Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

# What Do Resume–Job Matching Benchmarks Actually Measure? Keywords, Embeddings and LLMs on Public Data

**Nusrat Jahan Rita**

nusratrita00@gmail.com

Right Brain Solution Ltd.

**Research Article**

**Keywords:** resume–job matching, benchmark audit, label quality, sentence embeddings, LLM judge, formatting bias

**Posted Date:** November 11th, 2026

**DOI:** https://doi.org/10.21203/rs.3.rs-11697563/v1

**License:** This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

**Additional Declarations:** No competing interests reported.

# What Do Resume–Job Matching Benchmarks Actually Measure? Keywords, Embeddings and LLMs on Public Data

Nusrat Jahan Rita[1*]

[1*]Right Brain Solution Ltd., Dhaka, Bangladesh.

Corresponding author(s). E-mail(s): nusratrita00@gmail.com;

**Abstract**

Automated resume–job matching ranks job candidates, but most studies use private data. We examine what is, to our knowledge, the only public resume–job benchmark with graded fit labels, comparing keyword methods, sentence embeddings (zero-shot and fine-tuned) and a small open LLM judge (Qwen3-4B). Across 294 jobs, zero-shot embeddings perform slightly but significantly better than TF-IDF (NDCG@10 0.739 vs 0.706). Fine-tuning improves the results, but 430 of 434 test resumes also appear in training. In cross-validation, seeing them explains most of the gain even at equal training size, yet a model trained without them still beats zero-shot methods. The LLM judge does not perform significantly better than embeddings and is about 850 times slower. Good vs Potential Fit is largely set by the job and, within a job, follows the resume more than the match, even when estimated from unrelated jobs. Against the dataset labels, no job-aware method separates these levels significantly better than chance, while a baseline that ignores the job reaches an AUC of 0.727. In a blind relabelling of 150 pairs, the author agrees weakly with the dataset ($\boldsymbol{\kappa = 0.324}$ for fit vs no fit); on 40 pairs, a second annotator agrees with the dataset ($\boldsymbol{\kappa = 0.319}$) more than with the author ($\boldsymbol{\kappa = 0.093}$). Finally, adding markdown without changing any words increases a resume's E5 score by 0.91 standard deviations, regardless of fit. We therefore recommend unseen-resume splits, reliable labels, resume-only baselines, and markdown removal before embedding. We release our code, splits and labels.

**Keywords:** resume–job matching, benchmark audit, label quality, sentence embeddings, LLM judge, formatting bias

# 1 Introduction

Online hiring lets a single job posting attract more resumes than a recruiter can read, so job portals and applicant tracking systems rank and filter candidates automatically. These methods have moved from keyword search to neural text models and, recently, large language models (LLMs) [1, 2], but the goal is the same: to judge how well a resume matches a particular job.

Many strong results in resume–job matching come from private data: neural person–job fit models [3, 4], fine-tuned embedding matchers [5–7] and recent LLM re-rankers [2, 8] are trained and tested on data owned by recruiting platforms or companies, or on data that is no longer available. Other researchers cannot reproduce these experiments, compare new methods on the same test set, or see how the labels were assigned and the data split, so it is hard to tell whether the gains would hold elsewhere.

*resume-job-description-fit* [9] is, to our knowledge, the only public resume–job dataset with graded fit labels (Good Fit, Potential Fit, No Fit), and it has already been used to report model performance [10]. Yet it does not document who assigned the labels, with what guidelines, or how its train and test files were made. Other fields show how easily such gaps mislead: NLI models scored well using only half of each example [11, 12], and question-answering accuracy was inflated by test questions that also appeared in training [13]. To our knowledge, no such checks exist for resume–job matching.

We compare three practical families of methods on this benchmark, keyword methods (TF-IDF, BM25), sentence embeddings (zero-shot and fine-tuned) and a small open LLM judge (Qwen3-4B), and we audit the benchmark itself. We ask five research questions:

- **RQ1.** On public data, how much do sentence embeddings improve ranking compared with keyword methods?
- **RQ2.** How much of the improvement from fine-tuning remains when test resumes are excluded from the training data?
- **RQ3.** Can a small open LLM, used as a judge or a re-ranker, improve ranking performance, and what is the associated cost?
- **RQ4.** What does the benchmark's fine-grained label (Good vs Potential Fit) actually measure, and how reliable are its labels compared with human judgement?
- **RQ5.** Do matchers assign higher scores to resumes after an AI tool polishes or formats them, even when their content is no better?

Zero-shot embeddings beat TF-IDF slightly but significantly (NDCG@10 0.739 vs 0.706). Fine-tuning gains more, but 430 of 434 test resumes also appear in training, and at equal training size most of the gap between standard and strict training comes from those seen resumes. The LLM judge is not significantly better than embeddings and is about 850 times slower. The Good-vs-Potential label behaves unlike a fit label: against the dataset labels, no job-aware method separates the two levels significantly better than chance, while a baseline that never reads the job reaches an AUC of 0.727. A blind relabelling of 150 pairs agrees only weakly with the dataset ($\kappa = 0.324$ for fit vs no fit), and on 40 pairs a second annotator agrees with the dataset ($\kappa = 0.319$) more

than with the author ($\kappa = 0.093$). Finally, adding markdown to a resume without changing a word raises its E5 score by 0.91 SD, for fitting and non-fitting candidates alike.

This paper makes five main contributions:

1. **A reproducible ranking comparison** of keyword, embedding and LLM methods on the only public graded benchmark we know of, with per-job significance tests and cost on free hardware.
2. **Evidence of resume leakage** in the standard split, separated from the effect of training-set size.
3. **An audit of the labels**, showing that the fine-grained label is set mostly by the job and the resume, and that agreement between people is low.
4. **A presentation-bias test** showing that E5-family models reward markdown formatting, which AI writing tools can add, even when fit does not change; removing markdown before embedding removes most of the effect.
5. **Recommendations** (a strict split, fit vs no fit as the main target and a resume-only baseline), with our code, splits and human labels released.

# 2 Related work

De Ruijt and Bhulai [1] review job recommender systems broadly; here we focus on resume–job matching and on benchmark reliability.

## 2.1 Person–job fit before LLMs

Before LLMs, neural models learned the relationship between a job posting and a resume. PJFNN [3] represents both texts in a shared space with convolutional networks, APJFNN [4] adds attention to the ability requirements of the job, and Yang et al. [14] model the preferences of both candidates and employers through an interaction graph. Li et al. [15] match competence levels with context-aware transformers, but for a single job family. These models were trained and tested on private data from recruiting platforms or employers.

**How we differ:** We use only the text and labels of a public dataset, so other researchers can reproduce every experiment.

## 2.2 Embedding matchers

Sentence-BERT [16] made bi-encoders practical: each text is encoded once, and the two are compared by cosine similarity. conSultantBERT [5] fine-tunes it on labelled resume–vacancy pairs from a staffing agency and reports better results than TF-IDF and the off-the-shelf model, while JobBERT [17] learns job-title embeddings from skills. ConFit [6] trains a dense retriever with contrastive learning and data augmentation, and ConFit v2 [7] adds LLM-generated "hypothetical resumes" and hard-negative mining, reporting gains over BM25 and OpenAI embeddings. CareerBERT [18] matches resumes to ESCO occupations and was evaluated by HR experts, which informed the design of our human relabelling.

**How we differ:** With private data, nobody can check whether test resumes also appeared in training; we measure this on public data.

## 2.3 LLM judges and re-rankers

Sun et al. [19] showed that a general LLM can re-rank passages when prompted. In recruitment, GIRL [20] generates job descriptions for job seekers, JobRecoGPT [21] compares deterministic, LLM-based and hybrid approaches for explainable recommendations, and Du et al. [22] enrich resumes with an LLM inside adversarial training. Synapse [23] re-ranks dense retrieval results with an ensemble that includes an LLM and reports a 22% gain in NDCG@10. ConFit v3 [8], the work closest to ours, trains Qwen3 re-rankers (8B and 32B) with supervised fine-tuning and reinforcement learning. Because such models are expensive to run, LANTERN [2] at LinkedIn and Jouanneau et al. [24] distil a large LLM into a smaller model.

**How we differ:** We test a small open LLM (Qwen3-4B) as a zero-shot judge on a single free GPU and report its time per pair alongside its accuracy.

## 2.4 Studies on public data

Work on public resume data mostly describes small applied systems that rank resumes with TF-IDF and cosine similarity [25], classical machine learning [26] or BERT [27], or that extract information from resumes and match it against job requirements [28], often with little evaluation on labelled data. The only published study of the dataset we examine [9] is Jin [10], who treats matching as classification with TF-IDF, cosine and hybrid features and reports a macro-F1 of 0.450 for the three-class task and an AUROC of 0.699 for the binary task. It does not evaluate fine-tuned embeddings, LLMs or ranking metrics, or ask how the labels were assigned.

**How we differ:** We evaluate the same dataset as a ranking task across all three method families and audit its labels and split.

## 2.5 Benchmark and label quality

A model can score well on a benchmark without learning what the benchmark is meant to measure. In natural language inference (NLI), a classifier that sees only the hypothesis predicts about 67% of SNLI and 53% of MultiNLI labels [11], and hypothesis-only baselines beat the majority class on many NLI datasets [12]. In open-domain question answering, about 30% of test questions have a near-duplicate in the training set, and models do much worse on the rest [13]. The test sets of ten common benchmarks contain at least 3.3% label errors on average, and modest increases in such errors can change which model ranks first [29]. Datasheets [30] were proposed to document how a dataset was built and how it should be used.

**How we differ:** Our resume-prior baseline plays the role of the hypothesis-only baseline: it never reads the job. We also measure the overlap of training and test resumes and relabel a sample by hand.

### 2.6 Fairness and manipulation

Automated screening can also be unfair or manipulated. Embedding-based resume retrieval shows gender and race bias [31], as does resume screening with GPT [32]. Explicit gender and race bias has decreased in newer LLMs, but bias related to education remains [33], and the jobs that LLMs recommend differ between demographic groups [34]. Hidden text added to a resume can change the decision of an LLM screener [35].

**How we differ:** Instead of demographic changes or hidden instructions, we test ordinary changes to presentation that an AI writing tool might make: an LLM rewrite that adds no facts, and markdown added to a resume without changing a word.

## 3 Dataset

### 3.1 Source

We use *resume-job-description-fit* [9] (Hugging Face: `cnamuangtoun/resume-job-description-fit`), the only public resume–job dataset with graded fit labels that we found. Each row holds a resume, a job description and one of three labels, **Good Fit**, **Potential Fit** or **No Fit**: 6,241 pairs in the train file and 1,759 in the test file, **8,000 pairs** in total.

The dataset has no dataset card or licence and no information on how the labels were assigned: who labelled the pairs, with what instructions, and whether more than one person checked each pair. The earlier work on it [10] does not report this either, so Section 6.3 examines what the labels represent.

### 3.2 Preparation

Preprocessing is kept minimal so that no method gains an advantage: non-breaking spaces become normal spaces, repeated whitespace is collapsed, and texts are trimmed. Texts are not lowercased or truncated here; each method applies its own input limit (Section 4).

The data has no job or resume identifiers, so we give each distinct text an ID, the first 10 hex digits of the MD5 hash of the cleaned text. The 8,000 rows contain only 351 distinct jobs and 643 distinct resumes. Seven job–resume pairs appear twice; six of them have conflicting labels, so both copies are removed, and one copy of the seventh is kept. The final dataset has **7,987 pairs, 351 jobs and 643 resumes** (Table 1). These IDs catch only exact copies; 148 resumes and 32 jobs have a near-copy (Supplementary Section S4).

**Release split leaks resumes:** The original train and test files share no jobs but 476 resumes, so we do not use them and build our own split (Section 5.1).

**Formatting was lost before release:** Words are often joined where a line or section break used to be, as in `SkillsGAAP`, or a date run into a job title: about 11 spaces are missing per 100 words. The main experiments use the released text, and Section 6.4 tests whether restoring the spaces matters.

**Table 1** Dataset after cleaning.

| | Value |
|---|---|
| Pairs | 7,987 |
| Distinct jobs / resumes | 351 / 643 |
| Labels: No Fit / Potential Fit / Good Fit | 3,993 / 1,998 / 1,996 |
| Pairs per job, median | 16 |
| Pairs per resume, median | 7 |
| Job length in words, median (Q1–Q3) | 313 (184–518) |
| Resume length in words, median (Q1–Q3) | 643 (521–810) |
| Longest resume (words) | 3,134 |

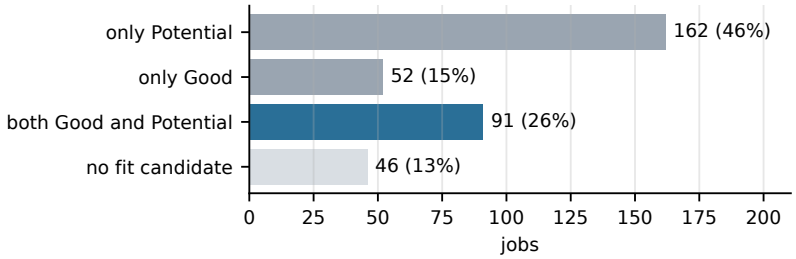



**Fig. 1** Label composition for each job: jobs where fit candidates are only Potential, only Good, or both, together with jobs that have no fit candidate.

## 3.3 Statistics

Two properties matter for the experiments.

**Resumes are long:** The median resume has 643 words, more than the 512-token limit (about 380 words) of the embedding models we use (Section 4.2).

**Each resume appears with many jobs** (median 7), so a split by job puts most test resumes into training as well, which matters for methods that learn from the labels (Section 5.1).

## 3.4 Label structure

The labels are unevenly spread over the 351 jobs. In 46 jobs no candidate is a fit. Of the other 305, **162 label every fit candidate Potential and 52 label every fit candidate Good**; only **91 jobs (30%) use both fit labels** (Fig. 1). For 70% of these 305 jobs, then, Good vs Potential is set by the job rather than by comparing its candidates (66 have one fit candidate; shuffling the two labels across all fit pairs gives about 83 such jobs, not 214), which questions whether this label measures what a ranking benchmark needs (Section 6.3).

## 3.5 Jobs used for ranking evaluation

A job is **usable** for ranking when it has at least 5 candidates, at least one fit candidate and at least two different labels. **294 of the 351 jobs are usable** (7,692 pairs, 641 resumes), including 45 of the 53 test-split jobs. Table 2 lists the job counts behind each analysis.

**Table 2** Job counts used in the paper.

| Jobs | Definition | Used in |
|---|---|---|
| 351 | all jobs | Table 1 |
| 305 | at least one fit candidate | label structure (6.3) |
| 294 | usable, whole dataset | zero-shot comparison (6.1), AI-polish test (6.4) |
| 91 | both Good and Potential among fit candidates | per-job Good-vs-Potential AUC (6.3) |
| 53 / 45 | test split: all / usable | fine-tuned models, LLM judge (6.1–6.2) |
| 86 | usable test + validation jobs | LLM judge paired tests (6.2) |
| 306 | jobs in the test cells of the 5 × 5 cross-validation grid | memorisation experiment (6.1) |
| 42 | jobs in the 150-pair human sample | human relabelling (6.3) |

**Table 3** Methods and settings.

| Family | Method | Size | Input limit | Uses labels | Score |
|---|---|---|---|---|---|
| Keyword | Random | – | – | no | random |
| | TF-IDF | – | full text | no | cosine |
| | BM25 | – | full text | no | Okapi BM25 |
| Embedding, zero-shot | E5-base-v2 | 110M | 512 tokens | no | cosine |
| | BGE-base-en-v1.5 | 110M | 512 tokens | no | cosine |
| | all-mpnet-base-v2 | 110M | 384 tokens | no | cosine |
| | (each, chunked) | | 200-word windows | no | max over windows |
| Embedding, fine-tuned | BGE-base + CoSENT | 110M | 512 tokens | yes | cosine |
| LLM judge | Qwen3-4B-Instruct-2507 | 4B | 450 + 900 words | no | $2P(A) + P(B)$ |
| | Qwen3-1.7B | 1.7B | 450 + 900 words | no | $2P(A) + P(B)$ |
| | E5 → Qwen3-4B re-rank | | top 10 | no | LLM order in top 10 |
| Resume-only | Resume-prior | – | – | yes (other jobs) | mean label of resume |
| | Resume-text classifier | – | full text | yes | $P$(Good) |

# 4 Methods

Every method gives each job–resume pair a score; the candidates of a job are then ranked by that score. Table 3 summarises the methods. Two LLMs are part of the experiments themselves: Qwen3 as a judge (Section 4.4) and Phi-4-mini as a resume rewriter (Section 4.6). Separately, the author used an AI assistant (Claude, Anthropic) to help draft the text and write the analysis code; all experiments were run, checked and interpreted by the author.

## 4.1 Keyword methods

**TF-IDF:** Jobs and resumes are represented by TF-IDF vectors of unigrams and bigrams, with English stop words removed, a minimum document frequency of 2, sublinear term frequency and at most 100,000 features, and compared by cosine similarity. The vocabulary and IDF weights are learned from the text only, never from the labels.

**BM25:** We use Okapi BM25 [36] through `rank_bm25` ($k_1 = 1.5$, $b = 0.75$), with the distinct job terms as the query and the resumes as documents. Text is lowercased, stop words are removed, and tokens may contain `+`, `#` and `.`, so that terms such as `C++`, `C#` and `.NET` stay intact.

**Random:** A random score gives the chance level for every metric.

### 4.2 Zero-shot sentence embeddings

We evaluate three commonly used bi-encoder models without training them on this dataset: **E5-base-v2** [37] (`intfloat/e5-base-v2`), **BGE-base-en-v1.5** [38] (`BAAI/bge-base-en-v1.5`) and **all-mpnet-base-v2** [16, 39] (`sentence-transformers/all-mpnet-base-v2`). For E5, we use the `query:` and `passage:` prefixes that were used during its training.

The job and resume are encoded separately, L2-normalised and compared by cosine similarity, so resumes can be encoded ahead of time.

**Long resumes:** E5 and BGE read up to 512 tokens and MPNet 384 (about 380 and 290 words), so most resumes (Section 3.3) are cut at the input limit. In a *chunked* variant, each resume is split into 200-word windows with a 150-word stride, and the best-scoring window gives the resume's score.

We also evaluate MiniLM-L6-v2 [40] (256 tokens), nomic-embed-text-v1.5 [41] truncated at 2,048 tokens, and the cross-encoder ms-marco-MiniLM-L-6-v2 [42], where the job and resume share 512 tokens.

### 4.3 Fine-tuned embedding model

To measure what training on the labels adds, we fine-tune BGE-base-en-v1.5 on the training pairs, with a target similarity of 0, 0.5 or 1 for No, Potential and Good Fit. We use the CoSENT loss for 2 epochs (batch size 16, learning rate $2 \times 10^{-5}$, 10% warm-up, fp16, seed 42, 512 tokens), without hard-negative mining or early stopping. The settings are the same for every training set (Section 5.1), so only the training data differ. Training takes about 11 minutes on a T4 GPU.

### 4.4 LLM judge

As judges we use small open instruction-tuned models [43] that run on a single free GPU: **Qwen3-4B-Instruct-2507** (main) and **Qwen3-1.7B**. Prompted to act as an experienced technical recruiter, the model reads the first 450 words of the job and the first 900 words of the resume and chooses A) Good Fit, B) Potential Fit or C) No Fit, each with a one-line definition (full prompt in Supplementary Information, Section S1).

Instead of generating text, we take the next-token probabilities of the three letters, renormalise them and compute

$$\text{score} = 2\,P(A) + P(B), \tag{1}$$

a score between 0 and 2 for ranking. The most probable letter is the hard prediction for macro-F1. Thinking mode is off for Qwen3-1.7B (the 4B model has none), and the models run in float16 with a batch size of 4.

**Re-ranking:** In a cheaper variant, E5 ranks all candidates and the LLM re-orders only the top 10.

### 4.5 Resume-only baselines

Two baselines **never read the job**.

**Resume-prior:** A resume's score is its mean label (0–2) over all *other* jobs. It can carry real fit when those jobs resemble this one, so we also compute it from dissimilar jobs only (Section 6.3). A resume with no other labelled job, as in the strict training sets, gets the training mean, which is constant within a fold.

**Resume-text-only classifier:** To rule out simple memorisation, a logistic regression on the resume's TF-IDF features predicts Good vs Potential, evaluated with 5-fold cross-validation grouped by resume, so every test resume is unseen.

### 4.6 AI-polished resumes (RQ5)

We used **Phi-4-mini-instruct** [44], from a different model family than the Qwen judge, to rewrite 200 resumes (seed 42) in a polished, professional style. It was told to keep every job, employer, date, degree, skill and number and to add nothing, with greedy decoding and no job description, so it improved the wording without tailoring it (prompt in Supplementary Information, Section S1). After removing outputs shorter than 0.6 or longer than 1.8 times the original and any cut-off responses, **181** resumes remained, with a stricter subset of **151** that adds no new number. In 20 rewrites checked by hand, none added a fact, but all 20 shortened or dropped specific details such as tools, duties or projects.

Each polished resume was scored against all its jobs, with every other resume unchanged, to measure the change in its score and rank. Two checks separate rewriting from cleaning and formatting. For **cleaning**, we restored the spaces lost in scraping with fixed rules (`SkillsGAAP` → `Skills GAAP`). For **formatting**, we stripped markdown from the polished resumes and, in reverse, added it to the *original* resumes in five versions: *flat* (cleaned text), *lines* (each heading and item on its own line), *bold* headings, *bullets*, and *full* (bold headings and bullets). Headings were marked only where the scraper had joined them to the next word, so every version keeps exactly the same letters and digits as the original. Finally, we ranked every job with all resumes in the full version.

We repeated the reverse test with seven more embedding models: E5-large-v2 [37], multilingual-E5-base [45], BGE-large-en-v1.5 [38], GTE-base [46], all-MiniLM-L6-v2 [40], Nomic-embed-text-v1.5 [41] and Qwen3-Embedding-0.6B [47], each with its recommended input format and all cut at 512 tokens. Before running it, we recorded the criterion in the notebook: a model shows the bias if full formatting raises its score by at least 0.30 SD with Holm-corrected $p < 0.05$. Nomic, run for Table 4 in another environment, could not be loaded here.

## 5 Experimental setup

### 5.1 Splits

Every split keeps all pairs of a job together, because the model must rank candidates for jobs it has not seen in training.

**Standard split:** The jobs are divided 70 / 15 / 15 into training, validation and test (`GroupShuffleSplit` by job, seed 42), giving 245 / 53 / 53 jobs. Because each resume appears with several jobs (Section 3.3), **430 of the 434 test resumes also appear in training pairs**, though with other jobs.

**Strict split:** The same test jobs, with every training pair that contains a test resume removed, so **both test jobs and test resumes are unseen in training**.

$5 \times 5$ **cross-validation grid:** With only 45 usable test jobs, one split cannot separate memorisation from other effects, so we divide the jobs into 5 groups and the resumes into 5 groups (seed 42). In fold $k$, a pair is a test pair when its job is in job group $k$ and its resume in resume group $k$, giving about 1,600 test pairs from 306 jobs across the folds. In each fold, three models are trained and tested on the same pairs:

- **standard:** every pair whose job is not in group $k$, so the test resumes may have been seen with other jobs;
- **strict:** only pairs whose job **and** resume are not in group $k$;
- **size-matched:** a random subset of the standard set with exactly as many pairs as the strict set.

*Standard* − *strict* is the total benefit of having seen the test resumes; *size-matched* − *strict* is the part that remains at equal training size, which comes from the resumes themselves.

The zero-shot methods do not learn from the labels, so we evaluate them on **all 294 usable jobs** (Section 3.5) for more statistical power. The LLM judge runs on the usable test and validation jobs ($45 + 41 = 86$), because it is slow (Section 6.2).

## 5.2 Metrics

The labels are graded relevance levels: Good Fit = 2, Potential Fit = 1, No Fit = 0. **Ranking** metrics are computed per job and averaged over jobs:

- **NDCG@$k$** [48] ($k = 5, 10$), with gain $2^{rel} - 1$ and a $\log_2$ position discount; NDCG@10 is the main metric.
- **P@5** and **MRR**, counting only Good Fit as relevant.

**Separation** is measured by the area under the ROC curve: **AUC fit-vs-no** (Good and Potential against No Fit) and **AUC Good-vs-Potential** (among fit pairs only, Section 6.3). AUC is either **pooled** over all pairs or computed **per job** and averaged over jobs with both classes. The per-job version matches the ranking task, while the pooled one can also reward separating jobs rather than candidates; each table says which is used.

**Classification:** Macro-F1 over the three labels is reported for the LLM judge only, to compare with the earlier result on this dataset (0.450 [10]).

**Agreement:** Cohen's $\kappa$ [49] measures agreement between human and dataset labels on three labels, fit vs no fit, and Good vs Potential (Section 5.4).

## 5.3 Statistical tests

All comparisons are paired, on the same jobs or pairs.

- **Methods over jobs** (zero-shot comparison, LLM judge): paired sign-flip permutation test on per-job differences (10,000 permutations); 95% CIs from 2,000 bootstrap resamples over jobs.
- **Cross-validation grid:** CIs and paired differences from a **cluster bootstrap over test jobs**, which keeps each job's pairs together; for NDCG@5 also a Wilcoxon signed-rank test [50] over jobs.
- **Human labels:** 2,000 bootstrap resamples over pairs for the CIs of $\kappa$ and of AUC changes, with the tests fixed in an analysis plan beforehand (Section 5.4).
- **AI-polished resumes:** Wilcoxon signed-rank test on each pair's score change.

$p$-values for several methods against one baseline are adjusted with the **Holm** method [51], with $\alpha = 0.05$, and we report effect sizes with confidence intervals rather than $p$-values alone.

## 5.4 Human relabelling protocol

To check what the labels measure, the author relabelled a sample of pairs by hand.

- **Sample:** 150 test-split pairs, 50 per dataset label (seed 42), covering 42 jobs, fixed before labelling; the sheet was then grouped by job so that a job's candidates could be read together.
- **Blind:** no dataset label or model score was visible while labelling.
- **Instruction:** judge how well each candidate fits **this specific job**, with the same three labels. Section S2 of the Supplementary Information summarises the criteria actually used, written afterwards from the annotator's notes.
- **Notes:** a short reason for every pair.
- **Analysis plan:** committed to the project repository on 30 Sep 2026, when 53 of the 150 pairs had been labelled and before any comparison with the dataset labels or a model, with the primary test, secondary measures and the reading of each possible outcome. Section 6.3 follows it.
- **Second annotator:** a second person labelled a stratified subset of 40 of the 150 pairs under the same conditions, with its own plan fixed in advance. A first round on 40 other pairs was discarded without analysis because an AI tool had been used against the instructions; the same person then labelled the 40 new pairs without one.

## 5.5 Hardware and cost

All GPU experiments ran on a single NVIDIA T4 (16 GB) in Google Colab. We report the median time to score one pair and to rank all candidates for one job, and, since a small portal would usually have no GPU, the embedding model's speed on a single CPU thread.

Random seeds are fixed (42 unless stated otherwise), and the code, data splits and human labels are released with the paper (see Declarations), with library versions, keyword lists and training-set sizes in the repository README.

**Table 4** Zero-shot ranking on all 294 usable jobs. Both AUC columns are per-job means. Δ and $p$ are against TF-IDF (paired permutation test, Holm-corrected).

| Method | NDCG@10 [95% CI] | Δ | $p$ | AUC fit/no | AUC G/P |
|---|---|---|---|---|---|
| E5-base | 0.739 [0.712, 0.765] | +0.033 | 0.014 | 0.741 | 0.474 |
| E5-base, chunked | 0.738 [0.712, 0.763] | +0.031 | 0.014 | 0.736 | 0.495 |
| MPNet-base | 0.736 [0.710, 0.763] | +0.030 | 0.019 | 0.732 | 0.511 |
| BGE-base | 0.735 [0.708, 0.760] | +0.028 | 0.038 | 0.746 | 0.450 |
| Nomic-v1.5 (2,048 tokens) | 0.733 [0.707, 0.758] | +0.026 | 0.059 | 0.737 | 0.472 |
| BGE-base, chunked | 0.731 [0.704, 0.758] | +0.025 | 0.040 | 0.750 | 0.498 |
| MPNet-base, chunked | 0.730 [0.703, 0.756] | +0.023 | 0.069 | 0.733 | 0.511 |
| MiniLM-L6, chunked | 0.708 [0.680, 0.733] | +0.001 | 1.000 | 0.701 | 0.454 |
| TF-IDF | 0.706 [0.680, 0.732] | – | – | 0.689 | 0.524 |
| MiniLM-L6 | 0.703 [0.677, 0.728] | −0.003 | 1.000 | 0.690 | 0.448 |
| BM25 | 0.672 [0.647, 0.698] | −0.034 | <0.001 | 0.640 | 0.505 |
| Resume-prior (no job text) | 0.671 [0.644, 0.699] | −0.035 | 0.134 | 0.665 | 0.727 |
| Cross-encoder MiniLM | 0.660 [0.634, 0.687] | −0.046 | 0.007 | 0.641 | 0.510 |
| Random | 0.582 [0.556, 0.605] | −0.125 | <0.001 | 0.514 | 0.520 |

# 6 Results

## 6.1 Keywords vs embeddings, and fine-tuning vs memorisation (RQ1, RQ2)

### Zero-shot ranking

Table 4 compares the methods that do not learn from the labels (plus the resume-prior) on all 294 usable jobs. The three main embedding models perform a little better than TF-IDF: E5 achieves an NDCG@10 of 0.739, MPNet 0.736 and BGE 0.735, against 0.706 for TF-IDF. The improvements are small (+0.028 to +0.033) but significant after Holm correction ($p = 0.014$, 0.019 and 0.038). With fit vs no fit as the relevance, the gap is clearer: their per-job AUC is 0.043 to 0.057 above TF-IDF (all Holm $p = 0.001$).

Four other results are worth noting.

- **BM25 performs worse than TF-IDF** (0.672, $p < 0.001$), possibly because long, repetitive resumes weaken its length normalisation.
- **Using more of the resume does not help.** Chunking gives almost the same result as cutting at the input limit (E5: 0.738 vs 0.739), and Nomic, which we ran with a 2,048-token window, scores 0.733.
- **The web-search cross-encoder performs worst among the neural models** (0.660, below TF-IDF), either because web-search relevance does not transfer or because job and resume must share its 512 tokens.
- **A method that never reads the job comes close to BM25.** The resume-prior scores 0.671 and is not significantly worse than TF-IDF ($p = 0.13$). Section 6.3 asks how much of the ranking comes from the resumes themselves.

**Table 5** Fine-tuned BGE under the standard, size-matched and strict training sets ($5 \times 5$ cross-validation, about 1,600 test pairs from 306 jobs, median 4 candidates per job). AUC fit/no per job: 179 jobs, CI: bootstrap over jobs; AUC G/P: per-job mean; NDCG@5 ties broken at random. The strict resume-prior is constant within a fold, so its pooled AUC reflects only fold differences.

| Method | AUC fit/no per job [95% CI] | pooled | AUC G/P | NDCG@5 |
|---|---|---|---|---|
| FT standard | 0.840 [0.800, 0.877] | 0.795 | 0.570 | 0.857 |
| FT size-matched | 0.823 [0.785, 0.858] | 0.787 | 0.552 | 0.846 |
| FT strict | 0.736 [0.694, 0.777] | 0.758 | 0.468 | 0.781 |
| BGE-base (zero-shot) | 0.665 [0.618, 0.710] | 0.659 | 0.447 | 0.734 |
| E5-base (zero-shot) | 0.648 [0.603, 0.695] | 0.644 | 0.492 | 0.722 |
| TF-IDF | 0.649 [0.606, 0.690] | 0.628 | 0.478 | 0.736 |
| Resume-prior (standard train) | 0.669 [0.619, 0.716] | 0.661 | 0.729 | 0.780 |
| Resume-prior (strict train) | 0.500 | 0.454 | 0.500 | 0.694 |

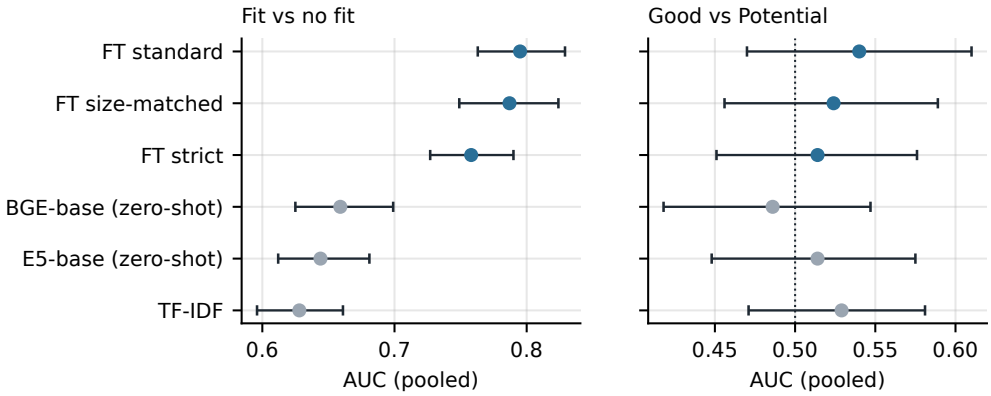



**Fig. 2** Pooled AUC in the $5 \times 5$ cross-validation (bars: 95% CIs). Left: fit vs no fit. Right: Good vs Potential (dotted line = chance).

## Fine-tuning and resume memorisation

Fine-tuning BGE gives the best result in the paper, a test NDCG@10 of 0.833. With the strict split, which keeps test resumes out of training, it drops to 0.791, close to the zero-shot models on the same jobs (0.78–0.79, Table 6).

The cross-validation grid shows this is not an unlucky split: per-job fit-vs-no AUC is 0.840 (standard), 0.823 (size-matched) and 0.736 (strict), with the same order when pooled (Table 5, Fig. 2). Its test lists are short (median 4 candidates per job), so we use per-job AUC rather than NDCG as the main measure here.

Four comparisons explain the drop.

- **Seeing the test resumes is worth +0.104 per-job AUC** (standard − strict, CI [0.066, 0.141]; pooled +0.037).
- **Training-set size does not explain it.** At equal size, the size-matched model is still +0.087 above the strict model (CI [0.052, 0.122]; pooled +0.029), while standard and size-matched differ by only +0.017 (Holm $p = 0.065$).

**Table 6** Ranking quality on the 45 usable test jobs and time to rank one job (median 20 candidates) on a T4 GPU. Macro-F1 is over the three dataset labels (judges only).

| Method | NDCG@10 | Macro-F1 | Seconds per job |
|---|---|---|---|
| Random | 0.644 | – | <0.001 |
| TF-IDF | 0.780 | – | 0.084 |
| BM25 | 0.695 | – | 0.119 |
| E5-base (zero-shot) | 0.784 | – | 0.032 |
| BGE-base (zero-shot) | 0.781 | – | 0.032 |
| MPNet-base (zero-shot) | 0.791 | – | 0.028 |
| BGE-base (fine-tuned, standard) | 0.833 | – | 0.016 |
| BGE-base (fine-tuned, strict) | 0.791 | – | 0.016 |
| Qwen3-4B judge (all pairs) | 0.807 | 0.368 | 27.4 |
| Qwen3-1.7B judge (all pairs) | 0.759 | 0.316 | 11.9 |
| E5 → Qwen3-4B re-rank top 10 | 0.780 | – | 13.7 |
| E5 → Qwen3-1.7B re-rank top 10 | 0.763 | – | 6.0 |

- **Ranking shows the same pattern.** NDCG@5 is +0.076 (standard) and +0.065 (size-matched) above the strict model (both Holm $p < 0.001$).
- **Near-copies do not explain it.** Without test pairs that have a near-copy in training, size-matched − strict stays positive (Supplementary Table S5).

The strict model still beats every zero-shot method: per-job AUC +0.071 over zero-shot BGE and +0.088 over E5 and TF-IDF (all Holm $p < 0.001$ over the six comparisons), and NDCG@5 +0.045 to +0.059.

So **fine-tuning learns real signals about job fit, but the usual job-grouped split makes the gain look larger than it is**.

## 6.2 LLM judge, re-ranking and cost (RQ3)

On the 45 test jobs, Qwen3-4B achieves an NDCG@10 of 0.807, against 0.784 for E5. On all 86 usable test and validation jobs, however, the gain shrinks to +0.012 and is not significant ($p = 0.59$), although 86 jobs cannot rule out a small advantage. Re-ranking only E5's top 10 does not help either (−0.007 on the 86 jobs, $p = 0.69$), and the smaller Qwen3-1.7B scores below E5 (0.759).

As a classifier, Qwen3-4B reaches a macro-F1 of 0.368 on the 45 test jobs, below the 0.450 reported by Jin [10], but the two are not comparable: our judge is zero-shot, Jin's classifiers were trained on the labels, and the test sets differ. Qwen3-1.7B scores 0.316 and never predicts Potential Fit.

The judge is also slow (Table 6, Fig. 3). On a T4 GPU it takes a median of 1.37 s per pair, about 27 s to rank one job, or 13.7 s when it only re-ranks E5's top 10. E5 ranks the same job in 0.03 s, about 850 times faster, and fine-tuned BGE in 0.016 s. On a single CPU thread, E5 indexes a new resume in about 2.0 s and ranks all 643 resumes for a new job in 1.6 s.

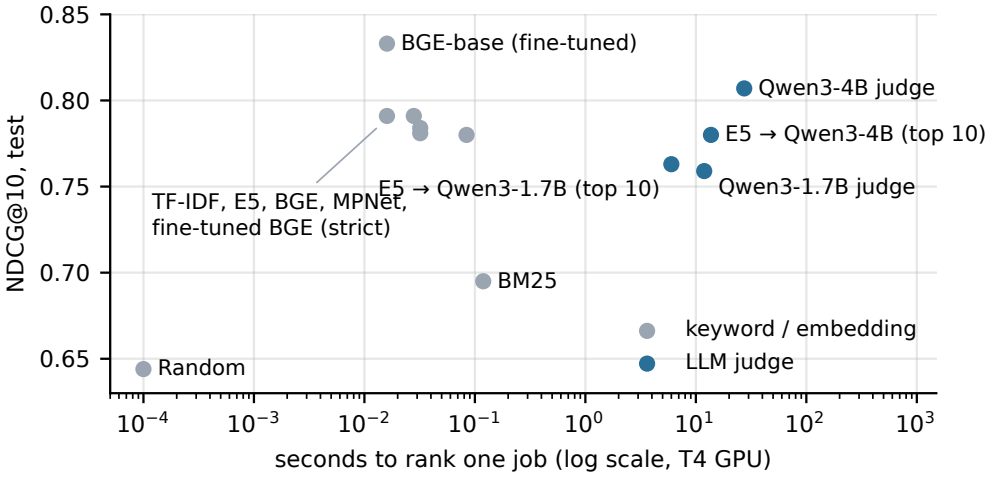



**Fig. 3** Ranking quality (test NDCG@10) against time to rank one job on a T4 GPU (log scale).

## 6.3 What does Good vs Potential measure? (RQ4)

### Within the dataset

Three findings suggest the same conclusion: the Good-vs-Potential label depends much more on which job and which resume are involved than on how well the two match each other.

**1. The label is largely determined by the job.** Of the 305 jobs with at least one fit candidate, 214 give every fit candidate the same level (Section 3.4). Job identity alone explains $R^2 = 0.70$ of the Good-vs-Potential label, and 0.64 on held-out pairs (cross-validated).

**2. Within the same job, the label depends more on the resume.** Among the 91 jobs that use both labels, resume identity explains $R^2 = 0.43$ (0.15 on held-out pairs). Against the dataset labels, methods that compare the job with the resume are not significantly above chance: the mean per-job AUC is 0.45–0.52 for the embedding and keyword methods (Table 4) and 0.61 [0.47, 0.74] for the Qwen3-4B judge (on the mixed jobs among the 86 it scored). The resume-prior, which never reads the job, reaches **0.727**. Part of this could be real fit on similar jobs, but from only the resume's other jobs less similar to this job than its median (E5 cosine), the prior still reaches **0.637** [0.593, 0.681] (TF-IDF similarity: 0.659), while its fit-vs-no AUC falls to chance (0.477, from 0.665). Good vs Potential thus keeps a resume-level signal that related-job fit does not explain, unlike fit vs no fit.

**3. Fine-tuning mainly learns this pattern when the resume is already familiar.** The per-job Good-vs-Potential AUC is 0.570 for the standard model and 0.468 for the strict one, while a resume-prior from the same training labels reaches 0.729 (Table 5).

**Memorising resumes, or reading them?** On unseen resumes, a classifier that uses only the resume text reaches a per-job AUC of 0.598 [0.552, 0.645] with near-copies kept in one fold (0.635 when grouped by exact ID; Supplementary Table S6),

**Table 7** Good-vs-Potential AUC (pooled over pairs) on the 150 relabelled pairs, against the dataset labels (100 fit pairs) and the author's labels (78 fit pairs; 77 for the judges). *p*: pair bootstrap, uncorrected; Holm-adjusted values in Supplementary Table S7.

| Method | AUC, dataset | AUC, human | Shift (human − dataset) | *p* |
|---|---|---|---|---|
| Resume-prior | 0.521 | 0.407 | −0.114 | 0.211 |
| TF-IDF | 0.368 | 0.740 | +0.372 | <0.001 |
| E5-base | 0.434 | 0.655 | +0.221 | 0.011 |
| BGE-base | 0.435 | 0.630 | +0.195 | 0.018 |
| MPNet-base | 0.467 | 0.698 | +0.232 | 0.007 |
| Qwen3-4B judge | 0.401 | 0.760 | +0.359 | <0.001 |
| Qwen3-1.7B judge | 0.404 | 0.670 | +0.266 | 0.003 |

while a job-only model scores 0.500 by construction. The label therefore also reflects information in the resume itself, though not simple properties such as length, stated experience or degree ($|\rho| \leq 0.07$, per-job AUC 0.44–0.52).

**A caveat: technical jobs.** The 91 jobs that use both fit labels are mostly technical (67%, against 33–41% of the jobs that use one level). With near-copies grouped, the resume-text classifier reaches 0.623 [0.570, 0.677] on the 61 mixed technical jobs but only 0.546 [0.464, 0.631] on the 30 non-technical ones ($p = 0.29$), and removing technical terms leaves 0.603. The resume signal is therefore clearer in technical jobs (the difference is not significant), and it is not just technical vocabulary.

## Human relabelling

We followed the analysis plan fixed before any comparison (Section 5.4). Agreement between the author and the dataset was low: $\kappa$ was **0.120** for three labels (CI [0.015, 0.233]) and **0.324** for fit vs no fit (CI [0.177, 0.476]), which the plan defines as noisy labels (outcome D). On the 64 pairs that both called a fit, Good-vs-Potential agreement was **−0.184** (CI [−0.415, 0.063]), no better than chance (Supplementary Table S1).

The disagreements were one-sided (Supplementary Table S2). From 50 pairs per dataset label, the author marked 72 No Fit, 36 Potential Fit and 42 Good Fit. Of the 26 strongest disagreements (Good Fit against No Fit), 19 were dataset Good Fit, human No Fit, usually a mismatch between the candidate's background and the job, such as a quality-assurance resume for a C++ developer role.

**Job-aware methods match the author's labels more closely than the dataset's labels** (Table 7, Supplementary Fig. S1). Against the author's labels, every method that reads both the job and the resume separates Good Fit from Potential Fit (AUC 0.63–0.76); against the dataset labels, AUC is only 0.37–0.47. E5, the content method named in the plan, gains +0.221 (CI [0.058, 0.384], $p = 0.011$), and the other methods +0.195 to +0.372 (all Holm $p < 0.05$, also when whole jobs are resampled; Supplementary Table S7). The two columns use different pairs (100 vs 78), but on the 64 pairs that both call a fit the shift is larger (E5: 0.328 to 0.669; Supplementary Table S8). The resume-prior moved the other way, from 0.521 to 0.407, but this change, the plan's primary test, was not significant (−0.114, CI [−0.281, 0.058], $p = 0.21$). Fit

**Table 8** Score change when a resume is replaced by its AI-polished version (181 resumes, 2,076 pairs; Qwen3-4B judge: 320 pairs). Changes are in standard deviations of each method's scores over all evaluated pairs (judge: its 320 pairs); CIs: bootstrap of the mean change. Places = mean rank gain among the job's candidates; top 5 = share of pairs newly reaching the top 5.

| Method | All pairs [95% CI] | Fit | No fit | Share up | Places | Top 5 |
|---|---|---|---|---|---|---|
| TF-IDF | +0.072 [0.031, 0.108] | +0.082 | +0.063 | 0.646 | +1.3 | 0.040 |
| E5-base | +0.743 [0.688, 0.801] | +0.744 | +0.743 | 0.948 | +10.9 | 0.341 |
| BGE-base | +0.184 [0.119, 0.252] | +0.191 | +0.177 | 0.699 | +2.9 | 0.088 |
| MPNet-base | −0.079 [−0.165, −0.001] | −0.052 | −0.105 | 0.423 | −1.5 | 0.039 |
| Qwen3-4B judge | +0.044 [−0.007, 0.092] | +0.045 | +0.044 | 0.522 | – | – |

vs no fit shows the same pattern (E5: 0.570 to 0.737; Qwen3-4B: 0.713 to 0.843; judge macro-F1 0.360 to 0.577).

**A second annotator:** A colleague labelled 40 of the 150 pairs (27 jobs), stratified by the author's labels, under the same blind conditions (Section 5.4). Agreement stayed low: the two humans had $\kappa = 0.115$ [−0.109, 0.347] on three labels and **0.093** [−0.191, 0.395] for fit vs no fit, while the colleague agreed with the dataset at $\kappa = 0.319$ [−0.003, 0.595]. The plan's condition that the two humans agree with each other more than with the dataset was not met (paired difference −0.226, CI [−0.550, 0.119]), so we treat this comparison as inconclusive. When the author relabelled 30 of the pairs two weeks later, her self-agreement for fit vs no fit was $\kappa = 0.426$ [0.146, 0.722], so the low agreement reflects different judges more than random labelling, although her second pass was stricter (Supplementary Table S9).

In an exploratory analysis outside the plan, TF-IDF matched the author's fit-vs-no labels better than the colleague's (AUC 0.812 vs 0.573), while E5 did not (0.717 vs 0.734). The author's weight on skill overlap may explain TF-IDF's large shift (+0.372).

**What do these findings tell us?** As the plan foresaw, **label reliability is the main concern**: the dataset labels agree only weakly with the author's, the colleague agrees with the dataset as much as the author does, and the two humans agree with each other even less, so no source is a gold standard. The findings that do not depend on human judgement remain: Good vs Potential is largely set by the job and, within a job, follows the resume.

## 6.4 AI-polished resumes and formatting bias (RQ5)

Each of the 181 rewritten resumes was scored against its jobs, with the other candidates' resumes unchanged (Section 4.6).

**E5 gives polished resumes a clear advantage, even when they are not a good fit for the job.** E5 scores rise by 0.74 SD (CI [0.69, 0.80]) after rewriting (Table 8, Supplementary Fig. S2): 95% of pairs score higher, rankings improve by 10.9 places on average, and 34% of pairs newly reach the top 5. The effect is almost exactly the same for fit and no-fit pairs (0.744 vs 0.743 SD). Other methods change much less: BGE +0.18 SD, TF-IDF +0.07 SD and MPNet −0.08 SD, and the Qwen3-4B judge barely moves (+0.04 SD, CI [−0.01, 0.09]). The results are the same for the 151 rewrites that add no new number.

**Table 9** Splitting the polish effect into steps (change in SDs). Cleaning: original → spaces restored. Polish beyond cleaning: cleaned → polished. Markdown symbols: polished with markdown stripped → polished (part of the previous column, not an extra step). All changes are significant (Wilcoxon, Holm $p < 0.05$) except those marked n.s.; for the judge, whose scores change in small steps, the test is significant although the CI of the mean in Table 8 includes 0.

| Method | Cleaning | Polish beyond cleaning | Markdown symbols | Total |
|---|---|---|---|---|
| TF-IDF | +0.140 | −0.068 | 0.000 (n.s.) | +0.072 |
| E5-base | −0.049 | +0.792 | +0.662 | +0.743 |
| BGE-base | −0.038 | +0.222 | +0.101 | +0.184 |
| MPNet-base | +0.048 | −0.127 | −0.062 | −0.079 |
| Qwen3-4B judge | −0.006 (n.s.) | +0.050 | – | +0.044 |

**The bigger issue is formatting, not the rewritten wording** (Table 9). Restoring the missing spaces (about 11 per 100 words) changes E5 scores by only −0.05 SD, while stripping markdown from the polished resumes, with the wording unchanged, removes 0.66 SD of the 0.74 SD increase.

In the reverse test, we added markdown to the original resumes without changing any words (Supplementary Table S3). E5 scores rose by **0.91 SD**, more than after the AI rewrites (+0.021 in raw cosine, against an SD of 0.023), and rankings improved for 99% of pairs by 12.6 places on average. Bold headings alone added +0.72 SD, bullets alone +0.71 SD and line breaks +0.11 SD. TF-IDF ignores the symbols, MPNet barely moved and BGE rose by 0.18 SD. Formatting again helped fit and no-fit pairs almost equally (difference +0.004 SD, $p = 0.45$).

Two other explanations do not hold. Markdown is rare in the job texts (5% contain an asterisk) and unrelated to E5's gain ($\rho = 0.06$). The formatted resumes are *longer in tokens*, so with its 512-token limit E5 sees less of each resume yet scores it higher. And formatting every resume leaves ranking quality unchanged (NDCG@10 0.736 for the cleaned text to 0.738, $p = 0.71$, uncorrected).

**Why does this matter?** The preference for markdown does not help E5 tell better matches from worse ones, but when formatted and plain resumes compete, a candidate whose resume carries markdown may outrank an equally qualified one.

**Is this only an E5 problem?** We repeated the reverse test with seven more embedding models, six of which loaded (Table 10). The effect is strongest in the E5 family: after E5-base (+0.91 SD), E5-large rose by 0.47 SD and multilingual-E5 by 0.49 SD (about 7 places). BGE-large rose by 0.23 SD, while GTE-base, Qwen3-Embedding and MiniLM moved by at most 1.6 places on average. By the criterion set before the experiment, two of the six models show the formatting bias: the effect is not limited to one model, but it is not a general property of embedding models either. The fit − no-fit difference never exceeded 0.075 SD, so the effect is largely independent of fit.

Formatting all resumes left NDCG@10 unchanged for six of the seven models; only E5-large fell, from 0.750 to 0.719 (Holm $p = 0.001$). All three E5 models use the `query:` and `passage:` prefixes and the other models do not, which may play a role, but we did not test this.

**Table 10** The markdown-injection test across embedding models (181 resumes, 2,076 pairs, all inputs cut at 512 tokens). Full = bold headings + bullets, in SDs. Fit − no fit: difference in the full-formatting change between fit and no-fit pairs (139 resumes). ΔNDCG@10: every resume flat vs every resume fully formatted (294 jobs, Holm-corrected $p$). Nomic-embed-text-v1.5 failed to load and is missing.

| Model | Size | Full [95% CI] | Share up | Places | Fit − no fit | ΔNDCG@10 ($p$) |
|---|---|---|---|---|---|---|
| E5-base | 110M | +0.912 [0.861, 0.968] | 0.992 | +12.6 | +0.004 | +0.002 (1.00) |
| E5-large | 335M | +0.474 [0.407, 0.545] | 0.900 | +7.1 | −0.006 | −0.031 (0.001) |
| Multilingual-E5-base | 278M | +0.488 [0.394, 0.577] | 0.842 | +7.7 | +0.075 | −0.009 (1.00) |
| BGE-large | 335M | +0.226 [0.188, 0.265] | 0.788 | +3.7 | +0.017 | +0.003 (1.00) |
| GTE-base | 110M | +0.096 [0.066, 0.124] | 0.647 | +1.6 | +0.045 | +0.002 (1.00) |
| all-MiniLM-L6 | 23M | −0.070 [−0.104, −0.035] | 0.365 | −1.5 | −0.018 | −0.006 (1.00) |
| Qwen3-Embedding-0.6B | 600M | +0.065 [0.017, 0.112] | 0.577 | +1.2 | −0.063 | −0.003 (1.00) |

# 7 Discussion and recommendations

## 7.1 What the benchmark measures

On this benchmark, methods separate fit from no fit reasonably well, but not Good from Potential Fit. At the fine level, most jobs give every fit candidate the same label, and where labels differ they follow the resume more than the match. Even the coarse label is not a gold standard: a careful human reader agrees with it only weakly, and two human readers agree with each other even less. A ranking score on this benchmark therefore shows how closely a method follows an unknown labelling process, not how well it judges fit.

The checks that revealed this were cheap: a resume-only baseline, a count of resumes shared between splits, and a small blind relabelling. Each works on private data too, and we recommend them for future work, as hypothesis-only baselines are now reported in NLI [11, 12].

## 7.2 Recommendations for people who use this benchmark

1. **Keep test resumes out of training,** or at least report a size-matched control: under the usual job-grouped split, most of the gap between standard and strict training (+0.087 of +0.104 per-job AUC) comes from resumes already seen (Section 6.1).
2. **Use fit vs no fit as the main target,** and treat Good vs Potential with caution until better labels show it is reliable (Section 6.3).
3. **Always report a resume-only baseline.** If a method that never reads the job scores close to yours, your method may be reading the resume rather than the fit.
4. **Evaluate per job, on enough jobs.** No embedding gain over TF-IDF was significant on the 45 test jobs (all Holm $p \geq 0.78$), but the three main embeddings' gains were on the 294 usable jobs (Section 6.1).

### 7.3 Recommendations for a small job portal

Without a GPU, a zero-shot embedding model gives most of the available benefit: E5 ranked about as well as the Qwen3-4B judge (a non-significant gap of 0.012; Section 6.2), indexes a resume in about 2 s on one CPU thread and ranks 643 resumes in about 1.6 s, while the judge needs a GPU and about 27 s per job. With an E5-family model, a formatted resume can rise 7 to 13 places with unchanged content (Section 6.4), so a portal should strip markdown before embedding and test its matcher on formatted and plain versions of the same resumes.

### 7.4 Recommendations for benchmark builders

A datasheet [30] stating who labelled the pairs, with what instructions, would let users judge the labels; labelling candidates job by job, with two or more annotators and reported agreement, would give Good vs Potential a clear meaning.

## 8 Threats to validity

**One dataset:** We study a single undocumented English dataset, so our claims concern this benchmark, not resume–job matching in general.

**Human labels:** The author labelled 150 pairs and a second annotator 40, so the confidence intervals are wide. The author, who knew the study's hypothesis, is a software QA engineer, not a recruiter, and judged mainly on skill and role overlap, which may favour keyword methods (Section 6.3), and the labelling guide was written after labelling; the stratified sample (one third No Fit, half in the data) also affects $\kappa$. We treat them as evidence on reliability, not as correct labels.

**Models and settings:** We tested only small open models (an LLM judge of at most 4B parameters, zero-shot, and embedding models of up to 600M); larger or fine-tuned LLMs may judge fit better. Embedding inputs were cut at 512 tokens; chunking did not change the results (Section 6.1). Each grid model was trained once (seed 42), so its CIs do not include variation between training runs.

**AI-polished resumes:** The rewrites condensed details, so their content was not identical to the originals; the formatting test, which changes no words, isolates presentation.

**Smaller analysis choices:** The tech-job flag is based on keyword counts and may misclassify some jobs, and the AI/ML subset (Supplementary Section S3) is one exploratory test.

## 9 Ethics

**Purpose:** Automated screening can treat groups of candidates unfairly [31, 32]. We propose no system for hiring decisions, but our findings show that a candidate's rank can depend on which resumes a model has seen, on unreliable labels, or on formatting, rather than on qualifications.

**Data and privacy:** We used a public dataset and made no attempt to identify anyone in it. Of the 643 resumes, 5 contain an email address, 7 a phone number and

17 a web link; we reproduce none of them. The released human labels contain only pair identifiers and labels. The second annotator, a colleague, volunteered without payment, was told how the labels would be used, and is not named.

## 10 Conclusion

Zero-shot embeddings beat keyword methods slightly but significantly (RQ1). Fine-tuning also helps, even on unseen resumes, but the usual job-grouped split inflates the gain (RQ2). The small open LLM judge is not significantly better than embeddings and is far slower (RQ3). The Good vs Potential label reflects the job and the resume more than the match between them, and neither the dataset labels nor one human reader is a reliable reference (RQ4). E5-family models also rank resumes higher when markdown is added, so a candidate can gain rank from formatting rather than suitability (RQ5).

The most important next step is a more reliable benchmark, labelled job by job by several annotators with recruiting experience. The leakage and formatting tests should be repeated on other datasets and with larger LLMs.

**Supplementary information.** The online Supplementary Information gives the prompts (Section S1), the labelling guide (Section S2), additional tables and figures (Section S3) and robustness checks (Section S4).

**Acknowledgements.** The author thanks the colleague who volunteered as the second annotator.

## Declarations

- **Funding:** No funding was received for this work.
- **Competing interests:** The author has no competing interests to declare.
- **Ethics approval and consent to participate:** Not applicable: the study uses a public dataset, and the second annotator volunteered with informed consent (Section 9).
- **Data availability:** The dataset is public [9]. The splits (as hashed IDs, with no resume or job text), human labels and result files are released at https://github.com/nusratrita/resume-job-benchmark-audit.
- **Code availability:** https://github.com/nusratrita/resume-job-benchmark-audit.
- **Use of AI tools:** The author used an AI assistant (Claude, Anthropic) to help with drafting the paper and writing the analysis code. All experiments were run, checked and interpreted by the author, who takes full responsibility for the content.

## References

[1] de Ruijt, C., Bhulai, S.: Job Recommender Systems: A Review. arXiv:2111.13576 (2021)

[2] Fu, Z., Cao, Y., Chen, Y.-L., Lunia, A., et al.: LANTERN: Scalable Distillation of Large Language Models for Job-Person Fit and Explanation. arXiv:2510.05490 (2025)

[3] Zhu, C., Zhu, H., Xiong, H., Ma, C., *et al.*: Person-job fit: Adapting the right talent for the right job with joint representation learning. ACM Transactions on Management Information Systems **9**(3), 1–17 (2018) https://doi.org/10.1145/3234465

[4] Qin, C., Zhu, H., Xu, T., Zhu, C., *et al.*: Enhancing person-job fit for talent recruitment: An ability-aware neural network approach. In: Proceedings of SIGIR 2018, pp. 25–34 (2018). https://doi.org/10.1145/3209978.3210025

[5] Lavi, D., Medentsiy, V., Graus, D.: conSultantBERT: Fine-tuned siamese sentence-BERT for matching jobs and job seekers. In: RecSys in HR Workshop 2021 (2021). arXiv:2109.06501

[6] Yu, X., Zhang, J., Yu, Z.: ConFit: Improving resume-job matching using data augmentation and contrastive learning. In: Proceedings of RecSys 2024, pp. 601–611 (2024). https://doi.org/10.1145/3640457.3688108

[7] Yu, X., Xu, R., Xue, C., Zhang, J., *et al.*: ConFit v2: Improving resume-job matching using hypothetical resume embedding and runner-up hard-negative mining. In: Findings of ACL 2025 (2025)

[8] Yu, X., Xu, R., Xue, C., Chen, J., et al.: ConFit v3: Improving Resume-Job Matching with LLM-based Re-Ranking. arXiv:2605.09760 (2026)

[9] cnamuangtoun: Resume-Job Description Fit. Hugging Face dataset, https://huggingface.co/datasets/cnamuangtoun/resume-job-description-fit. Accessed 2026-09-27 (2024)

[10] Jin, J.: Calibrated resume-job matching for trustworthy LLM-assisted recruiter screening: Pairwise matching, probability calibration, and selective refusal on two public recruitment datasets. Journal of Technology Informatics and Engineering **4**(3) (2025) https://doi.org/10.51903/jtie.v4i3.529

[11] Gururangan, S., Swayamdipta, S., Levy, O., Schwartz, R., Bowman, S.R., Smith, N.A.: Annotation artifacts in natural language inference data. In: Proceedings of NAACL-HLT 2018, pp. 107–112 (2018). https://doi.org/10.18653/v1/N18-2017

[12] Poliak, A., Naradowsky, J., Haldar, A., Rudinger, R., Van Durme, B.: Hypothesis only baselines in natural language inference. In: Proceedings of *SEM 2018, pp. 180–191 (2018). https://doi.org/10.18653/v1/S18-2023

[13] Lewis, P., Stenetorp, P., Riedel, S.: Question and answer test-train overlap in open-domain question answering datasets. In: Proceedings of EACL 2021, pp. 1000–1008 (2021). https://doi.org/10.18653/v1/2021.eacl-main.86

[14] Yang, C., Hou, Y., Song, Y., Zhang, T., *et al.*: Modeling two-way selection preference for person-job fit. In: Proceedings of RecSys 2022 (2022). arXiv:2208.08612

[15] Li, C., Fisher, E., Thomas, R., Pittard, S., *et al.*: Competence-level prediction and resume & job description matching using context-aware transformer models. In: Proceedings of EMNLP 2020 (2020). arXiv:2011.02998

[16] Reimers, N., Gurevych, I.: Sentence-BERT: Sentence embeddings using siamese BERT-networks. In: Proceedings of EMNLP-IJCNLP 2019 (2019). arXiv:1908.10084

[17] Decorte, J.-J., Van Hautte, J., Demeester, T., Develder, C.: JobBERT: Understanding job titles through skills. In: FEAST Workshop at ECML-PKDD 2021 (2021). arXiv:2109.09605

[18] Rosenberger, J., Wolfrum, L., Weinzierl, S., Kraus, M., *et al.*: CareerBERT: Matching resumes to ESCO jobs in a shared embedding space for generic job recommendations. Expert Systems with Applications **275**, 127043 (2025) https://doi.org/10.1016/j.eswa.2025.127043

[19] Sun, W., Yan, L., Ma, X., Wang, S., *et al.*: Is ChatGPT good at search? Investigating large language models as re-ranking agents. In: Proceedings of EMNLP 2023 (2023). arXiv:2304.09542

[20] Zheng, Z., Qiu, Z., Hu, X., Wu, L., et al.: Generative Job Recommendations with Large Language Model. arXiv:2307.02157 (2023)

[21] Ghosh, P., Sadaphal, V.: JobRecoGPT – Explainable job recommendations using LLMs. arXiv:2309.11805 (2023)

[22] Du, Y., Luo, D., Yan, R., Liu, H., *et al.*: Enhancing job recommendation through LLM-based generative adversarial networks. In: Proceedings of AAAI 2024, vol. 38 (2024). https://doi.org/10.1609/aaai.v38i8.28678

[23] Erol, A.K., Yoon, S., Hom, K., Zhang, X.: Synapse: Evolving Job-Person Fit with Explainable Two-phase Retrieval and LLM-guided Genetic Resume Optimization. arXiv:2604.02539 (2026)

[24] Jouanneau, W., Jouffroy, E., Palyart, M.: An Efficient Long-Context Ranking Architecture With Calibrated LLM Distillation: Application to Person-Job Fit. arXiv:2601.10321 (2026)

[25] Singh, I., Garg, A.: Resume ranking with TF-IDF, cosine similarity and named entity recognition. In: Proceedings of ICDCC 2024, pp. 224–229 (2024). https://doi.org/10.1109/ICDCC62744.2024.10961659

[26] Tejaswini, K., Umadevi, V., Kadiwal, S.M., *et al.*: Design and development of machine learning based resume ranking system. Global Transitions Proceedings **3**(2), 371–375 (2022) https://doi.org/10.1016/j.gltp.2021.10.002

[27] Deshmukh, A., Raut, A.B.: Applying BERT-based NLP for automated resume screening and candidate ranking. Annals of Data Science **12**(2), 591–603 (2025) https://doi.org/10.1007/s40745-024-00524-5

[28] Khelkhal, K., Lanasri, D.: Smart-Hiring: An Explainable end-to-end Pipeline for CV Information Extraction and Job Matching. arXiv:2511.02537 (2025)

[29] Northcutt, C.G., Athalye, A., Mueller, J.: Pervasive label errors in test sets destabilize machine learning benchmarks. In: NeurIPS 2021 Datasets and Benchmarks Track (2021). arXiv:2103.14749

[30] Gebru, T., Morgenstern, J., Vecchione, B., Vaughan, J.W., Wallach, H., Daumé III, H., Crawford, K.: Datasheets for datasets. Communications of the ACM **64**(12), 86–92 (2021) https://doi.org/10.1145/3458723

[31] Wilson, K., Caliskan, A.: Gender, race, and intersectional bias in resume screening via language model retrieval. In: Proceedings of AIES 2024, vol. 7, pp. 1578–1590 (2024). https://doi.org/10.1609/aies.v7i1.31748

[32] Armstrong, L., Liu, A., MacNeil, S., Metaxa, D.: The silicon ceiling: Auditing GPT's race and gender biases in hiring. In: Proceedings of EAAMO 2024, pp. 1–18 (2024). https://doi.org/10.1145/3689904.3694699

[33] Iso, H., Pezeshkpour, P., Bhutani, N., Hruschka, E.: Evaluating bias in LLMs for job-resume matching: Gender, race, and education. In: Proceedings of NAACL 2025: Industry Track, pp. 672–683 (2025). https://doi.org/10.18653/v1/2025.naacl-industry.55

[34] Salinas, A., Shah, P., Huang, Y., McCormack, R., Morstatter, F.: The unequal opportunities of large language models: Examining demographic biases in job recommendations by ChatGPT and LLaMA. In: Proceedings of EAAMO 2023, pp. 1–15 (2023). https://doi.org/10.1145/3617694.3623257

[35] Mu, H., Liu, J., Wan, K., Xing, R., et al.: AI Security Beyond Core Domains: Resume Screening as a Case Study of Adversarial Vulnerabilities in Specialized LLM Applications. arXiv:2512.20164 (2025)

[36] Robertson, S., Zaragoza, H.: The probabilistic relevance framework: BM25 and beyond. Foundations and Trends in Information Retrieval **3**(4), 333–389 (2009) https://doi.org/10.1561/1500000019

[37] Wang, L., Yang, N., Huang, X., Jiao, B., Yang, L., Jiang, D., Majumder, R., Wei, F.: Text Embeddings by Weakly-Supervised Contrastive Pre-training. arXiv:2212.03533 (2022)

[38] Xiao, S., Liu, Z., Zhang, P., Muennighoff, N., Lian, D., Nie, J.-Y.: C-Pack: Packed resources for general Chinese embeddings. In: Proceedings of SIGIR 2024, pp.

641–649 (2024). https://doi.org/10.1145/3626772.3657878

[39] Song, K., Tan, X., Qin, T., Lu, J., Liu, T.-Y.: MPNet: Masked and permuted pre-training for language understanding. In: Proceedings of NeurIPS 2020 (2020). arXiv:2004.09297

[40] Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., Zhou, M.: MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In: Proceedings of NeurIPS 2020 (2020). arXiv:2002.10957

[41] Nussbaum, Z., Morris, J.X., Duderstadt, B., Mulyar, A.: Nomic embed: Training a reproducible long context text embedder. Transactions on Machine Learning Research (2025)

[42] Nogueira, R., Cho, K.: Passage Re-ranking with BERT. arXiv:1901.04085 (2019)

[43] Yang, A., Li, A., Yang, B., et al.: Qwen3 Technical Report. arXiv:2505.09388 (2025)

[44] Abouelenin, A., et al.: Phi-4-Mini Technical Report: Compact yet Powerful Multimodal Language Models via Mixture-of-LoRAs. arXiv:2503.01743 (2025)

[45] Wang, L., Yang, N., Huang, X., Yang, L., Majumder, R., Wei, F.: Multilingual E5 Text Embeddings: A Technical Report. arXiv:2402.05672 (2024)

[46] Li, Z., Zhang, X., Zhang, Y., Long, D., Xie, P., Zhang, M.: Towards General Text Embeddings with Multi-stage Contrastive Learning. arXiv:2308.03281 (2023)

[47] Zhang, Y., Li, M., Long, D., Zhang, X., et al.: Qwen3 Embedding: Advancing Text Embedding and Reranking Through Foundation Models. arXiv:2506.05176 (2025)

[48] Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of IR techniques. ACM Transactions on Information Systems **20**(4), 422–446 (2002) https://doi.org/10.1145/582415.582418

[49] Cohen, J.: A coefficient of agreement for nominal scales. Educational and Psychological Measurement **20**(1), 37–46 (1960) https://doi.org/10.1177/001316446002000104

[50] Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics Bulletin **1**(6), 80–83 (1945) https://doi.org/10.2307/3001968

[51] Holm, S.: A simple sequentially rejective multiple test procedure. Scandinavian Journal of Statistics **6**(2), 65–70 (1979)

## Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- RitaJIISsupplementary20261109.pdf