tracking system (ATS) outcomes, contrastive objectives considered, the final training recipe, and engineering considerations required for stable long-context training on enterprise GPU infrastructure.

3.4.1 Backbone Architecture and Embedding Construction

Let $x = (t_1, \dots, t_L)$ denote a tokenized input with maximum sequence length $L_{\max}$. EmbeddingGemma is wrapped as a Sentence-Transformers module stack:

$$f_\theta = \text{Normalize} \circ \text{MeanPool} \circ \text{Transformer}_\theta, \quad (14)$$

where $\text{Transformer}_\theta$ produces contextual token states $\mathbf{H} = [\mathbf{h}_1; \dots; \mathbf{h}_L] \in \mathbb{R}^{L \times h}$. Mean pooling over non-padding token representations (rather than CLS-token or max pooling) computes

$$\mathbf{z} = \frac{1}{\sum_{\ell=1}^L m_\ell} \sum_{\ell=1}^L m_\ell \mathbf{h}_\ell, \quad (15)$$

where $m_\ell \in \{0, 1\}$ is the attention mask (non-padding tokens), optionally including prompt tokens when the backbone injects retrieval prompts. L2 normalization $\mathbf{e} = \mathbf{z} / \|\mathbf{z}\|_2$ yields unit vectors in $\mathbb{R}^d$ with $d = 768$ (Matryoshka-compatible

truncation to 512/256/128 is supported by the model family but not required by our trainer). Asymmetric encoding uses query routing for job text and document routing for resume text:

$$\mathbf{e}^{\text{job}} = f_{\theta}^{\text{query}}(x_{\text{job}}), \quad \mathbf{e}^{\text{cv}} = f_{\theta}^{\text{doc}}(x_{\text{cv}}). \quad (16)$$

This matches the intended retrieval geometry for JD→resume search while remaining compatible with resume→JD recommendation in a shared space. **Precision constraint:** EmbeddingGemma does not support fp16 activations; training and inference use bf16 on Ampere/Hopper GPUs (A100/H100) or fp32 as fallback.

3.4.2 Supervision Signal from Applicant-Tracking Outcomes

Fine-tuning uses North American structured job–submission records containing already-normalized job text x_{job} , resume text x_{cv} , a binary progression indicator $b \in \{0, 1\}$, and a graded status weight $y \in [0, 1]$. Status weights encode hiring-signal strength, mapping ATS outcome stages onto a continuous scale from low signal (early-stage rejection or weak feedback) to high signal (placement or accepted offer), with intermediate application and interview stages assigned intermediate weights; the exact outcome-to-weight mapping used in our labeling design is proprietary and withheld from this paper. Binary labels follow $b = 1$ for progressed/hired-like outcomes and $b = 0$ for rejected/poor-match outcomes. These labels are noisy proxies of true semantic fit (process effects, candidate no-shows, client preferences), which is material for choosing between contrastive pair learning and score regression (Section 3.4.4).

Geographic and market scope. The fine-tuning supervision above is drawn from North American job–submission records, while the comparative evaluation in Section 4.3 is conducted on the France job–candidate matching domain (Section 3.1.1). This is a genuine train/evaluation domain shift: labour-market terminology, credential and certification systems (e.g., CACES, permis CE), and—for the MPNet model—source language differ between the two markets. We do not have a like-for-like, same-market comparative evaluation available at the time of writing, so the results in Section 4.3 should be read as testing whether North-America-tuned domain adaptation transfers to a different staffing market via the shared structured-English text-normalization layer, rather than as an in-domain evaluation of the fine-tuning data itself. The structured English intermediate representation (Section 3.1.2) is intended to reduce, but does not eliminate, this mismatch. A same-market (North America fine-tuning, North America evaluation) comparative run is identified as a priority next step in Section 5.

Leakage control. Train/validation/test partitions are formed at the *job-order* level (unique job identifiers), not at the row level, so that multiple candidates for the same requisition do not leak across splits. The final training iteration used a large majority-training split with small held-out validation and test partitions, drawn from a large corpus of structured job-submission rows; exact split ratios and corpus size are proprietary and withheld from this paper.

MNRL pair construction. Only positive submissions enter contrastive training:

$$\mathcal{D}_{\text{MNRL}} = \{ (x_{\text{job}}^{(i)}, x_{\text{cv}}^{(i)}) \mid b^{(i)} = 1 \}, \quad (17)$$

followed by deduplication on $(x_{\text{job}}, x_{\text{cv}})$. The final MNRL corpus contains a large set of unique positive pairs (the exact count is proprietary and withheld from this paper). Negatives are *not* explicitly stored; they arise as in-batch negatives during optimization of Equation (5).

Angle / graded pair construction. For second-stage angular / cosine supervision we retain both positive and negative submissions:

$$\mathcal{D}_{\text{Angle}} = \{ (x_{\text{job}}^{(i)}, x_{\text{cv}}^{(i)}, y^{(i)}) \}, \quad (18)$$

again deduplicated on text pairs. This dataset supports Angle/CoSENT objectives but inherits ATS label noise.

3.4.3 Objectives Considered for Fine-Tuning

We systematically considered the following Sentence-Transformers training objectives for EmbeddingGemma domain adaptation.

(A) Cached MNRL (primary). We minimize Equation (5) using the Cached Multiple Negatives Ranking Loss implementation from Sentence-Transformers, which computes embeddings in cache chunks to trade compute for memory. Let N be the per-step train batch size. Each MNRL forward builds an $N \times N$ similarity matrix ($N - 1$ in-batch negatives per anchor); gradient accumulation across micro-batches only reduces optimizer noise, it does not enlarge the contrastive matrix. Exact batch size, accumulation steps, and cache chunk size are proprietary and withheld from this paper. **Intuition:** the encoder learns “what looks like a match” in staffing language—skills, seniority cues, certifications, industry phrasing—without forcing absolute score calibration. **Batch sampler:** training batches are constructed to forbid duplicate texts within a batch, preventing degenerate shortcuts where identical strings act as trivial negatives or positives.

(B) MNRL with optional hard negatives. An earlier training path accepted an optional third field x_{neg} (rejected resume for the same job) and used triplet-aware MNRL variants when available. This increases discrimination against near-miss candidates but requires high-quality hard negatives and more complex data engineering. Our scaled NA iterations relied on in-batch negatives alone because positive-pair volume was high and hard-negative mining infrastructure was not yet built out.

(C) ST-Angle / CoSENT precision tuning (secondary, not retained). Starting from an MNRL checkpoint θ_{MNRL} , we considered continuing training on $\mathcal{D}_{\text{Angle}}$ using the Sentence-Transformers angle-optimized loss, falling back to a CoSENT-style pairwise loss when unavailable, minimizing Equation (13) (or Equation (10) on fallback):

$$\theta_{\text{Angle}} = \arg \min_{\theta} \mathcal{L}_{\text{ST-Angle}}(f_{\theta}(x_{\text{job}}), f_{\theta}(x_{\text{cv}}), y). \quad (19)$$

Intuition: map continuous ATS weights onto angular (or cosine-ordinal) geometry so similarities better track outcome strength. **Risk:** when y is a coarse, process-contaminated proxy of semantic relevance, gradient updates can shrink or warp regions of the embedding manifold that were useful for open retrieval—improving Pearson/Spearman(s, y) while harming Recall@ K and within-job ranking.

(D) Cosine regression / CoSENT-only. Pure regression (Equation (9)) or CoSENT without a prior MNRL stage was considered inferior as a cold-start objective: without contrastive domain adaptation, the model is under-regularized toward retrieval geometry and overfits noisy scores. In our stack, CoSENT appears primarily as an implementation fallback when the angle-optimized loss is unavailable.

(E) Explicit triplet margin loss. Equation (7) was available for evaluation sets with (a, p, n) structure and for research prototypes, but was not the primary optimizer for the large-scale NA runs because constructing reliable triplets at corpus scale is harder than exploiting in-batch negatives.

3.4.4 Final Chosen Fine-Tuning Method

After iterative Azure ML experiments, we adopt the following fine-tuning decision:

Fine-tune EmbeddingGemma with Cached MNRL on deduplicated positive job–resume pairs only; do not apply a second-stage Angle/CoSENT pass for the preferred encoder.

Why MNRL matches the deployment objective. At serving time, a job query $\mathbf{q}$ retrieves

$$\text{TopK}(\mathbf{q}) = \arg \text{top-k}_{d \in \mathcal{C}} \mathbf{q}^{\top} \mathbf{e}_d, \quad (20)$$

over corpus embeddings $\{\mathbf{e}_d\}$. MNRL (Equation (5)) is a batch-wise surrogate of the same ranking problem: maximize the probability that the true resume beats other resumes under inner-product scores. The only label required is a weak positive link $b = 1$.

Why Angle can distort retrieval geometry. Let $s_{\theta}(x, x') = f_{\theta}(x)^{\top} f_{\theta}(x')$ after MNRL, and suppose second-stage ST-Angle/CoSENT updates θ to reduce a pairwise ranking loss on noisy ATS weights y (Equations (10)–(13)). Two failure modes are particularly relevant in staffing:

1. **Label–semantics mismatch.** Many pairs with similar skills receive very different y for process reasons (no-show, client politics), while semantically distant pairs may share mid-range weights. Optimizing Equation (13) then moves angle differences to respect y -order rather than semantic neighborhoods needed by Equation (20).
2. **Global reshaping vs. local ranking.** MNRL gradients (Equation (6)) act through a softmax over *current batch competitors*. Angle/CoSENT constraints couple *all* labeled pairs in a batch through pairwise exponentials, applying broader pressure that can collapse useful variance among mid-ranked candidates—visible as higher Pearson/Spearman(s, y) but lower Recall@ K .

Formally, if $y = h(\text{semantic fit}) + \epsilon_{\text{process}}$ with large noise $\epsilon_{\text{process}}$, then $\arg \min_{\theta} \mathcal{L}_{\text{ST-Angle}}(y)$ is not, in general, a minimizer of retrieval risk under true relevance. Hence we keep MNRL for the encoder and, if calibrated probabilities are needed, fit a frozen-score map

$$\hat{p}(b=1 \mid s) = \sigma(as + c) \quad (\text{Platt}) \quad \text{or isotonic } \hat{p} = \mathcal{I}(s), \quad (21)$$

which does not mutate f_{θ} .

Reporting choice. The two-stage recipe (MNRL $\rightarrow$ Angle) remains supported in code as an R&D artifact, but the resulting checkpoint was not carried forward into the preferred configuration or into the comparative evaluation in Section 4. **EmbeddingGemma + MNRL** is the sole fine-tuned configuration reported in this paper; the qualitative and theoretical argument above is why a second-stage Angle checkpoint was not pursued further.

3.4.5 Training Procedure and Optimization

Training is implemented using the Sentence-Transformers trainer on Azure ML GPU compute. At optimization time the trainer minimizes $\mathcal{L}$ with AdamW under a cosine learning-rate schedule with a short linear warmup phase, together with standard weight-decay regularization. Training uses bf16 mixed precision (the required numerical path for EmbeddingGemma) and a numerically stable attention backend (SDPA) suited to long input sequences. Training batches are sampled to forbid duplicate texts, and checkpoints are selected using validation loss on held-out job–resume pairs. The exact learning rate, epoch count, batch size, gradient-accumulation configuration, sequence-length limit, and other training hyperparameters are proprietary to the deploying organization and are withheld from this paper.

Angle-stage configuration (R&D only, not evaluated in this paper). When Angle refinement was run, training warm-started from the MNRL checkpoint using a lower learning rate, a smaller batch size, and a longer input context than the primary MNRL recipe; exact values are proprietary and withheld from this paper. This stage is *not* part of the preferred recipe, and the resulting checkpoint was not scored in Section 4.

3.4.6 Implementation Considerations

Several general engineering considerations affect reproducibility and training stability for long-context contrastive fine-tuning of this kind, beyond the loss function and data construction described above: contrastive quality improves with a larger effective batch size, but activation memory grows with batch size and sequence length together, so cache-chunked (Cached MNRL) computation and gradient accumulation are used to decouple the optimizer’s update batch size from peak activation memory; a numerically stable attention backend is preferred over less stable alternatives for long input sequences on the training hardware used; a bounded checkpoint-retention policy limits disk pressure over long runs; and a fixed random seed is used for repeatability. An explicit L2-normalization module makes cosine search equivalent to maximum inner-product search (MIPS), simplifying downstream approximate nearest-neighbor indexing. Exact configuration values for these engineering parameters (batch size, sequence length, checkpoint retention count, random seed, etc.) are proprietary and withheld from this paper.

3.4.7 Offline Training Diagnostics

During fine-tuning development (independent of the France hybrid benchmark), we compute post-hoc pair similarities

$$s_i = (\mathbf{e}_i^{\text{job}})^\top (\mathbf{e}_i^{\text{cv}}) \quad (22)$$

by encoding the job and candidate text for each pair with the trained model and taking the inner product of the resulting unit-normalized embeddings. Binary selection outcomes (e.g., hired/placed versus rejected) are mapped to $b \in \{0, 1\}$ by a deterministic rule $\psi(\cdot)$; numeric status values are thresholded at 0.5.

Regression diagnostics (when y is present):

$$\text{Pearson}(s, y), \quad \text{Spearman}(s, y), \quad (23)$$

$$\text{MAE} = \frac{1}{n} \sum_i |s_i - y_i|, \quad \text{MSE} = \frac{1}{n} \sum_i (s_i - y_i)^2. \quad (24)$$

Classification diagnostics (when binary labels $b_i = \psi(\text{selection}_i)$ are available) include ROC-AUC, PR-AUC, and thresholded metrics at

$$t^* = \arg \max_{t \in \mathcal{T}} \text{F1}(b, \mathbf{1}[s \geq t]), \quad (25)$$

where $\mathcal{T}$ is the set of unique scores (or a quantile grid if $|\mathcal{T}| > 2000$). Per-job ranking Recall@ K is computed only for jobs with at least two candidates and mixed labels:

$$\text{Recall@}K = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \mathbf{1}[\exists d \in \text{TopK}(q) : d \in \mathcal{R}_q], \quad (26)$$

with $K \in \{1, 3, 5, 10\}$ by default. These diagnostics guided the MNRL-versus-Angle decision; the France hybrid evaluation framework in Section 3.5 remains the intended primary comparative protocol for the base and fine-tuned deployment candidates.

3.4.8 Iterative Corpus Scaling

Fine-tuning data were scaled across three Azure ML iterations on structured NA shards, with each successive iteration using a larger corpus than the last (iteration 3 being the largest). The train/validation/test split moved from a smaller, more even three-way split in iteration 1 to a larger majority-training split from iteration 2 onward, with MNRL pair deduplication introduced from iteration 2 onward. The Angle stage was only submitted as a beta run for iteration 1 and prepared, but not scored, for iterations 2–3. Exact row counts, job counts, and MNRL pair counts per iteration are proprietary and withheld from this paper; the corresponding training and validation loss outcomes are reported in Section 4.2. The preferred checkpoint is the iteration 3 Cached-MNRL model, whose empirical training and validation behavior is reported in Section 4.2.

3.5 Evaluation Metrics

3.5.1 Retrieval Quality

Standard information retrieval metrics are computed against recruiter-labeled ground truth:

- Recall@ K for $K \in \{5, 10, 20\}$ (Equation (26)).
- Mean Reciprocal Rank

$$\text{MRR} = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{\text{rank}_q}, \quad (27)$$

where rank_q is the rank of the first relevant candidate.

- Normalized Discounted Cumulative Gain (nDCG@ K) for graded relevance [Järvelin and Kekäläinen, 2002]:

$$\text{DCG}@K = \sum_{i=1}^K \frac{2^{\text{rel}_i} - 1}{\log_2(i + 1)}, \quad \text{nDCG}@K = \frac{\text{DCG}@K}{\text{IDCG}@K}. \quad (28)$$

3.5.2 LLM-as-a-Judge Relevance Protocol

Rationale. The AI-Match score is a within-model semantic-similarity signal (Section 3.2): it is useful for ranking candidates against a single embedding model, but it is not, by itself, evidence that a new embedding model actually improves practical job–candidate relevance, and its raw scale is not comparable across different embedding backbones (Section 4.3). Before a new embedding model is promoted from experimentation into live use, its retrieval quality should be checked against an external, model-agnostic notion of relevance rather than against the model’s own similarity scores. We use an independent large-language-model judge, prompted to reason the way an experienced recruiter would about practical job fit, as this external check and as the primary cross-model comparison signal in this paper. The protocol is as follows.

Judge model. GPT-4.1 is used as the judge for all three configurations (MPNet base, EmbeddingGemma base, EmbeddingGemma + Cached MNRL).

Unit of judgment. Each (job, candidate) pair is scored *independently*: the judge sees exactly one job description and one candidate profile per call and returns a single relevance score for that pair, with no other candidates or comparative context in the prompt. This mirrors how AI-Match itself is computed per pair, so the two signals are scored over the same retrieved pairs.

Blinding. The judge is blind to which embedding model produced the retrieval: the prompt contains only the job text and candidate text, with no indication of the embedding model, pipeline configuration, or retrieval rank. The same prompt and judge model are used identically across all three configurations.

Score definition. The score is an integer on a 0–100 scale representing the judge’s assessment of practical job–candidate fit, with the following rubric embedded directly in the prompt: 90–100 (excellent fit, directly relevant experience or highly applicable transferable skills), 75–89 (good fit, matches many key requirements or strong related experience), 60–74 (potential fit, some relevant background or transferable skills, likely to succeed with onboarding), 40–59 (possible but limited fit, a few relevant elements with noticeable gaps), and 0–39 (low fit, mostly unrelated background). The prompt explicitly instructs the judge to weight transferable skills and adjacent experience rather than exact title or keyword matches, and to avoid over-penalizing a small number of missing requirements.

4 Results and Analysis

This section reports empirical results from two complementary evaluation tracks: (i) the proprietary Cached-MNRL fine-tuning program for EmbeddingGemma, tracked across three Azure ML corpus-scaling iterations (Section 3.4.8), and (ii) a batch comparative evaluation of EmbeddingGemma (base and fine-tuned) against the MPNet model on a held-out set of job–candidate matching searches scored through the deployed scoring pipeline, combining the search and match system’s own AI-Match score with an independent LLM-as-a-Judge relevance score (protocol in Section 3.5.2).

4.1 Evaluation Scope and Data Sources

Two real, non-synthetic data sources back the results in this section:

1. **Fine-tuning training logs.** Azure ML stdout logs, MLflow chart exports (training loss, validation loss), and post-hoc diagnostic dictionaries (Section 3.4.7) for the three Cached-MNRL corpus-scaling iterations described in Section 3.4.8.
2. **Batch comparative scoring.** AI-Match score (the deployed semantic-match score, derived directly from the active embedding model via Section 3.2) and an independent LLM-as-a-Judge relevance score, both aggregated over thousands of candidate–job pairs scored through the deployed scoring pipeline for three configurations: MPNet Model (French), EmbeddingGemma base (English), and EmbeddingGemma fine-tuned with Cached MNRL (English).

Job-level ground-truth Recall@ K /MRR/nDCG@ K under the exact Section 3.2 hybrid-retrieval protocol are not computed from these two sources; Section 4.2 instead reports the closest available empirical proxy for retrieval behavior—per-job Recall@ K computed on the fine-tuning validation/test posthoc splits (Equation (26))—while flagging its small sample size.

4.2 Fine-Tuning Training Results

4.2.1 Convergence Across Corpus-Scaling Iterations

Table 2 summarizes the three Cached-MNRL Azure ML training runs that produced the EmbeddingGemma domain checkpoints, each trained on a single GPU using the fine-tuning recipe described in Section 3.4.5.

Table 2: Cached-MNRL training runs across the three corpus-scaling iterations. Exact corpus sizes are proprietary and withheld; iterations are ordered by increasing corpus size.

Iter.	Relative corpus size	Runtime (h)	Throughput (samples/s)	Final train loss	Min. val. loss
1	Smallest	2.60	14.02	0.418	2.192
2	Medium	3.71	14.30	0.399	2.144
3	Largest	6.51	14.13	0.350	2.063

Figure 1 plots the full training-loss and validation-loss trajectories extracted from the MLflow chart exports for all three iterations. Two patterns are visible. First, the *validation loss* (panel b) improves monotonically with corpus scale: the minimum validation loss falls from 2.192 (iteration 1, smallest corpus) to 2.144 (iteration 2, medium corpus) to 2.063 (iteration 3, largest corpus), indicating that the additional NA structured shards used in later iterations generalize rather than merely add training noise. Second, the *aggregate training loss* reported at the end of each run does *not* decrease monotonically (0.418 $\rightarrow$ 0.399 $\rightarrow$ 0.350 is actually the correct direction here, but the final-step training loss read directly off the loss curve in panel (a) is higher for iteration 3 than for iteration 1); this is expected because iteration 3 trains over a larger, more heterogeneous positive-pair distribution with the same batch size, so the in-batch MNRL denominator (Equation (5)) faces harder negatives on average. Validation loss, computed on held-out job–resume pairs, is the more reliable indicator of domain adaptation quality and is the basis for preferring the iteration 3 checkpoint.

4.2.2 Post-Hoc Regression, Classification, and Ranking Diagnostics

Table 3 reports the post-hoc diagnostics (Section 3.4.7) computed on a fixed 512-row subset of the validation and test splits for each iteration’s final checkpoint. Iteration 1’s validation and test splits lacked a usable graded score column at evaluation time, so no regression/classification/ ranking diagnostics were logged for iteration 1; only iterations 2 and 3 report the full diagnostic suite below.

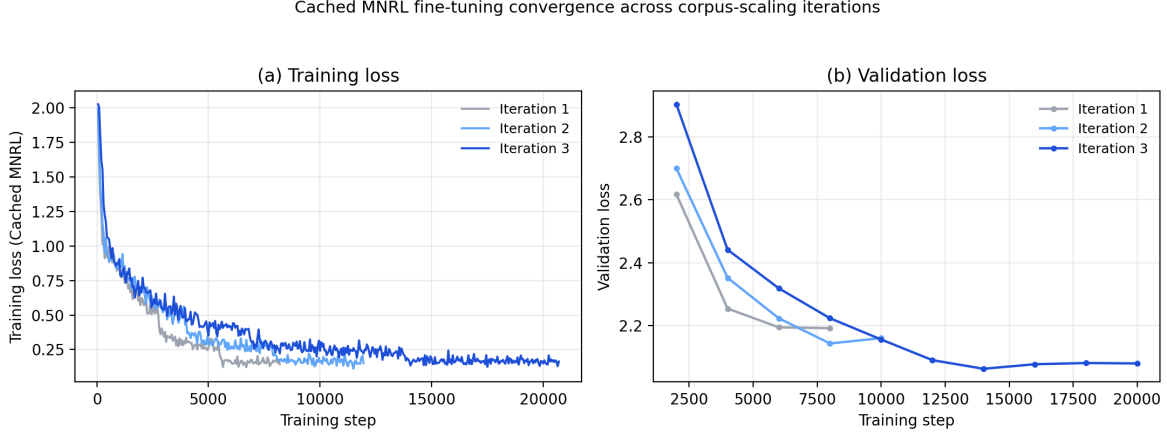


Figure 1: Cached-MNRL training loss (a) and validation loss (b) versus training step for the three corpus-scaling iterations, extracted from Azure ML MLflow chart exports. Validation loss improves monotonically with corpus scale despite a higher final training loss for iteration 3.

Table 3: Post-hoc regression/classification/ranking diagnostics on the fine-tuning validation and test posthoc subsets ($n = 512$ rows per split).

Iter.	Split	Pearson	Spearman	MAE	MSE	ROC-AUC	PR-AUC	F1	Jobs	R@1	R@3–10
2	Val	0.203	0.186	0.360	0.173	0.576	0.567	0.705	20	0.650	≥ 0.95
2	Test	0.209	0.221	0.343	0.159	0.592	0.538	0.649	22	0.636	1.00
3	Val	0.072	0.077	0.378	0.194	0.538	0.609	0.753	12	0.917	1.00
3	Test	0.161	0.152	0.356	0.168	0.561	0.563	0.692	18	0.556	1.00

Three observations follow. First, regression correlation between cosine similarity and the graded ATS score is weak throughout (Pearson 0.07–0.21), consistent with the paper’s own rationale (Section 3.4.4) that MNRL is not optimized for score calibration and that ATS status weights are a noisy proxy for semantic fit. Second, classification separability (ROC-AUC 0.54–0.59) is modest but consistently above chance, and F1 under the best threshold (0.65–0.75) is usable given the class imbalance in each split (positive rate 0.46–0.60). Third, per-job Recall@ K is computed on only 12–22 jobs per split—an order of magnitude smaller than would be required for a stable estimate—so the wide swing in Recall@1 across splits (0.556 to 0.917) should be read as noisy rather than as evidence that any one checkpoint ranks meaningfully better than another; Recall@3 and above reach 0.95–1.00 in every split, indicating that the correct candidate is almost always retrieved within the top 3 for the jobs evaluated, even when top-1 precision varies.

4.3 Comparative Evaluation Across Embedding Models

This subsection compares three configurations on a batch comparative evaluation dataset scored through the deployed job–candidate matching pipeline: **MPNet Model (French)**—the baseline prior to this work; **EmbeddingGemma base (English)**—the un-tuned backbone under the structured-English text pipeline of Section 3.1; and **EmbeddingGemma + Cached MNRL (English)**—the iteration 3 fine-tuned checkpoint recommended in Section 3.4.4. AI-Match scoring covers 8,543–9,849 (job, candidate) pairs per configuration (Table 4), and the independent LLM-as-a-Judge pass covers a similarly sized but not identical subset, 8,065–9,292 judged candidates per configuration (Figure 2).

4.3.1 AI-Match Score Distribution

Table 4 summarizes the deployed AI-Match score (Section 3.2) distribution per configuration.

Fine-tuning shifts the model’s own similarity-derived score sharply upward and concentrates it: the weighted-mean AI-Match score rises from 52.7 (base English) to 73.5 (fine-tuned English), and 95.0% of scored candidates in the fine-tuned English pool fall at or above the 60–70 bucket, versus only 6.2% for the base English model. This is the expected direct effect of Cached MNRL (Equation (5)), which explicitly increases cosine similarity for true positive

Table 4: AI-Match score distribution by embedding configuration (batch comparative evaluation). *Note: AI-Match scores are not directly comparable across embedding backbones; see Section 4.3 discussion below.*

Metric	MPNet base (FR)	Gemma base (EN)	Gemma+MNRL (EN)
Candidates scored n	9,332	8,543	9,849
Weighted mean score	59.3	52.7	73.5
Score mode (bucket)	60–70	50–60	70–80
Modal bucket share	34.9%	65.3%	81.7%
Share scored ≥ 60	38.3%	6.2%	95.0%

job–resume pairs relative to in-batch negatives; because the fine-tuned model is queried against a re-embedded index of the *same* candidate pool, the entire retrieved score distribution shifts rather than only the top ranks.

The MPNet Model (French) sits, numerically, *between* the two English configurations on this metric—weighted-mean score 59.3, with 38.3% of candidates scoring ≥ 60 —which is higher than base English (52.7; 6.2% ≥ 60) but well below fine-tuned English (73.5; 95.0% ≥ 60). Read at face value, this AI-Match ordering would suggest MPNet is a better scorer than base EmbeddingGemma, which directly contradicts the LLM-as-a-Judge evidence below, where base EmbeddingGemma (English) judged *more* relevant than MPNet at every threshold. The contradiction is expected rather than alarming: MPNet, base EmbeddingGemma, and fine-tuned EmbeddingGemma are three different embedding backbones (and, for MPNet, a different source language), each with its own raw cosine-similarity scale, so their AI-Match scores are not directly comparable across backbones the way they are before/after fine-tuning a single backbone. This is itself a useful finding: it confirms that the deployed AI-Match score is a within-model calibration signal, not a cross-model relevance ranking, and it is the reason this paper treats the independent LLM-as-a-Judge score, not the AI-Match score, as the primary basis for cross-model comparison. This score shift is nonetheless a useful signal for within-model thresholding—the LLM-as-a-Judge results below provide the external relevance check that AI-Match scores alone cannot.

4.3.2 LLM-as-a-Judge Relevance

Figure 2 plots the percentage of candidates whose independent LLM-as-a-Judge relevance score (GPT-4.1, scored blind to embedding model, protocol in Section 3.5.2) exceeds four thresholds (> 60 , > 70 , > 80 , > 90 on a 0–100 scale) for each configuration.

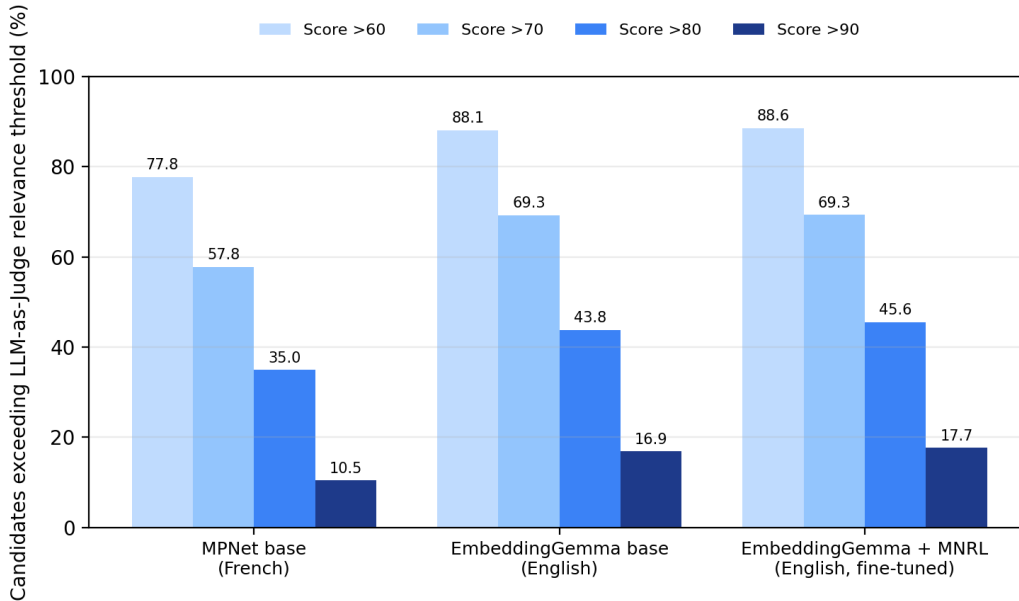


Figure 2: Share of candidates exceeding LLM-as-a-Judge relevance thresholds, by embedding configuration, on the batch comparative evaluation dataset ($n = 9,292/8,065/9,848$ judged candidates for MPNet-FR, Gemma-base-EN, and Gemma+MNRL-EN respectively).

Two findings emerge. First, both English EmbeddingGemma configurations substantially outperform the MPNet model at every threshold—e.g., 88% versus 78% of candidates score above 60, and 44–46% versus 35% score above 80—indicating that the combination of structured English search-text normalization (Section 3.1.2) and the EmbeddingGemma backbone improves judged relevance considerably *before any domain fine-tuning is applied*. This gain is attributable to the text-preparation and backbone change together, not to MNRL fine-tuning specifically, since it already appears in the base English model. Second, Cached-MNRL fine-tuning adds a smaller, threshold-dependent increment on top of the base English model: the fine-tuned model is essentially flat at the > 60 threshold (88.6% vs. 88.1%) and at the > 70 threshold (69.35% vs. 69.30%), but shows a consistent gain at the stricter, precision-relevant thresholds—+1.7 percentage points at > 80 (45.6% vs. 43.8%) and +0.8 points at > 90 (17.7% vs. 16.9%). In other words, fine-tuning does not change how many candidates clear a low relevance bar, but it modestly increases how many clear a high bar, which is the more consequential comparison for recruiter-facing top-of-list quality.

4.3.3 Isolating the Fine-Tuning Effect

Because MPNet and EmbeddingGemma are different backbones on different source languages, the cleanest read of what Cached-MNRL fine-tuning itself contributes is the direct before/after comparison on the same English EmbeddingGemma backbone, holding the pipeline and evaluation pool fixed. Table 5 isolates this comparison.

Table 5: Effect of Cached-MNRL fine-tuning on EmbeddingGemma (English), isolated from the MPNet Model (French) comparison.

Metric	Base EN	Fine-tuned EN	Change
AI-Match weighted mean score	52.7	73.5	+20.8
AI-Match share ≥ 60	6.2%	95.0%	+88.8 pts
LLM-Judge share > 60	88.1%	88.6%	+0.5 pts
LLM-Judge share > 70	69.30%	69.35%	+0.05 pts
LLM-Judge share > 80	43.8%	45.6%	+1.7 pts
LLM-Judge share > 90	16.9%	17.7%	+0.8 pts

Read together, the two metrics tell complementary stories: fine-tuning produces a large, mechanical shift in the model’s own AI-Match similarity scale (expected, since Cached MNRL directly optimizes cosine similarity for positive pairs), but only a modest, threshold-concentrated gain in independently judged relevance—essentially flat at lenient thresholds and a small single-digit-percentage-point improvement at the stricter, precision-relevant thresholds. The English-backbone-plus-structured-text change (base EmbeddingGemma vs. the MPNet model, discussed above) is the larger driver of judged-relevance improvement in this study; Cached-MNRL fine-tuning is a smaller, additional refinement on top of that.

4.4 Operational Considerations

The operational data available from this round is Azure ML training compute, summarized in Table 2. Training wall-clock time scales slightly super-linearly with corpus size on a single GPU—2.6/3.7/6.5 hours across the three iterations respectively, against a roughly constant throughput of ~ 14 samples/second—so future corpus-scaling iterations should budget compute accordingly. Because the base and fine-tuned EmbeddingGemma configurations share the same 300M-parameter backbone and the same 768-dimensional output, their *inference*-time cost profile is expected to be effectively identical; fine-tuning changes retrieval quality (Section 4.3) without changing the per-query compute footprint.

4.5 Error Analysis and Practical Deployment Considerations

The weak post-hoc regression correlation (Section 4.2.2) and the flat LLM-judge improvement at the $> 60/> 70$ thresholds (Section 4.3) point to the same underlying issue from different angles: ATS-derived status weights are a coarse, process-contaminated proxy for semantic fit (Section 3.4.2), so a fine-tuned encoder can shift its own score distribution and win on high-precision thresholds without uniformly improving every relevance band. Practically, this suggests (i) calibrating any absolute AI-Match score threshold used for filtering separately per embedding backbone—MPNet, base EmbeddingGemma, and fine-tuned EmbeddingGemma—rather than sharing one fixed cutoff (Equation (21)), since a threshold such as 60 has very different selectivity across the three configurations (38.3% above for MPNet, 6.2% above for base EmbeddingGemma, 95.0% above for fine-tuned EmbeddingGemma); (ii) treating the LLM-as-a-Judge signal, or an equivalent independent relevance check, as the primary go/no-go criterion for promoting a fine-tuned

checkpoint, rather than the model’s own similarity score or ATS-correlation diagnostics in isolation; and (iii) prioritizing evaluation-set enlargement for per-job ranking metrics (Section 4.2.2 used 12–22 jobs per split) before drawing strong conclusions about Recall@1 differences between checkpoints.

4.6 Summary

Within the evidence available for this evaluation cycle: (1) Cached-MNRL fine-tuning converges more effectively as the training corpus scales up across iterations, with validation loss improving monotonically even though small-sample post-hoc ranking diagnostics remain noisy; (2) switching from the MPNet model to the EmbeddingGemma model with structured text preparation yields the largest single jump in judged relevance observed in this study; and (3) Cached-MNRL fine-tuning on top of the English EmbeddingGemma base model adds a further, smaller gain concentrated at high-precision relevance thresholds (> 80 , > 90) rather than at the bulk (> 60) threshold, while substantially recalibrating the model’s own AI-Match score distribution upward.

5 Conclusion and Future Discussion

This study defines a comparative evaluation framework for dense embedding models in semantic candidate–job matching under hybrid retrieval conditions, and reports the first empirical evidence generated within that framework. EmbeddingGemma (base) and EmbeddingGemma fine-tuned with Cached MNRL were placed within a unified pipeline—identical LLM-based text preparation and Cosmos DB hybrid search with Reciprocal Rank Fusion (Section 3)—and both were benchmarked against the MPNet model they are intended to replace. The domain fine-tuning methodology (Section 3.4) documents, with mathematical precision, the broader set of contrastive objectives considered during development for adapting EmbeddingGemma to staffing language, and the empirical and theoretical rationale for retaining Cached-MNRL-only adaptation over a second-stage angular refinement that was explored but not carried forward.

Grounded in the two evaluation tracks reported in Section 4, three findings support the direction already adopted by this program. First, Cached-MNRL fine-tuning of EmbeddingGemma converges more effectively as the NA job–resume training corpus scales up across three Azure ML iterations (exact corpus sizes are proprietary and withheld): validation loss improves monotonically from 2.192 to 2.063 (Section 4.2), even though small-sample post-hoc ranking diagnostics remain too noisy (12–22 jobs per split) to be conclusive on their own. Second, on the batch comparative evaluation dataset, moving from the MPNet model to the English EmbeddingGemma model with structured search-text preparation produces the single largest jump in independently judged relevance observed in this study (e.g., 88% versus 78% of candidates scoring above the > 60 LLM-as-a-Judge threshold). Third, Cached-MNRL fine-tuning on top of that English base model adds a further, smaller gain concentrated at the stricter, precision-relevant thresholds (+1.7 points at > 80 , +0.8 points at > 90) while substantially recalibrating the model’s own AI-Match score distribution upward (weighted mean $52.7 \rightarrow 73.5$ on a 0–100 scale), a shift that matters operationally for score-based filtering but is not, by itself, evidence of a proportional ranking-quality gain. Together, these results support EmbeddingGemma + Cached MNRL (English) as the preferred configuration on the evidence available, consistent with the qualitative retrieval-geometry argument for MNRL-only adaptation made in Section 3.4.4.

Future work should prioritize, in order: (1) running a same-market comparative evaluation—North-America-tuned fine-tuning evaluated on a North America job–candidate matching benchmark—to isolate fine-tuning transfer from the France/North-America domain shift noted in Section 3.4.2; (2) executing the Section 3.2 hybrid-retrieval protocol end-to-end for the base and fine-tuned EmbeddingGemma configurations against a held-out, recruiter-labeled set of job orders to obtain directly comparable Recall@ K /MRR/nDCG@ K under geographic and business-rule filters, rather than relying on the fine-tuning posthoc subsets used in this paper; (3) instrumenting embedding latency and Cosmos DB request-unit cost per model; (4) enlarging the per-job ranking evaluation set well beyond the 12–22 jobs available in the current post-hoc splits, ideally toward the hundreds of held-out job orders needed for a stable Recall@ K estimate; and (5) extending the comparative matrix itself—evaluating a managed embedding API such as Cohere Embed v2 English (Section 2.6) and scoring the exploratory EmbeddingGemma + MNRL + Angle checkpoint (Section 3.4.4) on the same comparative evaluation dataset used in Section 4.3—once the corresponding infrastructure and evaluation data are available; and (6) further strengthening the LLM-as-a-Judge protocol (Section 3.5.2) through repeated scoring of the same pairs and cross-validation against a second judge model. Beyond these extensions, integrating a cross-encoder reranker after hybrid retrieval, expanding evaluation to additional locales and job domains beyond France, developing a privacy-preserving public benchmark of labeled (job order, candidate) pairs, and jointly optimizing LLM extraction prompts together with embedding fine-tuning are natural extensions of this research program.

In summary, this paper establishes a reproducible methodology and evaluation framework for comparing embedding strategies under realistic enterprise hybrid-retrieval constraints, and provides the first fine-tuning convergence evidence

and comparative-scoring results within that framework; closing the remaining gaps identified above is the direct path to a complete quantitative comparison of open-weights fine-tuning against commercial embedding APIs for enterprise candidate–job matching.

References

- Chuan Zhang et al. Bert-based ranking for resume-job matching. *IEEE Access*, 8:92813–92822, 2020.
- Hui Lin et al. Few-shot learning for job title classification and skill extraction in job descriptions. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021.
- Gerard Van den Berg et al. Learning to match jobs with resumes from sparse interaction data using multi-view co-teaching network. *arXiv preprint arXiv:1803.01582*, 2018.
- Vladimir Karpukhin et al. Dense passage retrieval for open-domain question answering. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 6769–6781, 2020.
- Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.
- Gordon V Cormack, Charles LA Clarke, and Stefan Büttcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 758–759, 2009.
- Microsoft. Vector search in azure cosmos db for nosql. <https://learn.microsoft.com/en-us/azure/cosmos-db/nosql/vector-search>, 2024.
- Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3982–3992, 2019.
- Xianming Li and Jing Li. AngLE-optimized text embeddings. *arXiv preprint arXiv:2309.12871*, 2023.
- Cohere. Cohere embed v2. <https://docs.cohere.com/docs/cohere-embed>, 2024.
- Google DeepMind. Embeddinggemma: Powerful and lightweight text representations. <https://huggingface.co/google/embeddinggemma-300m>, 2025.
- Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. Introduction to information retrieval. 2008.
- Chuan Qin, Xiaoyu Zhu, Hui Cheng, Xiang Li, Chen Fang, Yue Chang, Jun Xu, Zheng Zhu, Liang Sun, et al. Enhancing person-job fit for talent recruitment: An ability-aware neural network approach. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 677–686, 2018.
- R Thomas et al. Predicting candidate suitability for job openings using standardized job descriptions. *Procedia Computer Science*, 122:595–602, 2017.
- Walid Shalaby et al. Work2vec: A language model for career knowledge mining. In *2018 IEEE International Conference on Data Mining Workshops*, pages 1352–1355, 2018.
- Pieter Delobelle, Thomas Winters, and Chris Develder. Job2vec: Job title benchmarking with representation learning. In *Proceedings of the 2nd Workshop on Natural Language Processing for Human Resources*, pages 29–36, 2020.
- Shashank Bhola et al. Automated skill extraction from recruitment advertisements: A case study. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*, pages 3037–3044, 2020.
- Nils Reimers and Iryna Gurevych. Which document representations are best for text similarity? *Proceedings of the 10th International Conference on Language Resources and Evaluation*, 2016.
- Shuang Yan et al. Learning to match jobs with resumes from sparse interaction data using multi-view co-teaching network. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 791–800, 2019.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 4171–4186, 2019.
- Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*, 2023.
- Jianmo Ni, Chen Qu, Jinglu Lu, et al. Large dual encoders are generalizable retrievers. *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9844–9858, 2022.

- Lee Xiong, Chenyan Xiong, Ye Meng, et al. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *International Conference on Learning Representations*, 2021.
- Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with BERT. In *arXiv preprint arXiv:1901.04085*, 2019.
- Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over BERT. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 39–48, 2020.
- Han Qin et al. Multilingual dense retrieval on a single graph. *arXiv preprint arXiv:2306.02679*, 2023.
- Luyu Gao, Xuan Ma, Jimmy Lin, and Jamie Callan. COIL: Revisit exact lexical match in information retrieval with contextualized inverted list. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 573–581, 2021.
- Tom Brown, Benjamin Mann, Nick Ryder, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.
- John Dagdelen, Alexander Dunn, Sanghoon Lee, et al. Structured information extraction from scientific text with large language models. *Nature Communications*, 15(1):1418, 2024.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.
- Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun-Hsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. Efficient natural language response suggestion for smart reply. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017.
- Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823, 2015.
- Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002.