CONFIT: Improving Resume-Job Matching using Data Augmentation and Contrastive Learning

Anonymous ACL submission

Abstract

A reliable resume-job matching system helps a company find suitable candidates from a pool of resumes, and helps a job seeker find relevant jobs from a list of job posts. However, since job seekers apply only to a few jobs, interaction records in resume-job datasets are sparse. Different from many prior work that use complex modeling techniques, we tackle this sparsity problem using data augmentations and a simple contrastive learning approach. CONFIT first creates an augmented resume-job dataset by paraphrasing specific sections in a resume or a job post. Then, CONFIT uses contrastive learning to further increase training samples from B pairs per batch to $\mathcal{O}(B^2)$ per batch. We evaluate CONFIT on two real-world datasets and find it outperforms prior methods (including BM25 and OpenAI text-ada-002) by up to 19% and 31% absolute in nDCG@10 for ranking jobs and ranking resumes, respectively.[1]

1 Introduction

Online recruitment platforms, such as LinkedIn, have over 900 million users, with over 100 million job applications made each month (Iqbal, 2023). With the ever increasing growth of online recruitment platforms, building *fast* and *reliable* person-job fit systems is desiderated. A practical system should be able to quickly select suitable talents and jobs from large candidate pools, and also reliably quantify the "matching degree" between a resume and a job post.

Since both resumes and job posts are often stored as text data, many recent work (Zhu et al., 2018; Qin et al., 2018; Bian et al., 2020; Yang et al., 2022; Shao et al., 2023) focus on designing complex modeling techniques to model resume-job matching (or referred to as "person-job fit"). For example, APJFNN (Qin et al., 2018) uses hierarchical recurrent neural networks to process the job and resume content, and DPGNN (Yang et al., 2022) uses a dual-perspective graph neural network to model the relationship between resumes and jobs. However, these methods only show limited improvements, and they often: optimize only for a single task (e.g., interview classification); are hard to accommodate new, unseen resumes or jobs; and are designed for a particular data setting (e.g., applicable only if a specific recruitment platform is used).

In this work, we present a simple method to model resume-job matching, with strong performances in both resume/job ranking *and* resume-job pair classification tasks. We propose CONFIT, an approach to learn high-quality dense embeddings for resumes and jobs, which can be combined with techniques such as FAISS (Johnson et al., 2019) to rank tens of thousands of resumes and jobs in milliseconds. To combat label sparsity in person-job fit datasets, CONFIT first uses data augmentation techniques to increase the number of training samples, and then uses contrastive learning (Karpukhin et al., 2020; Wang et al., 2023) to train an encoder. We evaluate CONFIT on two resume-job matching datasets and find our approach outperforms previous methods (including strong baselines from information retrieval such as BM25) in almost all ranking *and* classification tasks, with up to 20-30% absolute improvement in ranking jobs and resumes.

2 Background

A resume-job matching (or often called a *person-job fit*) system models the suitability between a resume and a job, allowing it to select the most suitable candidates given a job post, or recommend the most relevant jobs given a candidate's resume (Bian et al., 2020; Yang et al., 2022; Shao et al., 2023). A job post J (or a resume R) is commonly structured as a collection of texts $J = \{\mathbf{x}_i^J\}_{i=1}^p$, where each piece of text may represent certain sections of the document, such as "Required Skills" for a job post or "Experiences" for a resume (Bian

[1] We will release our code upon acceptance.

et al., 2019; Shao et al., 2023). Thus, the person-job fit problem is often formulated as a text-based task to model a "matching" score:

$$\text{match}(R, J) = f_\theta(R, J) \to \mathbb{R}$$

where f_θ typically involves neural networks (Zhu et al., 2018; Yang et al., 2022; Shao et al., 2023) such as BERT (Devlin et al., 2019).

With the ever-increasing growth of online recruitment data, there is a large number of job posts and resumes (privately) available. However, since a candidate applies to only a small selection of jobs, interactions between resumes and jobs is *very sparse* (Bian et al., 2020). Often, the resulting dataset $\mathcal{D} = \{R_i, J_i, y_i\}$ has size $|\mathcal{D}| \ll n_R \times n_J$, where n_R and n_J are the total number of resumes and jobs respectively, and $y_i \in \{0, 1\}$ is a *binary* signal representing whether a resume R_i is accepted for an interview by a job J_i. While some private datasets (Yang et al., 2022) may contain additional labels, such as whether the candidate or recruiter "requested" additional information from the other party, we focus on the more common case where only a binary signal is available.

3 Approach

We propose ConFit, a simple and general-purpose approach to model resume-job matching using contrastive learning and data augmentation. ConFit produces a dense embedding of a given resume or job post, and models the matching score between an $\langle R, J \rangle$ pair as the inner product of their representations. This simple formulation allows ConFit to quickly rank a large number of resumes or jobs when combined with retrieval techniques such as FAISS (Johnson et al., 2019). In Section 3.1, we describe our approach to augment a person-job fit dataset, and in Section 3.2 we describe the contrastive learning approach used during training.

3.1 Data Augmentation

A person-job fit dataset may be considered as a sparse bipartite graph, where each resume R_i and job J_i is a node, and a label (accept or reject) is an edge between the two nodes. Given a resume R_i, we first create augmented versions $\hat{R}_i$ by paraphrasing certain sections such as "Experiences" (see Appendix E for more details). Since $\hat{R}_i$ includes semantically similar information as R_i, we inherit the *same edges* from R_i to $\hat{R}_i$, i.e., any job J_j that accepts R_i for interview also accepts $\hat{R}_i$. Next, we perform the same augmentation procedure for jobs, creating $\hat{J}_i$ that inherits the same edges as J_i in the graph (which involves augmented $\hat{R}_i$s). As a result, augmenting n_{aug} resumes and jobs each *for once* approximately *doubles the number of labeled pairs* (often $\gg n_{\text{aug}}$) in the dataset. ConFit thus first performs data augmentation to increase the number of labeled pairs, and then uses contrastive learning (Section 3.2) to train a high-quality encoder. Below, we briefly describe paraphrasing methods used in this work.

EDA Augmentation Given a piece of text from a resume or a job, we use EDA (Wei and Zou, 2019) to randomly replace, delete, swap, or insert words to create a paraphrased version of the text. We find this to be a simple and fast method to create semantically similar text.

ChatGPT Augmentation Besides EDA, we also use ChatGPT (OpenAI, 2022b) to perform paraphrasing. ChatGPT has been used on many data augmentation tasks (Cegin et al., 2023; Dai et al., 2023), and in this work, we similarly prompt ChatGPT to paraphrase a given piece of text (see Appendix E for more details).

3.2 Contrastive Learning

Given an augmented dataset, ConFit uses contrastive learning (Chen et al., 2020; Wang et al., 2023) to further increase the number of training instances from B per batch to $\mathcal{O}(B^2)$ per batch. Contrastive learning is also an effective technique for learning a high-quality embedding space, and is used in various domains such as information retrieval (Karpukhin et al., 2020) and representation learning (Chen et al., 2020; Wang et al., 2023).

First, we construct contrastive training instances from a dataset $\mathcal{D} = \{R_i, J_i, y_i\}$:

$$\mathcal{D}_{\text{con}} = \{\langle R_i^+, J_i^+, R_{i,1}^-, ..., R_{i,l}^-, J_{i,1}^-, ..., J_{i,l}^- \rangle\},$$

where each instance contains one positive pair of matched resume-job $\langle R_i^+, J_i^+ \rangle$ with $y_i = 1$, and l unsuitable resumes $R_{i,l}^-$ for a job J_i^+ as well as l unsuitable jobs $J_{i,l}^-$ for a resume R_i^+.[2] Following prior work in contrastive learning (Chen et al., 2020; Gao et al., 2021; Wang et al., 2023), we

[2]Different from information retrieval (Karpukhin et al., 2020; Wang et al., 2022) where ranking is an asymmetric task (given a query, rank passages), the person-job fit problem is symmetric (given a resume, rank jobs, and vice versa).

optimize the following cross-entropy loss:

$$\mathcal{L} = \mathcal{L}_R + \mathcal{L}_J \quad (1)$$

$$\mathcal{L}_R = -\log \frac{e^{s_\theta(R_i^+, J_i^+)}}{e^{s_\theta(R_i^+, J_i^+)} + \sum_{j=1}^{l} e^{s_\theta(R_i^+, J_{i,j}^-)}}$$

$$\mathcal{L}_J = -\log \frac{e^{s_\theta(R_i^+, J_i^+)}}{e^{s_\theta(R_i^+, J_i^+)} + \sum_{j=1}^{l} e^{s_\theta(R_{i,j}^-, J_i^+)}}$$

Similar to training retrieval systems (Karpukhin et al., 2020), we find the number and choice of negative samples important to obtain a high-quality encoder. We discuss how ConFit chooses negative samples below.

In-batch negatives Let there be B positive pairs $\{\langle R_1^+, J_1^+\rangle, ..., \langle R_B^+, J_B^+\rangle\}$ in a mini-batch during training. For each resume R_i^+, we use the other $B-1$ jobs $\{J_{j\neq i}^+\}$ as negative samples, and similarly for each job J_i^+, we use the other $B-1$ resumes as negative samples. The trick of in-batch negatives thus trains on B^2 resume-job pairs in each batch, and is highly computationally efficient (Gillick et al., 2019; Karpukhin et al., 2020; Wang et al., 2022). In person-job fit, this has a natural interpretation that random (in-batch) negative samples are unsuitable resumes/jobs for a given job/resume. In practice, we find that using in-batch negatives alone is sufficient to yield competitive ranking performances compared to prior approaches (see Section 4.7).

Hard negatives In addition to in-batch negative samples, we also sample up to $2 \times B_{\text{hard}}$ hard negatives for each batch to further improve ConFit training. In information retrieval systems, hard negatives (Karpukhin et al., 2020; Wang et al., 2022) are often passages that are relevant to the query (e.g. have a high BM25 (Robertson and Zaragoza, 2009) score) but do not contain the correct answer. In person-job fit, we believe that this extends to resumes/jobs that are explicitly *rejected* for a given job/resume. This is because often when a candidate *submits* a resume for a given job post, the resume is *already highly relevant* regardless of whether the candidate is accepted or rejected. Thus, we sample up to B_{hard} rejected resumes as hard negatives for any of the B jobs in the mini-batch, as well as B_{hard} jobs that rejected any of the B resumes. These $2 \times B_{\text{hard}}$ hard negatives are then used by all resumes/jobs in the batch, increasing the number of training pairs to $(B + B_{\text{hard}})^2 - B_{\text{hard}}^2$ per batch.

3.3 ConFit

To address the label sparcity problem in person-job fit datasets, ConFit first augments the dataset using techniques introduced in Section 3.1. Then, ConFit trains an encoder network E_θ using contrastive learning described in Section 3.2. Given resumes and job posts during inference, ConFit first uses the encoder E_θ to obtain a dense representation for each resume R and job J. Then, ConFit produces a matching score s_θ between the $\langle R, J\rangle$ pair using inner product:

$$\text{match}(R, J) = E_\theta(R)^T E_\theta(J) \equiv s_\theta(R, J)$$

This simple formulation allows ConFit to combined with techniques such as FAISS (Johnson et al., 2019) to efficiently rank tens of thousands of resumes and jobs in milliseconds (Section 4.6).

4 Experiments

We evaluate ConFit on two real-world person-job fit datasets, and measure its performance and runtime on ranking resumes, ranking jobs, as well as on a fine-grained interview classification task.

4.1 Dataset and Preprocessing

AliYun Dataset To our knowledge, the 2019 Alibaba job-resume intelligent matching competition[3] *provided* the only publicly available person-job fit dataset. All resume and job posts were desensitized and were already parsed into a collection of text fields, such as "Education", "Age", and "Work Experiences" for a resume (see Appendix A for more details). All resumes and jobs are in Chinese.

Intellipro Dataset The resumes and job posts are collected from a global hiring solution company, called "Intellipro Group Inc."**??**. To protect the privacy of candidates, all records have been anonymized by removing sensitive identity information. For each resume-job pair, we record whether the candidate is accepted ($y = 1$) or rejected ($y = 0$) for an interview. For generalizability, we parse all resumes and jobs into similar sections/fields as the AliYun dataset. Both English and Chinese resumes and jobs are included.

Since neither dataset has an official test set, we first construct test sets with statistics shown in Table 2. To measure the ranking ability of current methods, we consider two tasks: 1) ranking $q = 100$ resumes given a job post (denoted as *Rank*

[3] https://tianchi.aliyun.com/competition/entrance/231728

Train	Intellipro Dataset	Aliyun Dataset
# Jobs	1794	19542
# Resumes	6435	2718
# Labels	6751	22124
(# accept)	2809	10185
(# reject)	3942	11939
# Industries	16	20
# Fields per R	8	12
# Fields per J	9	11
# Words per R	915.2	101.9
# Words per J	174.7	153.1

Table 1: Training dataset statistics. *# Words per R/J* represent the *average* number of words per resume/job.

	Intellipro Dataset			AliYun Dataset		
Test	Rank R	Rank J	Classify	Rank R	Rank J	Classify
# Samples	120	120	120	300	300	300
# Jobs	120	427	104	300	2903	299
# Resumes	1154	120	117	1006	300	280

Table 2: Test dataset statistics. *Classify* is a binary classification task to predict whether a resume-job pair is accepted or rejected for interview.

Resume), and 2) ranking $q = 100$ jobs given a resume (denoted as *Rank Job*). Since only a few resumes and jobs are labeled, we fill in random resumes/jobs to reach q slots when needed. We further consider the "fine-grained" scoring ability of current methods, by measuring how well a method can distinguish between an accepted resume-job pair and a rejected one (denoted as *classification*). We exclude all resumes and jobs used in test and validation sets from the training set, and present the training, test, and validation set statistics in Table 1, Table 2 and Table A1, respectively.

4.2 Model Architecture

Since both datasets represent resumes and job posts as a collection of text fields, we simplify the model architecture from InEXIT (Shao et al., 2023), outlined in Figure 1. InEXIT encodes each text field (e.g., "education: Bachelor;...") in a resume or a job independently using a pre-trained encoder, and considers a hierarchical attention mechanism to model person-job fit as interactions between these fields. Following InEXIT, we first encode each field independently, and model the "internal interaction" between the fields *within* a resume/job using attention (Vaswani et al., 2023). InEXIT then uses another attention layer on all text fields of the resume-job pair to model the "external interaction" *between* a resume and a job post, and finally produces a score using an MLP layer (see Appendix B for more details). Since ConFit models person-job fit based on *independently* produced resume/job embeddings, we replace the last attention and MLP layer with a linear layer, which directly fuses the field representations into a single dense vector for a given resume or a job post (Figure 1).

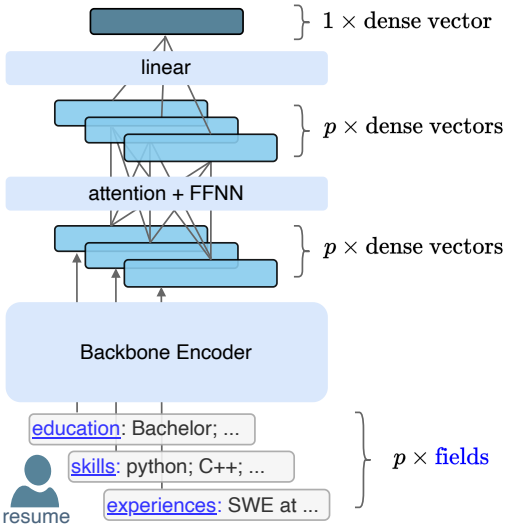


Figure 1: Model architecture used to encode a resume or a job post, formatted as a collection of p text fields (see Appendix A for a full example of resume/job).

4.3 Baselines

We compare ConFit against both recent best person-job fit systems and strong baselines from information retrieval systems.

Recent person-job fit systems can be grouped into two categories: classification-targeted and ranking-targeted. The **best** classification-targeted system include *MV-CoN* (Bian et al., 2020) and *InEXIT* (Shao et al., 2023). MV-CoN considers a co-teaching network (Han et al., 2018) to learn from sparse, noisy person-job fit data, and InEXIT uses hierarchical attention to model interactions between the text fields of a resume-job pair. Both methods optimize for the classification task. The **best** ranking-targeted systems include *DPGNN* (Yang et al., 2022). DPGNN considers a dual-perspective graph view of person-job fit and uses a BPR loss (Rendle et al., 2012) to optimize for resume and job ranking.

We also compare against methods from information retrieval systems such as: *BM25* (Robertson and Zaragoza, 2009; Trotman et al., 2014) and *RawEmbed*. BM25 is a strong baseline used for many text ranking tasks (Thakur et al., 2021; Wang et al., 2022; Kamalloo et al., 2023), and RawEm-

		Intellipro Dataset							AliYun Dataset						
		Rank Resume		Rank Job		Classification			Rank Resume		Rank Job		Classification		
Method	**Encoder**	MAP	nDCG	MAP	nDCG	F1	Prc+	Rcl+	MAP	nDCG	MAP	nDCG	F1	Prc+	Rcl+
XGBoost	BoW	14.42	11.84	3.89	2.98	49.51	37.74	41.67	6.67	6.25	13.63	12.53	69.57	72.00	54.14
	TF-IDF	10.18	7.87	4.11	2.94	55.00	43.75	43.75	5.56	4.96	13.85	13.21	68.78	69.16	55.64
	BERT-base	8.45	7.69	5.57	5.24	61.01	50.98	**54.17**	5.34	5.10	13.45	12.22	70.63	**73.27**	55.64
RawEmbed.	E5-small	28.61	33.88	25.48	30.26	54.20	42.86	31.25	16.06	17.89	20.26	22.84	38.64	27.78	22.56
	BERT-base	13.07	13.94	4.41	3.62	49.25	34.38	22.92	9.18	10.63	12.35	12.63	46.71	40.00	40.60
MV-CoN	BERT-base	10.81	10.00	3.34	2.17	58.00	50.00	33.33	5.41	5.15	13.44	12.67	**74.25**	*72.22*	*68.32*
InEXIT	BERT-base	12.27	12.98	4.11	3.46	55.55	44.74	35.42	5.25	4.98	13.02	12.30	*71.75*	66.67	**72.18**
DPGNN	BERT-base	19.64	21.95	17.86	19.60	*61.16*	*52.38*	45.83	19.96	24.64	27.23	30.07	50.31	45.24	57.14
BM25	-	*39.13*	*44.96*	*37.88*	*43.15*	-	-	-	**34.71**	**40.56**	*27.30*	*31.18*	-	-	-
Ours	BERT-base	**44.47**	**49.51**	**39.57**	**45.67**	**63.78**	**55.81**	*50.00*	*30.79*	*37.71*	**36.13**	**41.65**	47.16	41.10	45.11

Table 3: Comparing ranking and classification performance of various approaches when a small encoder is used. *F1* is weighted F1 score, *nDCG* is nDCG@10, *Prc+* and *Rcl+* are precision and recall for positive classes. Results for non-deterministic methods are averaged over 3 runs. Best result is shown in **bold**, and runner-up is in *gray*.

		Intellipro Dataset							AliYun Dataset						
		Rank Resume		Rank Job		Classification			Rank Resume		Rank Job		Classification		
Method	**Encoder**	MAP	nDCG	MAP	nDCG	F1	Prc+	Rcl+	MAP	nDCG	MAP	nDCG	F1	Prc+	Rcl+
XGBoost	BoW	14.42	11.84	3.89	2.98	49.51	37.74	41.67	6.67	6.25	13.63	12.53	69.57	72.00	54.14
	TF-IDF	10.18	7.87	4.11	2.94	55.00	43.75	43.75	5.56	4.96	13.85	13.21	68.78	69.16	*55.64*
	text-ada-002	9.87	9.94	4.15	3.58	**62.43**	**53.19**	*52.08*	6.40	6.08	13.46	12.93	62.43	53.19	52.08
RawEmbed.	xlm-roberta-l	14.46	14.95	13.22	13.94	51.27	40.00	16.67	7.07	6.90	11.25	10.75	53.48	47.62	52.63
	E5-large	35.10	40.10	26.93	30.61	*59.39*	*51.43*	37.50	32.11	37.45	24.56	28.15	44.67	35.64	27.07
	text-ada-002	*42.85*	*48.11*	**43.28**	*51.11*	58.92	48.89	45.83	31.47	37.06	21.94	24.80	39.72	32.35	33.08
MV-CoN	E5-large	12.23	10.98	4.06	3.09	52.75	40.51	27.08	5.60	5.15	12.92	12.67	**70.85**	**77.53**	51.88
InEXIT	E5-large	13.60	13.63	3.14	1.91	55.17	45.16	29.17	5.49	4.33	13.39	13.21	*70.41*	*74.74*	53.58
DPGNN	E5-large	21.31	24.08	13.90	17.69	55.56	48.00	25.00	33.98	40.63	*42.76*	*46.98*	53.88	48.03	45.86
BM25	-	39.13	44.96	37.88	43.15	-	-	-	*34.71*	*40.56*	27.30	31.18	-	-	-
Ours	E5-large	**43.08**	**50.28**	*42.88*	**51.74**	54.09	45.21	**68.75**	**64.32**	**71.71**	**59.13**	**65.91**	59.71	53.66	**66.17**

Table 4: Comparing ranking and classification performance of various approaches when a larger encoder is used. *F1* is weighted F1 score, *nDCG* is nDCG@10, *Prc+* and *Rcl+* are precision and recall for positive classes. Results for non-deterministic methods are averaged over 3 runs. Best result is shown in **bold**, and runner-up is in *gray*.

bed is based on dense retrieval methods (Karpukhin et al., 2020; Johnson et al., 2019) that directly concatenates all text fields and uses a pre-trained encoder to produce a single dense embedding for inner product scoring. Finally, we also consider *XGBoost* (Chen and Guestrin, 2016) as a generic method for classification and ranking tasks, where features can be Bag-of-Words (BoW), TF-IDF vectors, and pre-trained embeddings from RawEmbed.

Unless otherwise indicated, ConFit first uses data augmentation with both EDA and ChatGPT, each augmenting 500 resumes and 500 jobs for each dataset (Section 3.1), followed by contrastive learning with $B = 8$ and $B_{\text{hard}} = 8$ (Section 3.2). See Appendix D for other hyperparameters used by ConFit, and see Appendix C for more implementation details of the baselines.

4.4 Metrics

Following prior work (Karpukhin et al., 2020; Yang et al., 2022), we use Mean Average Precision (MAP) and normalized Discounted Cumulative Gain (nDCG) to measure the ranking ability of each method. Since most resume-job pairs are unlabeled, we report nDCG@10. To measure the fine-grained classification ability of a method, we follow prior work in person-job fit (Qin et al., 2018; Zhu et al., 2018; Bian et al., 2020; Shao et al., 2023) and use weighted F1, precision, and recall. Since correctly predicting a positive sample (i.e., a suitable job for a resume) is important in practice, we report precision and recall for the positive class (denoted as *Prc+* and *Rcl+*, respectively).

4.5 Main Results

Table 3 summarizes ConFit's performance in comparison to other baselines, when an encoder with ∼180M parameters is used as the backbone. This includes using BERT-base[4] (Devlin et al., 2019) and E5-small (Wang et al., 2022). In general, we find that classification-targeted systems

[4]Since the AliYun dataset is solely in Chinese, we use BERT-base-chinese for the AliYun dataset and BERT-base-multilingual-cased for the Intellipro dataset.

such as *MV-CoN* and *InEXIT* achieve a high F1 score but have poor ranking ability, while ranking-targeted methods such as *RawEmbed*, *DPGNN*, and *BM25* perform much better in ranking. With a small encoder model, we find CONFIT achieves the best ranking performance in three out of the four tasks, and BM25 achieves the best in the remaining task. CONFIT also achieves the best F1 score on the Intellipro classification task compared to other classification-targeted systems.

Table 4 summarizes each method's performance when a larger backbone encoder (∼560M parameters) is used. This includes multilingual-E5-large (Wang et al., 2022), xlm-roberta-large (Conneau et al., 2019; Liu et al., 2019), or OpenAI text-ada-002[5] (OpenAI, 2022a). Similar to Table 3, we find that classification-targeted methods such as *MV-CoN* reach a high F1 score, while ranking-targeted methods achieve a better MAP and nDCG score. We also find that CONFIT now achieves the best ranking performances in all cases, except for the MAP score in the IntelliPro's job ranking task. We believe this is because the IntelliPro dataset contains much less data compared to the AliYun dataset (Table 1). In the AliYun dataset, CONFIT improves up to ∼30% absolute in MAP and nDCG score for ranking resumes and up to ∼20% for ranking jobs. We believe this is because the AliYun dataset not only has more data, but also uses much shorter and concise texts compared to the Intellipro dataset (Table 1). Lastly, we find CONFIT remains competitive in classification task for both datasets, despite not directly optimizing for them.

4.6 Runtime Analysis

A practical recruitment system needs to *quickly* rank a large number of resumes given a job post, or vice versa. We measure the runtime to rank 100; 1,000; and 10,000 jobs for a given resume from the AliYun dataset, and compare the speed of various neural-based methods from Table 3. We present the results in Figure 2. In general, methods that ranks by inner product search (*RawEmbed* and CONFIT) can utilize FAISS (Johnson et al., 2019) to achieve a runtime in milliseconds in all cases[6]. However, methods such as *MV-CoN*, *InEXIT*, and *DPGNN* requires a (partial) forward pass for each resume-job pair to produce a score between (see Appendix F for more details). We believe this is highly inefficient, especially when the number of documents to rank (e.g., job posts) is large.

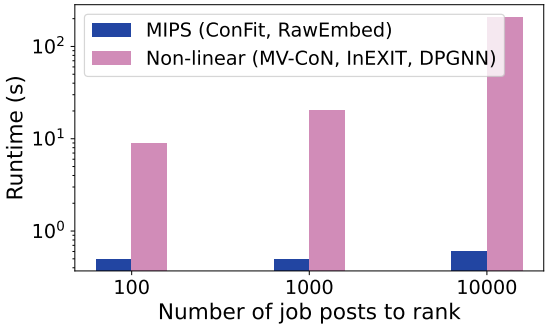


Figure 2: Runtime comparison between neural-based methods. *MIPS* are maximum inner product search methods that are supported by FAISS (Johnson et al., 2019). *Non-linear* methods require an additional forward pass to produce a score between a resume-job pair. Results are averages over three runs.

4.7 Ablation Studies

Table 5 presents our ablation studies for each component of CONFIT training. We focus on using BERT-base from Table 3 as it is less resource-intensive to train.

First, we consider CONFIT to only use contrastive learning (denoted as *+contrastive*) under various settings, such as $B = 8, B_{\text{hard}} = \{0, 2, 4, 8\}$. In Table 5, we find that: a) increasing the number of hard negatives (B_{hard}) improves ranking performance, and b) using contrastive learning alone already outperforms many baselines in Table 3. This suggests that contrastive learning plays a major role in CONFIT's performance.

Next, we add data augmentation to training, and measure the performance of: 1) using only ChatGPT to augment 500 resumes and jobs, denoted as *ChatGPT only*; 2) using EDA to augment 500 resumes and jobs, denoted *EDA only*; 3) using EDA to augment all resume/job seen during training, denoted as *EDA-all*; and 4) combining both 1) and 2), denoted as *+Data Aug*. In general, we find combining both ChatGPT and EDA augmentation can most often achieve the best performance. We believe this is because such approach includes both *semantically paraphrased* content from ChatGPT and *syntactically altered* content (e.g., inserting or removing words) from EDA. Especially for the AliYun dataset, we find using any form of data augmentation improves over using contrastive learning alone. We believe this is because AliYun's resume/job texts are much shorter and more concise

[5]Model size unknown.

[6]After embedding all relevant resume and job posts, which only needs to be computed *once*.

	Intellipro Dataset							AliYun Dataset						
	Rank Resume		Rank Job		Classification			Rank Resume		Rank Job		Classification		
Modification	MAP	nDCG	MAP	nDCG	F1	Prc+	Rcl+	MAP	nDCG	MAP	nDCG	F1	Prc+	Rcl+
+contrastive	**42.96**	**49.28**	**42.23**	**49.21**	59.17	56.00	29.17	**27.53**	**33.34**	**34.05**	**37.83**	**47.60**	**40.77**	39.85
$B_{\text{hard}} = 0$	31.15	36.88	38.94	44.89	52.02	44.16	**70.83**	13.73	15.43	32.52	37.23	43.31	34.51	29.32
$B_{\text{hard}} = 2$	41.85	47.77	41.24	47.46	56.14	56.58	19.79	25.42	31.08	31.59	35.64	45.65	38.64	38.35
$B_{\text{hard}} = 4$	40.86	45.69	41.29	47.78	**61.69**	**62.36**	33.33	25.60	30.65	32.72	36.15	44.51	38.67	**43.61**
+Data Aug.	**44.47**	**49.51**	39.57	45.67	**63.78**	55.81	**50.00**	**30.79**	**37.71**	**36.13**	**41.65**	**47.16**	**41.10**	**45.11**
ChatGPT only	41.36	47.45	39.54	**46.98**	58.75	**71.43**	20.83	29.69	35.59	35.11	40.13	47.00	39.67	36.09
EDA only	39.49	46.03	**40.25**	46.03	60.64	65.00	27.08	27.06	33.08	34.69	39.16	46.30	38.39	32.33
EDA-all	39.03	45.56	40.09	45.76	60.64	65.00	27.08	28.21	33.77	33.77	38.13	45.59	37.82	33.83

Table 5: ConFit ablation studies. ConFit uses contrastive learning (+*contrastive*) with $B_{\text{hard}} = 8$, and Data Augmentation (+*Data Aug.*) with both ChatGPT and EDA. Best result in each ablation group is highlighted in **bold**.

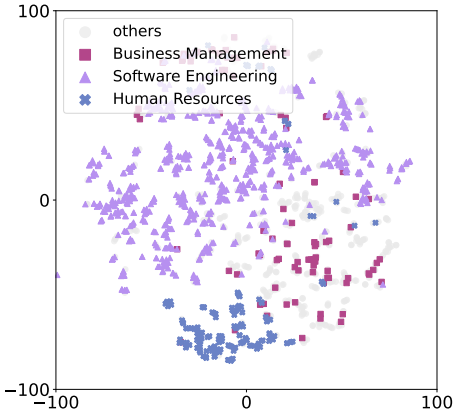


Figure 3: Visualizing resume embeddings from ConFit using t-SNE. Colors are assigned using each resume's desired industry. Top-3 most frequent industries are color-coded for easier viewing.

than those of the Intellipro dataset, thus making data augmentation easier to perform.

Since ConFit training is model-agnostic, we also experiment with *completely removing neural networks*, and only use TF-IDF representations with XGBoost. Despite seeing performance degradation compared to ConFit with pretrained encoders, we find this approach is *still competitive* against prior best person-job fit systems that uses BERT (see Appendix G and Table A4 for more details). This suggests that the contrastive learning and data augmentation *procedure* from ConFit is effective for the person-job fit task.

5 Analysis

In this section, we provide both qualitative visualization and quantitative analysis of the embeddings learned by ConFit. We mainly focus on the Intellipro dataset as it is more challenging.

5.1 Qualitative Analysis

ConFit aims to learn a high-quality embedding space for a resume or a job post. In Figure 3 we visualize the resume embeddings learned by ConFit. We use ConFit with BERT-base (see Table 3) to embed all 1457 resumes from the test set in the Intellipro dataset, and perform dimensionality reduction using t-SNE (van der Maaten and Hinton, 2008). In Figure 3, we find ConFit learned to cluster resumes based on important fields such as "Desired Industry". We believe this is consistent with how a human would determine person-job fit, as resumes aiming for similar industries are likely to contain similar sets of experiences and skills. For comparison with embeddings generated by other baselines, please see Appendix H.

5.2 Error Analysis

To analyze the errors made by ConFit, we manually inspect 30 *negative* resume-job pairs from the ranking tasks that are *incorrectly ranked at top 5%* and is before at least one positive pair, and 20 pairs from the classification task that was incorrectly predicted as a match. For each incorrectly ranked or classified pair, we compare against other positive resume-job pairs from the dataset, and categorize the errors with the following criteria: *unsuitable*, where some requirements in the job post are not satisfied by the resume; *less competent*, where a resume satisfies all job requirements, but many competing candidates have a higher degree/more experience; *out-of-scope*, where a resume satisfies all requirements, appears competitive compared to other candidates, but is still rejected due to other (e.g., subjective) reasons not presented in our resume/job data themselves; and *potentially suitable*, where a resume from the ranking tasks satisfied the requirements and seemed competent, but had no label in the original dataset.

We present our analysis in Figure 4, and find that a significant portion of errors are *out-of-scope*, where we believe information in resumes/job posts

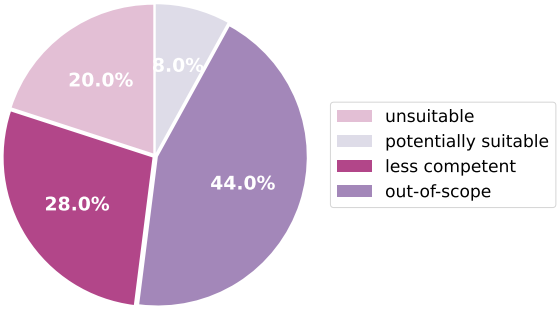


Figure 4: ConFit error analysis. We find 44% of the errors made are due to reasons not identifiable using resume/job documents alone, and 28% due to a candidate's resume satisfying all the job requirements but is less competent than *other competing candidates*.

is limited. The next most frequent error is *less competent*, which is understandable since ConFit scores a resume-job pair independent of other competing candidates. Lastly, we also find that about 20% of the wrong predictions were *unsuitable*, with resumes not satisfying certain job requirements such as "4 years+ with Docker, K8s". We believe *unsuitable* errors may be mitigated by combining ConFit with better feature engineering techniques along with keyword-based approaches (such as BM25), which we leave for future work.

6 Related Work

Person-job fit systems Early neural-based methods in person-job fit (Guo et al., 2016) typically focus on network architecture to obtain a good representation of a job post or a resume. These methods include Qin et al. (2018); Zhu et al. (2018); Rezaeipourfarsangi and Milios (2023); Jiang et al. (2020); Mhatre et al. (2023), which explores architectures such as RNN, LSTM (Staudemeyer and Morris, 2019) and CNN (O'Shea and Nash, 2015). Recent deep learning methods include Maheshwary and Misra (2018); Rezaeipourfarsangi and Milios (2023), which uses deep siamese network to learn an embedding space for resume/jobs, Bian et al. (2019) which uses a hierarchical RNN to improve domain-adaptation of person-job fit systems, and Zhang et al. (2023) which uses federated learning to perform model training while preserving user privacy. However, as person-job fit systems involve sensitive data, most systems do not open-source datasets *or implementations*, and are often optimized for one particular dataset. Recent work with public implementations includes MV-CoN (Bian et al., 2020), which uses a co-teaching network (Malach and Shalev-Shwartz, 2018) to perform gradient updates based model's confidence to data noises; InEXIT (Shao et al., 2023), which uses hierarchical attention to model resume-job interactions; and DPGNN (Yang et al., 2022), which uses a graph-based approach with a novel BPR loss to optimize for resume/job ranking. ConFit uses contrastive learning and data augmentation techniques based on powerful pre-trained models such as BERT (Devlin et al., 2019), and achieves the best performance in almost all ranking and classification tasks across two person-job fit datasets.

Information retrieval systems ConFit benefits from contrastive learning techniques, which have seen wide applications in many information retrieval and representation learning tasks (Chen et al., 2020; Radford et al., 2021). Given a query (e.g., user-generated question), an information retrieval system aims to find top-k relevant passages from a large reserve of candidate passages (Joshi et al., 2017; Kwiatkowski et al., 2019). Popular methods in information retrieval include BM25 (Robertson and Zaragoza, 2009; Trotman et al., 2014), a keyword-based approach used as the baseline in many text ranking tasks (Nguyen et al., 2016; Thakur et al., 2021; Muennighoff et al., 2022), and dense retrieval methods such as Karpukhin et al. (2020); Izacard et al. (2021); Wang et al. (2022), which uses contrastive learning to obtain high-quality passage embeddings and typically performs top-k search based on inner product. To our knowledge, ConFit is the first attempt to use contrastive learning for person-job fit, achieving the best performances in almost all person-job ranking tasks across two different person-job fit datasets.

7 Conclusion

We propose ConFit, a general-purpose approach to model person-job fit. ConFit trains a neural network using contrastive learning to obtain a high-quality embedding space for resumes and job posts, and uses data augmentation to alleviate data sparsity in person-job fit datasets. Our experiments across two person-job fit datasets show that ConFit achieves the best performance in almost all ranking and classification tasks. We believe ConFit is easily extensible, and can be used as a strong foundation for future research on person-job fit.

8 Limitations

Recruiter/Job Seeker Preference CONFIT produces dense representations for resumes and jobs *independently*, and uses inner-product to score the resume-job pair. While this approach can be easily combined with retrieval methods such as FAISS (Johnson et al., 2019) to efficiently rank a large number of resumes/jobs, it ignores certain aspects of how a real recruiter or a job seeker may choose a resume or a job. In our error analysis (Section 5.2), we find a significant portion of incorrectly ranked/rated resume-job pairs *could* be either due to subjective choices made by the recruiters, or due to a very competitive candidate pool for a certain job position. This suggests that additionally modeling the recruiter or job seeker's past preferences (e.g., using profiling approaches (Yan et al., 2019) from recommendation systems (Eliyas and Ranjana, 2022)) may be beneficial, and that developing a scoring metric that is *aware of the other candidates* in the pool could also be useful. In general, we believe CONFIT embeddings would serve as a foundation for these approaches, and we leave this for future work.

Sensitive Data To our knowledge, there is no standardized, public person-job fit dataset[7] that can be used to compare performances of existing systems (Zhu et al., 2018; Qin et al., 2018; Bian et al., 2020; Yang et al., 2022; Shao et al., 2023). This is understandable, as resume contents contain highly sensitive information and that large-scale person-job datasets are often proprietary. We provide our best effort to make CONFIT reproducible and extensible for future work: we will open-source full implementations of CONFIT and all relevant baselines, our data processing scripts, and dummy train/valid/test data files that can be used test drive our system end-to-end. We will also privately release our model weights and full datasets to researchers under appropriate license agreements. We hope these attempts can make future research in person-job fit more accessible.

9 Ethical Considerations

CONFIT uses pretrained encoders such as BERT and E5 (Devlin et al., 2019; Wang et al., 2022), and it is well-known that many powerful encoders contain biases (Brunet et al., 2019; May et al., 2019; Jentzsch and Turan, 2022; Caliskan et al., 2022). For person-job fit systems, we believe it is crucial to ensure that the systems do **not** bias towards certain groups of people, such as preferring a certain gender for certain jobs. Although both datasets used in this work already removed any sensitive information such as gender, we do not recommend directly deploying CONFIT for real-world applications without using debiasing techniques such as Bolukbasi et al. (2016); Cheng et al. (2021); Gaci et al. (2022); Guo et al. (2022); Schick et al. (2021), and we do not condone the use of CONFIT for any morally unjust purposes. To our knowledge, there is little work on investigating or mitigating biases in existing person-job fit systems, and we believe this is an important direction for future work.

[7] The AliYun dataset used in this work is no longer publicly available as of 09-11-2023.

References

Shuqing Bian, Xu Chen, Wayne Xin Zhao, Kun Zhou, Yupeng Hou, Yang Song, Tao Zhang, and Ji-Rong Wen. 2020. Learning to match jobs with resumes from sparse interaction data using multi-view co-teaching network.

Shuqing Bian, Wayne Xin Zhao, Yang Song, Tao Zhang, and Ji-Rong Wen. 2019. Domain adaptation for person-job fit with transferable deep global match network. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4810–4820, Hong Kong, China. Association for Computational Linguistics.

Tolga Bolukbasi, Kai-Wei Chang, James Zou, Venkatesh Saligrama, and Adam Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings.

Dorian Brown. 2020. Rank-BM25: A Collection of BM25 Algorithms in Python.

Marc-Etienne Brunet, Colleen Alkalay-Houlihan, Ashton Anderson, and Richard Zemel. 2019. Understanding the origins of bias in word embeddings. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 803–811. PMLR.

Aylin Caliskan, Pimparkar Parth Ajay, Tessa Charlesworth, Robert Wolfe, and Mahzarin R. Banaji. 2022. Gender bias in word embeddings: A comprehensive analysis of frequency, syntax, and semantics. In *Proceedings of the 2022 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '22. ACM.

Jan Cegin, Jakub Simko, and Peter Brusilovsky. 2023. Chatgpt to replace crowdsourcing of paraphrases for

intent classification: Higher diversity and comparable model robustness.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16. ACM.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations.

Pengyu Cheng, Weituo Hao, Siyang Yuan, Shijing Si, and Lawrence Carin. 2021. Fairfil: Contrastive neural debiasing method for pretrained text encoders.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *CoRR*, abs/1911.02116.

Haixing Dai, Zhengliang Liu, Wenxiong Liao, Xiaoke Huang, Yihan Cao, Zihao Wu, Lin Zhao, Shaochen Xu, Wei Liu, Ninghao Liu, Sheng Li, Dajiang Zhu, Hongmin Cai, Lichao Sun, Quanzheng Li, Dinggang Shen, Tianming Liu, and Xiang Li. 2023. Auggpt: Leveraging chatgpt for text data augmentation.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Sherin Eliyas and P Ranjana. 2022. Recommendation systems: Content-based filtering vs collaborative filtering. In *2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*, pages 1360–1365. IEEE.

William Falcon and The PyTorch Lightning team. 2019. PyTorch Lightning.

Yacine Gaci, Boualem Benatallah, Fabio Casati, and Khalid Benabdeslem. 2022. Debiasing pretrained text encoders by paying attention to paying attention. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9582–9602, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Daniel Gillick, Sayali Kulkarni, Larry Lansing, Alessandro Presta, Jason Baldridge, Eugene Ie, and Diego Garcia-Olano. 2019. Learning dense representations for entity retrieval. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 528–537, Hong Kong, China. Association for Computational Linguistics.

Shiqiang Guo, Folami Alamudun, and Tracy Hammond. 2016. Résumatcher: A personalized résumé-job matching system. *Expert Systems with Applications*, 60:169–182.

Yue Guo, Yi Yang, and Ahmed Abbasi. 2022. Autodebias: Debiasing masked language models with automated biased prompts. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1012–1023, Dublin, Ireland. Association for Computational Linguistics.

Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. 2018. Co-teaching: Robust training of deep neural networks with extremely noisy labels.

Mansoor Iqbal. 2023. LinkedIn usage and revenue statistics (2023). a. Accessed: 2023-12-29.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning.

Sophie Jentzsch and Cigdem Turan. 2022. Gender bias in BERT - measuring and analysing biases through sentiment rating in a realistic downstream classification task. In *Proceedings of the 4th Workshop on Gender Bias in Natural Language Processing (GeBNLP)*, pages 184–199, Seattle, Washington. Association for Computational Linguistics.

Junshu Jiang, Songyun Ye, Wei Wang, Jingran Xu, and Xiaosheng Luo. 2020. Learning effective representations for person-job fit by feature fusion.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension.

Ehsan Kamalloo, Nandan Thakur, Carlos Lassance, Xueguang Ma, Jheng-Hong Yang, and Jimmy Lin. 2023. Resources for brewing beir: Reproducible reference models and an official leaderboard.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. 2020. Dense passage retrieval for open-domain question answering.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova,

Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization.

Yuanhua Lv and ChengXiang Zhai. 2011. When documents are very long, bm25 fails! In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, page 1103–1104, New York, NY, USA. Association for Computing Machinery.

Saket Maheshwary and Hemant Misra. 2018. Matching resumes to jobs via deep siamese network. *Companion Proceedings of the The Web Conference 2018*.

Eran Malach and Shai Shalev-Shwartz. 2018. Decoupling "when to update" from "how to update".

Chandler May, Alex Wang, Shikha Bordia, Samuel R. Bowman, and Rachel Rudinger. 2019. On measuring social biases in sentence encoders. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 622–628, Minneapolis, Minnesota. Association for Computational Linguistics.

Sonali Mhatre, Bhawana Dakhare, Vaibhav Ankolekar, Neha Chogale, Rutuja Navghane, and Pooja Gotarne. 2023. Resume screening and ranking using convolutional neural network. In *2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS)*, pages 412–419.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2022. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. *CoRR*, abs/1611.09268.

OpenAI. 2022a. New and improved embedding model.

OpenAI. 2022b. OpenAI: Introducing ChatGPT.

Keiron O'Shea and Ryan Nash. 2015. An introduction to convolutional neural networks.

Chuan Qin, Hengshu Zhu, Tong Xu, Chen Zhu, Liang Jiang, Enhong Chen, and Hui Xiong. 2018. Enhancing person-job fit for talent recruitment: An ability-aware neural network approach. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, page 25–34, New York, NY, USA. Association for Computing Machinery.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: Memory optimizations toward training trillion parameter models.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. Bpr: Bayesian personalized ranking from implicit feedback.

Sima Rezaeipourfarsangi and Evangelos E. Milios. 2023. Ai-powered resume-job matching: A document ranking approach using deep neural networks. In *Proceedings of the ACM Symposium on Document Engineering 2023*, DocEng '23, New York, NY, USA. Association for Computing Machinery.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Timo Schick, Sahana Udupa, and Hinrich Schütze. 2021. Self-diagnosis and self-debiasing: A proposal for reducing corpus-based bias in NLP. *Transactions of the Association for Computational Linguistics*, 9:1408–1424.

Taihua Shao, Chengyu Song, Jianming Zheng, Fei Cai, and Honghui Chen. 2023. Exploring internal and external interactions for semi-structured multivariate attributes in job-resume matching. In *International Journal of Intelligent Systems*.

Ralf C. Staudemeyer and Eric Rothstein Morris. 2019. Understanding lstm – a tutorial into long short-term memory recurrent neural networks.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models.

Andrew Trotman, Antti Puurula, and Blake Burgess. 2014. Improvements to bm25 and language models examined. In *Proceedings of the 19th Australasian Document Computing Symposium*, ADCS '14, page 58–65, New York, NY, USA. Association for Computing Machinery.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2023. SimLM: Pre-training with representation bottleneck for dense passage retrieval. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2244–2258, Toronto, Canada. Association for Computational Linguistics.

Jason Wei and Kai Zou. 2019. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6383–6389, Hong Kong, China. Association for Computational Linguistics.

Rui Yan, Ran Le, Yang Song, Tao Zhang, Xiangliang Zhang, and Dongyan Zhao. 2019. Interview choice reveals your preference on the market: To improve job-resume matching through profiling memories. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 914–922, New York, NY, USA. Association for Computing Machinery.

Chen Yang, Yupeng Hou, Yang Song, Tao Zhang, Ji-Rong Wen, and Wayne Xin Zhao. 2022. Modeling two-way selection preference for person-job fit. In *RecSys*.

Yunchong Zhang, Baisong Liu, and Jiangbo Qian. 2023. Fedpjf: federated contrastive learning for privacy-preserving person-job fit. *Applied Intelligence*, 53:27060 – 27071.

Chen Zhu, Hengshu Zhu, Hui Xiong, Chao Ma, Fang Xie, Pengliang Ding, and Pan Li. 2018. Person-job fit: Adapting the right talent for the right job with joint representation learning.

A More Details on Dataset and Preprocessing

Intellipro Dataset The talent-job pairs come from the headhunting business in Intellipro Group Inc. The original resumes/job posts are parsed into text fields using techniques such as OCR. Some of the information is further corrected by humans. All sensitive information, such as names, contacts, college names, and company names, has been either removed or converted into numeric IDs. Example resume and job post are shown in Table A2 and Table A3, respectively.

AliYun Dataset The 2019 Alibaba job-resume intelligent matching competition provided resume-job data that is already desensitized and parsed into a collection of text fields. There are 12 fields in a resume (Table A2) and 11 fields in a job post (Table A3) used during training/validation/testing. Sensitive fields such as "居住城市" (living city) were already converted into numeric IDs. "工作经验" (work experience) was processed into a list of keywords. Overall, the average length of a resume or a job post in the AliYun dataset is much shorter than that of the Intellipro dataset (see Table 1). In our analysis, we also manually remapped the industries mentioned in the AliYun dataset into 20 categories such as "Agriculture", "Manufacturing", "Financial Services", etc., to be more comparable with the Intellipro dataset.

B More Details on Model Architecture

In this work, all data (resumes and job posts) are formatted as a collection of text fields. We simplify the model architecture from InEXIT (Shao et al., 2023) to produce a single dense vector for a job post or a resume. InEXIT first encodes each field independently using a pre-trained encoder, by encoding the field name (e.g., "education") and the value (e.g., "Bachelor; major: Computer Science...) separately and then concatenating the two representations to obtain a representation for the entire field. Then, InEXIT models the "internal interaction" between fields from the same document using a self-attention layer. Next, InEXIT views resume-job matching as a non-linear interaction between the fields from a resume-job pair, and uses another self-attention to model the "external interaction" between all representations from both documents. Finally, InEXIT merges the representations obtained so far into a dense vector for a resume/job, concatenates the dense vectors to represent an resume-job pair, and finally uses an MLP layer to produce a matching score.

Validation	**Intellipro Dataset**	**Aliyun Dataset**
# Jobs	109	299
# Resumes	120	278
# Labels	120	300

Table A1: Validation dataset statistics

Compared to concatenating all text fields into a single string and using an encoder to directly produce an embedding, this approach of encoding each text field independently can effectively increase maximum context length (often 512). For example, we find fields such as "Experiences" and "Projects" in a resume from the Intellipro dataset often contain long texts. By encoding each field independently, we can include up to 512 tokens from *each field*, compared to 512 tokens in total if the two fields are concatenated. We believe this is particularly suitable for modeling resume and job posts, as text fields (i.e., sections) from a resume/job post can be understood independently of other fields.

Since CONFIT models resume-job match using inner product (compatible with efficient retrieval frameworks such as FAISS (Johnson et al., 2019)), we propose a few simplications to InEXIT's model architecture. First, since field names (e.g. "education", "experiences") are short, we directly concatenate them with the value to obtain a single string for each field (e.g., "education: Bachelor in Computer Science, ..."). We then use a pre-trained encoder to directly obtain a representation for the entire field. Next, we follow InEXIT to use self-attention in a transformer layer to model the "internal interaction" between fields from the same document. After that, as we aim to model a resume and a job as dense vectors independent of each other, we remove the self-attention layer and the final MLP layer used to model a non-linear interaction between a resume-job pair. Instead, we use a linear layer to merge the representations for each text field, and output a dense vector for a resume or a job. This can then be used to perform inner product scoring, and can be combined with FAISS (see Section 4.6) to rank thousands of documents under miliseconds.

C More Details on Baselines

XGBoost We use "XGBoost-classifier" (Chen and Guestrin, 2016) for classification based metrics,

R from Intellipro Dataset	R from AliYun Dataset
User ID: xxxxx Languages: ENGLISH; Education: start_date: xxxx-xx-xx; end_date: xxxx-xx-xx; college_ranking: 20; major_name: Computer Science; degree: BACHELOR; Location: city_id: 115; province_id: 827; country_id: 14; Preferred Locations: city_id: 115; province_id: 827; country_id: 14; Industry: SOFTWARE_ENGINEERING; Skills: azure; python; ... // other entries omitted Experiences: title: Machine Learning Engineer; start_date: 2017-09; end_date: UNKNOWN; company_ranking: -1; location: UNKNOWN; description: Lead several MLOps projects... title: Software Engineer; ... // other entries omitted Projects: project_name: xxxxx; title: Leader; start_date: xxxx-xx-xx; end_date: xxxx-xx-xx; description: Deploy template-based ...	User ID: xxxxx 学历: 大专; 年龄: 24; 开始工作时间: 2018; 居住城市: 551; 期望工作城市: 551,763,-; 期望工作类型: 工程造价/预结算 期望工作行业: 房地产/建筑/工程 当前工作类型: 土木/建筑/装修/... 当前工作行业: 房地产/建筑/工程 期望薪资: xxxx-xxxx元/月 当前薪资: xxxx-xxxx元/月 工作经验: 停车｜现场｜专家｜公园｜... // other entries omitted

Table A2: Example resume from the Intellipro dataset and AliYun dataset. The Intellipro dataset contains resumes in both English and Chinese, while the AliYun dataset contains resumes only in Chinese. All documents are prepared as a collection of fields, displayed as: "field name: content". Certain details are hidden for privacy concerns. *User_ID* is removed during training/validation/testing. Fields with multiple entries (e.g., *Experiences* in the Intellipro dataset) are concatenated using newlines.

J from Intellipro Dataset	J from AliYun Dataset
Job ID: xxxxx Company Rank: 12 Company Description: Energetic, exciting Silicon Valley startup. Job Title: Deep Learning Specialist Job Location: city_id: 123; province_id: 335; country_id: 56 Job Position Type: Full-time; Job Description/Responsibilities: Use computer vision, computational geometry, and ... // other details omitted Required Qualifications/Skills: Strong programming experience in Python, C++, or Java; PhD in Computer Science, Electrical Engineering, ... // other details omitted Preferred Qualifications/Skills: UNKNOWN	Job ID: xxxxx 工作名称: 工程预算 工作类型: 工程/造价/预结算 工作城市: 719 招聘人数: 3 薪资: 最低xxxx-最高xxxx元每月 招聘开始时间: 2019xxxx 招聘结束时间: 2019xxxx 工作描述: 工程预算员岗位职责：1.能够独立完成...// other details omitted 最低学历: 大专 是否要求出差: 0 工作年限: 五年到十年

Table A3: Example job posts from the Intellipro dataset and AliYun dataset. The Intellipro dataset contains job posts in both English and Chinese, while the AliYun dataset contains job posts only in Chinese. All documents are prepared as a collection of fields, displayed as: "field name: content". Certain details are omitted. *Job_ID* is removed during training/validation/testing.

and "XGBoost-ranker" for ranking based metrics in Table 3 and Table 4. Similar to other classification-targeted methods such as *MV-CoN* and *InEXIT*, we use $\mathcal{D}$ without "contrastive learning". Hyperparam-

eters are tuned using grid search, and classification thresholds are found using the validation set.

RawEmbed We first concatenate all fields in a resume/job post into a single string, and use pre-trained encoders such as BERT (Devlin et al., 2019), E5 (Wang et al., 2022), xlm-roberta (Conneau et al., 2019), and OpenAI text-ada-002 (OpenAI, 2022a) to produce a dense embedding. We use inner product to produce a score for ranking tasks, and use cosine similarity with a threshold found using the validation set for classification tasks.

MV-CoN We follow the official implementations from Bian et al. (2020), but replace the fixed embedding layer with the architecture shown in Section 4.2 and Figure 1, since our test set considers *unseen* resumes and job posts. We use AdamW optimizer (Loshchilov and Hutter, 2019) with a learning rate of 5e-6, a linear warm-up schedule for the first 10% of the training steps, and a weight decay of 1e-2 for both datasets. We use a batch size of 4 with a gradient accumulation of 4 when a small encoder (e.g., BERT-base) is used, and use DeepSpeed Zero 2 (Rajbhandari et al., 2020) with BF16 mixed precision training when a large encoder (e.g., E5-large) is used.

InEXIT We follow the official implementation from Shao et al. (2023) to model both the "internal" and "external" interaction between a resume-job pair. We use AdamW optimizer (Loshchilov and Hutter, 2019) with a learning rate of 5e-6, a linear warm-up schedule for the first 10% of the training steps, and a weight decay of 1e-2 for both datasets. We use a batch size of 8 with a gradient accumulation of 2 when a small encoder is used, and a batch size of 4 with a gradient accumulation of 4 when a large encoder is used.[8]

DPGNN We follow the official implementation from Yang et al. (2022), but remove the fixed-size embedding layer in the graph neural network for encoding a resume or a job, since our test set considers *unseen* resumes and job posts. We replace the embedding layer with a pre-trained encoder (e.g., BERT), and keep other aspects the same, such as modeling both the "active" and "passive" representation of a resume or a job post. We also removed the GraphCNN module as we do not have "interaction records" (e.g., *recruiters reaching out* to job seekers) used to train this module, and the total number of labels in our resume-job datasets is also small. Finally, we modified the proposed BPR loss (Yang et al., 2022) by first normalizing all embedding vectors, since we found training DPGNN with the original BPR loss results in high numerical instability. We use AdamW optimizer (Loshchilov and Hutter, 2019) with a learning rate of 1e-5, a linear warm-up schedule for the first 5% of the training steps, and a weight decay of 1e-2 for both datasets. We use a batch size of 8 with a gradient accumulation of 2 when using a small encoder, and a batch size of 4 with a gradient accumulation of 4 when using a large encoder.

BM25 Since resumes in the Intellipro dataset can be long, we use BM25L (Lv and Zhai, 2011; Trotman et al., 2014) for ranking tasks. We use the implementation from Brown (2020) with the default hyperparameters.

In general, all neural-network-related code is implemented using PyTorch Lightning (Falcon and The PyTorch Lightning team, 2019), and all training is performed on a single A100 80GB GPU. We train all models for 10 epochs and save the best checkpoint based on validation loss for testing. On average, it takes about 1 hour and 4 hours to train *MV-CoN*, *InEXIT*, *DPGNN* using a small encoder on the Intellipro dataset and the AliYun dataset, respectively. When using a large encoder (e.g., E5-large), it takes about 5-8 hours and 19-24 hours to train on the Intellipro dataset and the AliYun dataset, respectively.

D ConFit Training Hyperparameters

In general, ConFit first performs data augmentation using both ChatGPT and EDA (see Section 3.1 and Appendix E for more details), and then trains the model architecture shown in Figure 1 using contrastive learning (see Section 3.2). Similar to baseline methods (see Appendix C), we use the AdamW optimizer (Loshchilov and Hutter, 2019), a linear warm-up schedule for the first 5% of the training steps, and a weight decay of 1e-2 for both datasets. We use a batch size of $B = 8, B_{\text{hard}} = 8$ with a gradient accumulation of 2 when using a small encoder for both datasets. When using a large

[8] In our experiment, we find that InEXIT (Shao et al., 2023) performs slightly worse than MV-CoN (Bian et al., 2020) on the AliYun dataset (see Table 3), while Shao et al. (2023) reports the contrary. We believe this is because InEXIT considers a test setting where part of the resumes/job posts can be *seen* in training, since training/validation/testing pairs are simply randomly sampled. In contrast, in our experiment, we consider test and validation set with only resumes/job posts *not seen* during training.

encoder (e.g., E5-large) on the Intellipro dataset, we keep the same batch size of $B = 8$, but with $B_{\text{hard}} = 4$ and DeepSpeed Zero 2 (Rajbhandari et al., 2020) with BF16 mixed precision training due to GPU memory constraints. On the AliYun dataset, we simply use $B = 8, B_{\text{hard}} = 8$ without DeepSpeed as input sequences are much shorter compared to those from the Intellipro dataset.

We train ConFit models for 10 epochs and save the best checkpoint based on validation loss for testing. On average, ConFit takes about 1.5 hours and 4.5 hours to train when using a small encoder on the Intellipro dataset and the AliYun dataset, respectively. When using a large encoder (e.g., E5-large), ConFit takes about 3 hours and 9 hours to train on the Intellipro dataset and the AliYun dataset, respectively.

E More Details on Data Augmentation

In Section 3.1, we discussed how ConFit can increase the number of resume-job labels by first creating augmented resumes $\hat{R}_i$ and jobs $\hat{J}_i$ that carry semantically similar information as R_i and J_i, and then replicating the labels from R_i and J_i to $\hat{R}_i$ and $\hat{J}_i$. Since much information in a resume or a job post contains formal names such as "Job Title", we *only paraphrase certain sections*. For resumes in the Intellipro dataset, we paraphrase the "description" subsection in the "Experiences" section and the "description" subsection in the "Projects" section (see Table A2). For job posts in the Intellipro dataset, we paraphrase the "Company Description" section, the "Job Description/Responsibilities" section, the "Required Qualifications/Skills", and the "Preferred Qualifications/Skills" section (see Table A3). For the AliYun dataset, we paraphrase the "工作经验" (work experience) section for resumes, and the "工作描述" (job description) section for job posts.

ConFit performs data augmentation using both ChatGPT and EDA for 500 resumes and 500 jobs for each dataset. With only 1000 augmented documents on each dataset, we increased the number of resume-job labels by 5330 and 9706 for the Intellipro dataset and the AliYun dataset, respectively.

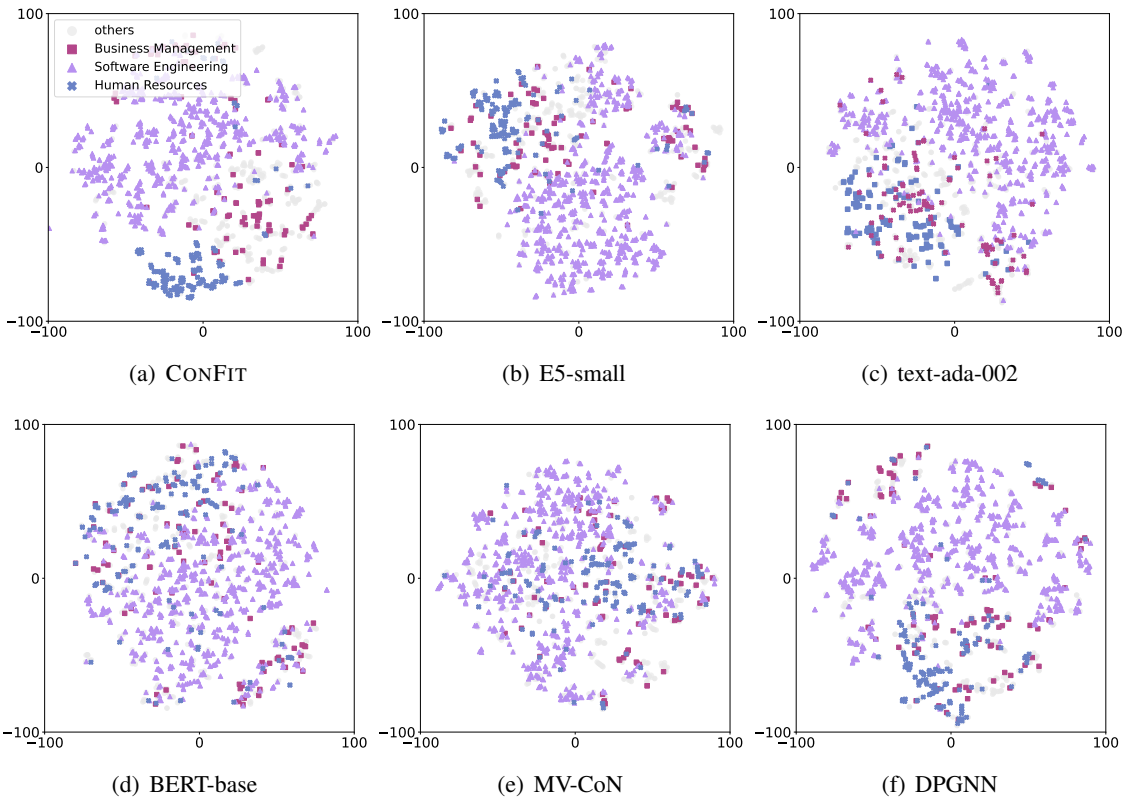
F More Details on Runtime Comparison

In Section 4.6, we compared the runtime of various neural-based methods from Table 3. We categorize neural-based methods into two types when doing inference: Maximum Inner Product Search (*MIPS*) methods and Non-linear (*Non-linear*) methods. MIPS methods compute a matching score between two dense vectors using inner product, and can be efficiently implemented using FAISS (Johnson et al., 2019) to scale to billions of documents. MIPS-based approach includes *RawEmbed* and ConFit. Non-linear methods produce a matching score by modeling non-linear interactions between a resume and a job's (intermediate) representations. For example, *InEXIT* first concatenates the intermediate representations of a resume and a job, and then passes them into a self-attention layer and an MLP layer for scoring. Non-linear methods include *MV-CoN*, *InEXIT*, and *DPGNN*.

All experiments are performed using the test set from the AliYun dataset on a single A100 80GB GPU. For MIPS-based methods, we precompute all the relevant embeddings (excluded from runtime calculation), and record the average runtime for FAISS to retrieve the top 10 job posts from a pool of 100, 1000, and 10000 job posts when given a resume embedding. For non-linear methods, we record the average runtime to perform all the needed forward passes for each of the 100, 1000, and 10000 resume-job *pairs*. However, we do note that the runtime for non-linear methods *can be further optimized* by precomputing certain intermediate representations before passing them into their respective non-linear scoring layers. We did not perform this optimization because 1) this is highly architecture- and method-dependent, and 2) it still does not scale well when the number of job posts is large, or when there are multiple resumes to query.

G More Details on Ablation Studies

Our ablation studies in Section 4.7 also experimented with removing neural networks completely, to decouple our methodology from any particular choice of neural networks. To achieve this, we first mimic the batches used during contrastive training in ConFit and construct a dataset $\mathcal{D}_{\text{con}}$ which contains a positive resume-job pair $\langle R_i^+, J_i^+ \rangle$ along with l negative resumes and l negative job posts (see Section 3.2). Then, we treat all negative resumes and job posts that have a label of $y = 0$ when paired with J_i^+ and R_i^+, respectively. Finally, we encode all resumes and job posts using TF-IDF, and train an XGBoost ranker using $\mathcal{D}_{\text{con}}$. To be comparable with ConFit which uses $B = 8, B_{\text{hard}} = 8$, we use $l = 16$ for each positive resume-job pair,



(a) ConFit (b) E5-small (c) text-ada-002

(d) BERT-base (e) MV-CoN (f) DPGNN

Figure A1: Resume embeddings produced by various methods in Table 3 with BERT-base-multilingual-cased as backbone encoder. Colors assigned using each resume's desired industry. Top-3 most frequent industries are color-coded for easier viewing. *BERT-base* refers to raw embedding produced by BERT-base-multilingual-cased.

		Intellipro Dataset							**AliYun Dataset**						
		Rank Resume		Rank Job		Classification			Rank Resume		Rank Job		Classification		
Method	**Encoder**	MAP	nDCG	MAP	nDCG	F1	Prc+	Rcl+	MAP	nDCG	MAP	nDCG	F1	Prc+	Rcl+
MV-CoN	BERT-base	10.81	10.00	3.34	2.17	*58.00*	*50.00*	33.33	5.41	5.15	13.44	12.67	**74.25**	**72.22**	*68.32*
InEXIT	BERT-base	12.27	12.98	4.11	3.46	55.55	44.74	35.42	5.25	4.98	13.02	12.30	*71.75*	*66.67*	**72.18**
DPGNN	BERT-base	*19.64*	*21.95*	**17.86**	**19.60**	**61.16**	**52.38**	*45.83*	*19.96*	*24.64*	*27.23*	*30.07*	50.31	45.24	57.14
Ours+XGBoost	TF-IDF	**24.04**	**27.29**	*15.60*	*17.23*	43.60	37.66	**60.42**	**24.19**	**28.95**	**30.29**	**33.66**	52.31	47.51	64.67

Table A4: ConFit without neural networks (denoted as *Ours+XGBoost*) is competitive against many prior person-job fit methods with BERT-base as a backbone encoder. *F1* is weighted F1 score, *nDCG* is nDCG@10, *Prc+* and *Rcl+* are precision and recall for positive classes. Results for non-deterministic methods are averaged over 3 runs. Best result is shown in **bold**, and runner-up is in *gray*.

with 14 random negatives and 2 hard negatives.

We denote this approach as *Ours+XGboost*, and compare its performance against other person-job fit systems in Table A4. We find our approach is still competitive against these methods that use a BERT-base (Devlin et al., 2019) encoder. This suggests that the contrastive learning and data augmentation procedure from ConFit is effective for the person-job fit task.

H More Details on Qualitative Analysis

Figure A1 presents the resume embeddings produced by various methods in Table 3 with BERT-base-multilingual-cased as the backbone encoder (with the exception of OpenAI text-ada-002, which is from Table 4). Since methods such as *MV-CoN*, *InEXIT*, and *DPGNN* does not explicitly learn a resume or a job embedding, we extract the representations from the last layer before their resume-job pair scoring layers (e.g., the final MLP layer in *MV-CoN*, or the self-attention layers in *InEXIT*).

In general, we find embeddings produced by *MV-CoN*, *DPGNN*, and *BERT-base* tend to scatter "Software Engineering"-related resumes across the entire embedding space, while embeddings produced by ConFit, *E5-small*, and *text-ada-002*

has a clearer separation between "Software Engineering" and other industries such as "Human Resource". In Table 3, we similarly find the ranking performances of ConFit, *E5-small*, and *text-ada-002* are better than *MV-CoN*, *DPGNN*, and *BERT-base* on the Intellipro dataset. Therefore, we believe Figure A1 qualitatively shows that having a high-quality embedding space is beneficial for modeling person-job fit.