Abstract

The labor market is changing rapidly, prompting increased interest in the automatic extraction of occupational skills from text. With the advent of English benchmark job description datasets, there is a need for systems that handle their diversity well. We tackle the complexity in occupational skill datasets tasks—combining and leveraging multiple datasets for skill extraction, to identify rarely observed skills within a dataset, and overcoming the scarcity of skills across datasets. In particular, we investigate the retrieval-augmentation of language models, employing an external datastore for retrieving similar skills in a dataset-unifying manner. Our proposed method, Nearest Neighbor Occupational Skill Extraction (NNOSE) effectively leverages multiple datasets by retrieving neighboring skills from other datasets in the datastore. This improves skill extraction without additional fine-tuning. Crucially, we observe a performance gain in predicting infrequent patterns, with substantial gains of up to 30% span-F1 in cross-dataset settings.

1 Introduction

Labor market dynamics, influenced by technological changes, migration, and digitization, have led to the availability of job descriptions (JD) on platforms to attract qualified candidates (Brynjolfsson and McAfee, 2011, 2014; Balog et al., 2012). JDs consist of a collection of skills that exhibit a characteristic long-tail pattern, where popular skills are more common while niche expertise appears less frequently across industries (Autor et al., 2003; Autor and Dorn, 2013), such as “teamwork” vs. “system design”\footnote{Examples are from the CEDEFOP Skill Platform.}. This pattern poses challenges for skill extraction (SE) and analysis, as certain skills may be underrepresented, overlooked, or emerging in JDs. This complexity makes the extraction and analysis of skills more difficult, resulting in a sparsity of skills in SE datasets. We tackle this by combining three different skill datasets.

To address the challenges in SE, we explore the potential of Nearest Neighbors Language Models (NNLMs; Khandelwal et al., 2020). NNLMs calculate the probability of the next token by combining a parametric language model (LM) with a distribution derived from the k-nearest context–token pairs in the datastore. This enables the storage of large amounts of training instances without the need to retrain the LM weights, improving language modeling. However, the extent to which NNLMs enhance application-specific end-task performance beyond language modeling remains relatively unexplored. Notably, NNLMs offer several advantages, as highlighted by Khandelwal et al. (2020): First, explicit memorization of the training data aids generalization. Second, a single LM can adapt to multiple domains without domain-specific training, by incorporating domain-specific data into the datastore (e.g., multiple datasets). Third, the NNLM architecture excels at predicting rare patterns, particularly the long-tail.

Therefore, we seek to answer the question: How effective are nearest neighbors retrieval methods for occupational skill extraction? Our contributions are as follows:

- To the best of our knowledge, we are the first to investigate encoder-based $k$NN retrieval by leveraging multiple datasets.
- Furthermore, we present a novel domain-specific RoBERTa\textsubscript{base}-based language model, JobBERTa, tailored to the job market domain.
- We conduct an extensive analysis to show the advantages of $k$NN retrieval, in contrast to prior work that primarily focuses on hyperparameter-specific analysis.\footnote{Code and data: https://github.com/mainlp/nnose.}
2 Nearest Neighbor Skill Extraction

Skill Extraction. The task of SE is formulated as a sequence labeling problem. We define a set of job description sentences \( \mathcal{X} \), where each \( d \in \mathcal{X} \) represents a set of sequences with the \( j \)th input sequence \( \mathcal{X}^j_d = \{x_1, x_2, ..., x_i\} \), with a corresponding target sequence of B1O-labels \( \mathcal{Y}^j_d = \{y_1, y_2, ..., y_i\} \). The labels include “B” (beginning of a skill token), “I” (inside skill token), and “O” (any outside token). The objective is to use \( \mathcal{D} \) in training a labeling algorithm that accurately predicts entity spans by assigning an output label \( y_i \) to each token \( x_i \).

2.1 NNOSE

The core idea of NNOSE is that we augment the extraction of skills during inference with a \( k \)NN retrieval component and a datastore consisting of context–token pairs. Figure 1 outlines our two-step approach. First, we extract skills using token representation \( h_i \) from \( x_i \) and assign a probability distribution \( p_{SE} \) for each \( h_i \) in the input sentence. Second, we use each \( h_i \) to find the most similar token representations in the datastore and get the probability distribution \( p_{kNN} \), aggregated from the \( k \)-nearest context–token pairs. Last, we obtain the final probability distribution \( p \) by interpolating between the two distributions. In addition to formalizing NNOSE, we apply the Whitening Transformation (Section 2.2) to the embeddings, an important process for \( kNN \) approaches as used in previous work (Su et al., 2021; Yin and Shang, 2022).

Datstore. The datastore \( \mathcal{D} \) comprises key–value pairs \((h_i, y_i)\), where each \( h_i \) represents the contextualized token embedding computed by a fine-tuned SE encoder, and \( y_i \in \{B, I, O\} \) denotes the corresponding gold label. Typically, the datastore consists of all tokens from the training set. In contrast to the approach employed by Wang et al. (2022b) for \( kNN–NER \), where they only store B and I tags in the datastore (only named entities), we also include the O-tag in the datastore. This allows us to retrieve non-named entities, which is more intuitive than assigning non-entity probability mass to the B and I tokens.

Inference. During inference, the NNOSE model aims to predict \( y_i \) based on the contextual representation of \( x_i \) (i.e., \( h_i \)). This representation is used to query the datastore for \( kNN \) using an \( L^2 \) distance measure (following Khandelwal et al., 2020), denoted as \( d(\cdot, \cdot) \). Once the neighbors are retrieved, the model computes a distribution over the neighbors by applying a softmax function with a temperature parameter \( T \) to their negative distances (i.e., similarities). This aggregation of probability mass for each label (B, I, O) across all occurrences in the retrieved targets is represented as:

\[
p_{kNN}(y_i | x_i) \propto \sum_{(h_i, y_i) \in \mathcal{D}} 1_{y=y_i} \exp \left( -\frac{d(h_i, k)}{T} \right).
\]

Items that do not appear in the retrieved targets have zero probability. Finally, we interpolate the nearest neighbors distribution \( p_{kNN} \) with the fine-tuned model distribution \( p_{SE} \) using a tuned parameter \( \lambda \) to produce the final NNOSE distribution \( p \):

\[
p(y_i | x_i) = \lambda \times p_{kNN}(y_i | x_i) + (1 - \lambda) \times p_{SE}(y_i | x_i).
\]

2.2 Whitening Transformation

Several works (Li et al., 2020a; Su et al., 2021; Huang et al., 2021) note that if a set of vectors are isotropic, we can assume it is derived from the Standard Orthogonal Basis, which also indicates
Table 1: Dataset Statistics. We provide statistics for all three datasets, including the location and license. Input granularity is at the token level, with performance measured in span-F1. The size of the datastore $D$ is in tokens and determined by embedding tokens and their context from the training sets, resulting in approximately 350K keys. See Appendix B for examples.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Location</th>
<th>License</th>
<th>Train</th>
<th>Dev.</th>
<th>Test</th>
<th>$D$ (Tokens)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SKILLSPAN</td>
<td>*</td>
<td>CC-BY-4.0</td>
<td>5,866</td>
<td>3,992</td>
<td>4,680</td>
<td>86.5K</td>
</tr>
<tr>
<td>SAYFULLINA</td>
<td>UK</td>
<td>Unknown</td>
<td>3,706</td>
<td>1,854</td>
<td>1,853</td>
<td>53.1K</td>
</tr>
<tr>
<td>GREEN</td>
<td>UK</td>
<td>CC-BY-4.0</td>
<td>8,670</td>
<td>963</td>
<td>336</td>
<td>209.5K</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td></td>
<td>349.2K</td>
<td>6,677</td>
<td>963</td>
<td></td>
</tr>
</tbody>
</table>

that we can properly calculate the similarity between embeddings. Otherwise, if it is anisotropic, we need to transform the original sentence embedding to enforce isotropic form, and then measure similarity. **Su et al. (2021); Huang et al. (2021)** applies the vector whitening approach (Koivunen and Kostinski, 1999) on BERT (Devlin et al., 2019). The Whitening Transformation (WT), initially employed in data preprocessing, aims to eliminate correlations among the input data features for a model. In turn, this can improve the performance of certain models that rely on uncorrelated features. Other works (Gao et al., 2019; Ethayarajh, 2019; Li et al., 2020b; Yan et al., 2021; Jiang et al., 2022b, among others) found that (frequency) biased token embeddings hurt final sentence representations. These works often link token embedding bias to the token embedding anisotropy and argue it is the main reason for the bias. We apply WT to the token embeddings like previous work for nearest neighbor retrieval (Yin and Shang, 2022). In short, WT transforms the mean value of the embeddings into 0 and the covariance matrix into the identity matrix, and these transformations are then applied to the original embeddings. We apply WT to the embeddings before putting them in the datastore and before querying the datastore. The workflow of WT is detailed in Appendix A.

3 Experimental Setup

3.1 Data

All datasets are in English and have different label spaces. We transform all skills to the same label space and give each token a generic tag (i.e., B, I, O). We give a brief description of each dataset below and Table 1 summarizes them:

**SKILLSPAN (Zhang et al., 2022a).** This job posting dataset includes annotations for skills and knowledge derived from the ESCO taxonomy. To fit our approach, we flatten the two label layers into one layer (i.e., B10). The baseline is the JobBERT model, which was continuously pre-trained on a dataset of 3.2 million job posting sentences. The industries represented in the data range from tech to more labor-intensive sectors.

**SAYFULLINA (Sayfullina et al., 2018)** is used for soft skill sequence labeling. Soft skills are personal qualities that contribute to success, such as teamwork, dynamism, and independence. Data originated from the UK. This is the smallest dataset among the three, with no specified industries.

**GREEN (Green et al., 2022).** A dataset for extracting skills, qualifications, job domain, experience, and occupation labels. The dataset consists of jobs from the UK, and the industries represented include IT, finance, healthcare, and sales. This is the largest dataset among the three.

3.2 Models

We use 3 English-based LMs: 1 general-purpose and 2 domain-specific models. Implementation details for fine-tuning and NNOSE are in Appendix C, including inference costs of our proposed method.

**JobBERT (Zhang et al., 2022a)** is a 110M parameter BERT-based model continuously pre-trained (Gururangan et al., 2020) on 3.2M English job posting sentences. It outperforms BERTbase on several skill-specific tasks.

**RoBERTa (Liu et al., 2019).** We also use RoBERTabase (123M parameters). It showed to outperform JobBERT in our initial experiments and we therefore include this model as a baseline.

**JobBERTa (Ours).** Given that RoBERTa outperformed JobBERT, we create another baseline and release a model named JobBERTa. This is a
We evaluate the performance of fine-tuning models enhanced with NNOSE. We consider different setups: First, we compare using the Whitening Transformation (+WT) or without. Second, we explore two datastore setups: One using an in-dataset datastore (\{D\}), where each respective training set is stored separately, and another where all datasets are stored in the datastore (\forall D). In the latter setup, we encode all three datasets with each fine-tuned model, and each model has its own WT matrix. For example, we fine-tune a model on \textsc{skillspan} and encode the training set tokens of \textsc{skillspan}, \textsc{sayfullina}, and \textsc{green} to populate the datastore. From the results on the development set (Table 11, Appendix D), we observe that adding WT consistently improves performance. Therefore, we only report the span-F1 scores on each test set (Table 2) with WT and the average over all three datasets.

### Best Model Performance

In Table 2, we show that the best-performing baseline model is JobBERTa, achieving more than 4 points span-F1 improvement over JobBERT and 2 points higher than RoBERTa on average. This confirms the effectiveness of DAPT in improving language models (Han and Eisenstein, 2019; Alsentzer et al., 2019; Gururangan et al., 2020; Lee et al., 2020; Nguyen et al., 2020; Zhang et al., 2022a).

### Best NNOSE Setting

We confirm the trends from dev. on test: The largest improvements come from using the setup with WT, especially in the \forall D+WT setting. All models seem to benefit from the NNOSE setup, JobBERT and JobBERTa show the largest improvements, with the largest gains observed in the \forall D+WT datastore setup. In summary, \forall D+WT consistently demonstrates performance enhancements across all experimental setups.

### Analysis

As we store training tokens from all datasets in the datastore, we expect the model to recall a greater number of skills based on the current context during inference. In turn, this would lead to improved downstream model performance. We want to address the challenges of SE datasets by predicting long-tail patterns, and if we observe improvements in detecting unseen skills in a cross-dataset setting.

To investigate in which situations our model improves, we are analyzing the following: (1) The predictive capability of NNOSE in relation to rarely occurring skills compared to regular fine-tuning (Section 5.1). Skills exhibit varying frequencies across datasets, we categorize the skill frequencies into buckets and compare the performance between vanilla fine-tuning and the inclusion of kNN. (2) If NNOSE actually retrieves from other datasets when they are combined (Section 5.2), and if there is a sign of leveraging multiple datasets, then; (3) How much does NNOSE enhance performance in a cross-dataset setting (Section 5.3)? Our results in-
5.1 Long-tail Skills Prediction

Khandelwal et al. (2020) observed that due to explicitly memorizing the training data, NNLMs effectively predict rare patterns. We analyze whether the performance of “long-tail skills” improves using NNOSE. A visualization of the long-tail distribution of skills is in Figure 8 (Appendix E).

We present the results in Figure 2. We investigate the performance of JobBERTa with and without kNN based on the occurrences of skills in the evaluation set relative to the train set. We count the skills in the evaluation set that occur a number of times in the training set, ranging from 0–15 occurrences and is grouped into low, mid–low, mid–high, and high–frequency bins (0–3, 4–6, 7–10, 10–15, respectively). This approach estimates the number of skills the LM recalls from the training stage.

Our findings reveal that low-frequent skills are the most difficult and make up the largest bucket, and our approach is able to improve on them on all three datasets. For SKILLSPAN, we observe an improvement in the low-frequency bin, from 53.9→54.5 span-F1. Similarly, GREEN exhibits a similar trend with an improvement in the low-frequency bin (49.2→50.1). Interestingly, it also shows gains in most other frequency bins. Last, for SAYFULLINA, there is also an improvement (69.7→70.7 in the low bin). It is worth pointing out that there are many skills that fall in the low bin in SKILLSPAN and GREEN. This is exactly where NNOSE improves most for these datasets. For SAYFULLINA, we notice the largest number of predicted skills is in the mid–low bin. This is where we also see improvements for NNOSE.

5.2 Retrieving From All Datasets

We presented the best improvements of NNOSE in the ∀D+WT datastore in Section 4. An important question remains: Does the ∀D+WT setting retrieve from all datasets? Qualitatively, Figure 3 shows the UMAP visualization (McInnes et al., 2018) of representations stored in each ∀D+WT datastore. We mark the retrieved neighbors with orange for each downstream dev. set. In all plots, we observe that GREEN is prominent in the representation space (green), while SKILLSPAN (darkcyan) and SAYFULLINA (blue) form distinct clusters. Each plot has its own pattern: SKILLSPAN and SAYFULLINA have well-shaped clusters, while GREEN consists of one large cluster. SKILLSPAN and SAYFULLINA mostly retrieve from their own clusters. In contrast, GREEN retrieves from the entire space, which can explain the largest span-F1 performance gains (Table 2). This suggests that kNN effectively leverages multiple datasets in most cases.

5.3 Prediction of Unseen Skills

The UMAP plots in Figure 3 suggest that some datasets are closer to each other than others. To quantify this, we investigate the overlap of annotated skills between datasets and assess cross-dataset performance of NNOSE on unseen skills.
Figure 3: UMAP Visualization of Nearest Neighbors Retrieval. The datastore consists of the training set (+WT) of all three datasets used in this work. Each colored dot represents a non-0 token from the training set. The embeddings are generated using JobBERTa. The orange shade represents the retrieved neighbors with $k = 4$ for each token that is a skill (i.e., not an O token). Note that for the middle plot, the orange shade covers the blue clusters SAYFULLINA. GREEN has the green shade and SKILLSPAN are the darkcyan colors.

Table 3: Results of Unseen Skills based on JobBERTa ($\forall +WT$). In the vanilla setting, models trained on one skill dataset are applied to another on test, showing varied performance. However, applying $k$NN improves the detection of unseen skills. Diagonal results can be found in Table 2. Refer to Table 10 for tuned hyperparameters.

<table>
<thead>
<tr>
<th>Trained on</th>
<th>SKILLSPAN</th>
<th>SAYFULLINA</th>
<th>GREEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vanilla</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SKILLSPAN</td>
<td>9.44</td>
<td>18.05</td>
<td>43.17</td>
</tr>
<tr>
<td>SAYFULLINA</td>
<td>29.67</td>
<td>15.93</td>
<td>11.79</td>
</tr>
<tr>
<td>ALL</td>
<td>59.33</td>
<td>90.16</td>
<td>44.59</td>
</tr>
<tr>
<td>$k$NN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SKILLSPAN</td>
<td>26.16</td>
<td>45.86</td>
<td>45.44</td>
</tr>
<tr>
<td>SAYFULLINA</td>
<td>42.22</td>
<td>46.58</td>
<td>25.38</td>
</tr>
<tr>
<td>GREEN</td>
<td>41.22</td>
<td>41.22</td>
<td>46.58</td>
</tr>
<tr>
<td>ALL</td>
<td>59.51</td>
<td>90.33</td>
<td>45.63</td>
</tr>
</tbody>
</table>

Overlap of Datasets. We calculate the exact span overlap of skills between the training sets of the datasets using the Jaccard similarity coefficient (Jaccard, 1901): $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$, where $A$ and $B$ are sets of multi-token spans (e.g., “manage a team”) from two separate training sets. The Jaccard similarity coefficients are as follows: $J$(SKILLSPAN, SAYFULLINA) = 0.35, $J$(SAYFULLINA, GREEN) = 0.10, and $J$(SKILLSPAN, GREEN) = 0.29. These Jaccard coefficients indicate overlap between unique skill spans across datasets, suggesting that NNOSE can introduce the model to new and unseen skills.

Results. Table 3 presents the performance of JobBERTa across datasets. For completeness, we include a baseline where JobBERTa is fine-tuned on the union of all datasets (ALL). We notice training on the union of the data never leads to the best target dataset performance. Generally, we observe that in-domain data is best, both in vanilla and NNOSE setups (diagonal in Table 3). Performance drops when a model is applied to a dataset other than the one it was trained on (off-diagonal). Using NNOSE leads to substantial improvements across the challenging off-diagonal (cross-dataset) settings, while performance remains stable within datasets. We observe the largest improvements when applied to SAYFULLINA, with up to a 30% increase in span-F1. This is likely due to SAYFULLINA consisting mostly of soft skills, which are less prevalent in SKILLSPAN and GREEN, making it beneficial to introduce soft skills. Conversely, when the model is trained on SAYFULLINA, the absolute improvement on SKILLSPAN is lower, indicating that skill datasets can benefit each other to different extents.

Cross-dataset Long-tail Analysis. Table 3 shows improvements when NNOSE is used in favor of vanilla fine-tuning. Figure 4 presents the long-tail performance analysis in the cross-dataset scenario, similar to Figure 2. We observe the largest gains with NNOSE in the low or mid–low frequency bins. However, exceptions are SKILLSPAN$\rightarrow$GREEN and SAYFULLINA$\rightarrow$GREEN, where most gains occur in the mid–high bin. Notably, SAYFULLINA$\rightarrow$GREEN demonstrates higher performance with NNOSE, where all 6 skills are incorrectly predicted in the mid–high bin. An analysis of precision and recall in Table 12 (Appendix F) substantiates that the improvements are both precision and recall-based, with
gains of up to 40 recall points and 35.4 precision points in \textit{GREEN}$\rightarrow$\textit{SAYFULLINA}. There is also an improvement up to 35.5 recall points and 34.1 precision points for \textit{SKILLSPAN}$\rightarrow$\textit{SAYFULLINA}. This further solidifies that memorizing tokens (i.e., storing all skills in the datastore) helps recall as mentioned in Khandelwal et al. (2020), and more importantly, highlighting the benefits of NNOSE in cross-dataset scenarios for SE.

5.4 Qualitative Check on Prediction Errors

We perform a qualitative analysis on the false positives (fp) and false negatives (fn) of NNOSE predictions compared to vanilla fine-tuning for each dataset. This analysis tells us whether a prediction corresponds to an actual skill, even if it does not contribute positively to the span-F1 metric. We observe that NNOSE produces a significant number of false positives that are “similar” to genuine skills. In Table 4, for each dataset, we picked five fps and fns that represent hard, soft, and personal skills well (if applicable). We show the fps and fns for JobBERTa with NNOSE, we only show predictions that are \textit{not} in the vanilla model predictions. In \textit{SAYFULLINA}, there is only one fn. We notice from the errors, and especially the fps, that these are definitely skills, indicating the benefit of NNOSE helping to predict new skills or missed annotations. For a general qualitative check on predictions, we refer to Appendix G. We show that NNOSE predicts a variety of close tokens, but also the same tokens if the model is confident about the predictions (i.e., high softmax scores).

6 Related Work

Skill Extraction. The dynamic nature of labor markets has led to an increase in tasks related to JD, including skill extraction (Kivimäki et al., 2013; Zhao et al., 2015; Sayfullina et al., 2018; Smith et al., 2019; Tamburri et al., 2020; Shi et al., 2020; Chernova, 2020; Bhola et al., 2020; Gugnani and Misra, 2020; Fareri et al., 2021; Konstantinidis et al., 2022; Zhang et al., 2022a,b,c; Green et al., 2022; Gnehm et al., 2022; Beauchemin et al., 2022; Decorte et al., 2022; Ao et al., 2023; Goyal et al., 2023; Zhang et al., 2023). These works employ methods such as sequence labeling (Sayfullina et al., 2018; Smith et al., 2019; Chernova, 2020; Zhang et al., 2022a,c), multi-label classification (Bhola et al., 2020), and graph-based meth-
<table>
<thead>
<tr>
<th>True Positives</th>
<th>False Positives</th>
</tr>
</thead>
<tbody>
<tr>
<td>cleaning GCP</td>
<td></td>
</tr>
<tr>
<td>decisive IBM MQ</td>
<td></td>
</tr>
<tr>
<td>Apache Camel</td>
<td></td>
</tr>
<tr>
<td>building consumer demand for sustainable products</td>
<td></td>
</tr>
<tr>
<td>budget responsible</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>True Negatives</th>
<th>False Negatives</th>
</tr>
</thead>
<tbody>
<tr>
<td>empathy leadership management</td>
<td>software engineering</td>
</tr>
<tr>
<td>communication</td>
<td>development</td>
</tr>
<tr>
<td>ability to manage and prioritise multiple assignments and tasks</td>
<td>DevOps</td>
</tr>
<tr>
<td>SQL scripting languages</td>
<td>Cisco network administration</td>
</tr>
<tr>
<td>Manage a team</td>
<td></td>
</tr>
<tr>
<td>troubleshooting activities</td>
<td></td>
</tr>
<tr>
<td>dealing with tenants</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: FPs & FNs of NNOSE. We show several examples of false positives and false negatives in each dataset. We only show the predictions of NNOSE that are not in the vanilla model predictions.

General Retrieval-augmentation. In retrieval augmentation, LMs can utilize external modules to enhance their context-processing ability. Two approaches are commonly used: First, using a separately trained model to retrieve relevant documents from a collection. This approach is employed in open-domain question answering tasks (Petroni et al., 2021) and with specific models such as ORQA (Lee et al., 2019), REALM (Guu et al., 2020), RAG (Lewis et al., 2020), FiD (Izacard and Grave, 2021), and ATLAS (Izacard et al., 2022).

Second, previous work on explicit memorization showed promising results with a cache (Grave et al., 2017), which serves as a type of datastore. The cache contains past hidden states of the model as keys and the next word as tokens in key–value pairs. Memorization of hidden states in a datastore, involves using the $k$NN algorithm as the retriever. The first work of the $k$NN algorithm as the retrieval component was by Khandelwal et al. (2020), leading to several LM decoder-based works.

Decoder-based Nearest Neighbor Approaches. Decoder-based nearest neighbors approaches are primarily focused on language modeling (Khandelwal et al., 2020; He et al., 2021; Yogatama et al., 2021; Ton et al., 2022; Shi et al., 2022; Jin et al., 2022; Bhardwaj et al., 2022; Xu et al., 2023) and machine translation (Khandelwal et al., 2021; Zheng et al., 2021; Jiang et al., 2021, 2022a; Wang et al., 2022a; Martins et al., 2022a,b; Zhu et al., 2022; Du et al., 2023; Zhu et al., 2023; Min et al., 2023b,a). These approaches often prioritize efficiency and storage space reduction, as the datasstores for these tasks can contain billions of tokens.

Encoder-based Nearest Neighbor Approaches. Encoder-based nearest neighbor approaches have been explored in tasks such as named entity recognition (Wang et al., 2022b) and emotion classification (Yin and Shang, 2022). Here, the datasstores are limited to single datasets with the sentence (or token) gold label pairs. Instead, we show the potential of adding multiple datasets in the datastore.

7 Conclusion

We introduce NNOSE, an LM that incorporates and leverages a non-parametric datastore for nearest neighbor retrieval of skill tokens. To the best of our knowledge, we are the first to introduce the nearest neighbors retrieval component for the extraction of occupational skills. We evaluated NNOSE on three relevant skill datasets with a wide range of skills and show that NNOSE enhances the performance of all LMs used in this work without additionally tuning the LM parameters. Through the combination of train sets in the datastore, our analysis reveals that NNOSE effectively leverages all the datasets by retrieving from each. Moreover, NNOSE not only performs well on rare skills but also enhances the performance on more frequent patterns. Lastly, we observe that our baseline models exhibit poor performance when applied in a cross-dataset setting. However, with the introduc-
tion of NNOSE, the models improve across all settings. Overall, our findings indicate that NNOSE is a promising approach for application-specific skill extraction setups and potentially helps discover skills that were missed in manual annotations.

Limitations

We consider several limitations: One is the limited diversity of the datasets used in this work. Our study was constrained by the use of only three English datasets. By focusing solely on English data, the method might not generalize other languages.

Future research includes incorporating a wider range of datasets from diverse sources to obtain a more comprehensive understanding of the topic. Potential interesting future work should include validation on whether NNOSE works in a multilingual setting.

Another limitation is that we do skill detection and not specific labeling of the extracted spans, i.e., extracting generic B, I, 0 tags. This was to ensure that the datasets could be used all together in the datastore.

Last, we only applied the nearest neighbors with the datastore to the job market domain. In contrast, Wang et al. (2022b) have used a similar approach on a more generic domain, e.g., CoNLL data (Tjong Kim Sang and De Meulder, 2003), but also keep it limited to the number of labels in this dataset (i.e., four fine-grained labels: Person, Location, Organization, and Misc.). We believe with coarse-grained span labeling (i.e., B10), our proposed method and positive results have the potential to transfer to other domains.

Ethics Statement

The subject of job-related language models is a highly contentious topic, often sparking intense debates surrounding the issue of bias. We acknowledge that LMs such as JobBERTa and NNOSE possess the potential for inadvertent consequences, such as unconscious bias and dual-use when employed in the candidate selection process for specific job positions. There are research efforts to develop fairer recommender systems in the field of human resources, focusing on mitigating biases (e.g., Mujtaba and Mahapatra, 2019; Raghavan et al., 2020; Deshpande et al., 2020; Köchling and Wehner, 2020; Sánchez-Monedero et al., 2020; Wilson et al., 2021; van Els et al., 2022; Arafan et al., 2022). Nevertheless, one potential approach to alleviating such biases involves the retrieval of sparse skills for recall (e.g., this work). It is important to note, however, that we have not conducted an analysis to ascertain whether this particular method exacerbates any pre-existing forms of bias.

Acknowledgements

We thank the MaiNLP and NLPhnorth group for feedback on an earlier version of this paper, and WING for hosting MZ for a research stay. In particular, thanks to Elisa Bassignana, Robert Litschko, Max Müller-Eberstein, Yanxia Qin, and Tongyao Zhu for helpful suggestions and feedback. This research is supported by the Independent Research Fund Denmark (DFF) grant 9131-00019B and in parts by ERC Consolidator Grant DIALECT 101043235.

References


Erik Brynjolfsson and Andrew McAfee. 2011. Race against the machine: How the digital revolution is accelerating innovation, driving productivity, and irreversibly transforming employment and the economy. Brynjolfsson and McAfee.


Maria Chernova. 2020. Occupational skills extraction with FinBERT. Master’s Thesis.


Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don’t stop pretraining:


Mike Zhang, Rob van der Goot, and Barbara Plank. 2023. ESCOXLM-R: Multilingual taxonomy-driven pre-training for the job market domain. ArXiv preprint, abs/2305.12092.

Meng Zhao, Faizan Javed, Ferosh Jacob, and Matt McNair. 2015. SKILL: A system for skill identification


A Whitening Transformation Algorithm

Algorithm 1: Whitening Transformation Workflow

1. **Input:** Embeddings \( \{x_i\}_{i=1}^{N} \);
2. Compute \( \mu = \frac{1}{N} \sum_{i=1}^{N} x_i \) and \( \Sigma \) of \( \{x_i\}_{i=1}^{N} \);
3. Compute \( U, \Lambda, U^\top = \text{SVD}(\Sigma) \);
4. Compute \( W = U \sqrt{\Lambda^{-1}} \);
5. For \( i = 1, 2, ..., n \) do
6. \( \tilde{x}_i = (x_i - \mu) W \)
7. End
8. Return \( \{\tilde{x}_i\}_{i=1}^{N} \);

We apply the whitening transformation to the query embedding and the embeddings in the datastore. We can write a set of token embeddings as a set of row vectors: \( \{x_i\}_{i=1}^{N} \). Additionally, a linear transformation \( \tilde{x}_i = (x_i - \mu) W \) is applied, where \( \mu = \frac{1}{N} \sum_{i=1}^{N} x_i \). To obtain the matrix \( W \), the following steps are conducted: First, we obtain the original covariance matrix

\[
\Sigma = \frac{1}{N} \sum_{i=1}^{N} (x_i - \mu)^\top (x_i - \mu). \tag{3}
\]

Afterwards, we obtain the transformed covariance matrix \( \tilde{\Sigma} = W^\top \Sigma W \), where we specify \( \tilde{\Sigma} = I \). Therefore, \( \Sigma = (W^\top)^{-1} W^{-1} = (W^{-1})^\top W^{-1} \). Here, \( \Sigma \) is a positive definite symmetric matrix that satisfies the following singular value decomposition (SVD; Golub and Reinsch, 1971) as indicated by Su et al. (2021): \( \Sigma = U \Lambda U^\top \). \( U \) is an orthogonal matrix, \( \Lambda \) is a diagonal matrix, and the diagonal elements are all positive. Therefore, let \( W^{-1} = \sqrt{\Lambda} U^\top \), we obtain the solution: \( W = U \sqrt{\Lambda^{-1}} \). Putting it all together, as input, we have the set of embeddings \( \{x_i\}_{i=1}^{N} \). We compute \( \mu \) and \( \Sigma \) of \( \{x_i\}_{i=1}^{N} \). Then, we perform SVD on \( \Sigma \) to obtain matrices \( U, \Lambda, \) and \( U^\top \). Using these matrices, we calculate the transformation matrix \( W \). Finally, we apply the transformation to each embedding in the set by subtracting \( \mu \) and multiplying by \( W \). We are left with \( \tilde{x}_i = (x_i - \mu) W \).

Note that we do not store the embedding in the datastore, and apply \( W \) to the token embedding before we query the datastore.

We show the Whitening Transformation procedure in **Algorithm 1**. Note that Li et al. (2020a); Su et al. (2021) introduced a dimensionality reduction factor \( k \) on \( W \) (\( W[; : k] \)). The diagonal elements in the matrix \( \Lambda \) obtained from the SVD algorithm are in descending order. One can decide to keep the first \( k \) columns of \( W \) in line 6. This is similar to PCA (Abdi and Williams, 2010). However, empirically, we found that reducing dimensionality had a negative effect on downstream performance, thus we omit that in this implementation.

B Data Examples

Table 5: Data example references for each dataset.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Figure</th>
</tr>
</thead>
<tbody>
<tr>
<td>SKILLSpan</td>
<td>Figure 5</td>
</tr>
<tr>
<td>SAYFULLINA</td>
<td>Figure 6</td>
</tr>
<tr>
<td>GREEN</td>
<td>Figure 7</td>
</tr>
</tbody>
</table>

In Table 5, we refer to several listings of examples of the datasets. Notably in SKILLSpan, the original samples contain two columns of labels. These refer to skills and knowledge. To accommodate for the approach of NNOSE, we merge the labels together and thus removing the possible nesting of skills. Zhang et al. (2022a) mentions that there is not a lot of nesting of skills. Following Zhang et al. (2022a), we prioritize the skills column when merging the labels. When there is nesting, we keep the labels of skills and remove the knowledge labels.

C Implementation Details

**General Implementation.** We obtain all LMs from the Transformers library (Wolf et al., 2020) and implement JobBERTa using the same library. All learning rates for fine-tuning are \( 5 \times 10^{-5} \) using the AdamW optimizer (Loshchilov and Hutter, 2019). We use a batch size of 16 and a maximum sequence length of 128 with dynamic padding. The models are trained for 20 epochs with early stopping using a patience of 5. We implement the retrieval component using the FAISS library (Johnson et al., 2019), which is a standard for nearest neighbors retrieval-augmented methods.

**JobBERTa.** We apply domain-adaptive pre-training (Gururangan et al., 2020), which involves continued self-supervised pre-training of a large LM on domain-specific text. This approach enhances the modeling of text for downstream tasks within the domain. We continue pre-training on a roberta-base checkpoint with 3.2M job posting examples of the datasets. Notably in S

3https://faiss.ai/
Experience 0 in 0 working B on I a I cloud-based I application I running O on 0 Docker B . 0 A 0 degree B in I Computer I Science I or 0 related O fields 0 . 0

ability 0 to 0 work B under I stress I condition 0 due O to 0 the O dynamic B nature 0 of 0 the 0 qualification I and 0 current O Nursing B registration I are 0 essential O for 0 this 0 role 0

Figure 5: Data Example for SkillSpan. In SkillSpan, note the long skills.

Figure 6: Data Example for Sayfullina. In Sayfullina, the skills are usually soft-like skills.

Figure 7: Data Example for Green. There are many qualification skills (e.g., certificates).

sentences from Zhang et al. (2022a). We use a batch size of 8 and run MLM for a single epoch following Gururangan et al. (2020). The rest of the hyperparameters are set to the defaults in the Transformer library.

**NNOSE Setup.** Following previous work, the keys used in NNOSE are the 768-dimensional representation logits obtained from the final layer of the LM (input to the softmax). We perform a single forward pass over the training set of each dataset to save the keys and values, i.e., the hidden representation and the corresponding gold BIO tag. The FAISS index is created using all the keys to learn 4096 cluster centroids. During inference, we retrieve $k$ neighbors. The index looks up 32 cluster centroids while searching for the nearest neighbors. For all experiments, we compute the squared Euclidean ($L^2$) distances with full precision keys. The difference in inference speed is almost negligible, with the $k$NN module taking a few extra seconds compared to regular inference. For the exact hyperparameter values, we indicate them in the next paragraph.

**Hyperparameters NNOSE.** The best-performing hyperparameters and search space can be found in Table 6, Table 7, Table 8, and Table 9. We report the $k$-nearest neighbors, $\lambda$ value, and softmax temperature $T$ for each dataset and model.

In Table 10, we show the hyperparameters for the cross-dataset analysis. In the vanilla setting, we apply the models trained on a particular skill dataset to another skill dataset, similar to transfer learning. We observe a significant discrepancy in performances cross-dataset, indicating a wide range of skills. However, when $k$NN is applied, it improves the detection of unseen skills. The datastore contains tokens from all datasets.

**Inference Cost.** Due to the current size of the datasets (less than 1M tokens in total), it has no noticeable effect on inference time with the fast nearest neighbor search of FAISS (Johnson et al., 2019). We imagine if the datasets come closer to billions of tokens e.g., in machine translation (Khandelwal...
Table 6: Tuned Hyperparameters on Dev. These are for \{D\}.

<table>
<thead>
<tr>
<th>Dataset →</th>
<th>SKILL</th>
<th>SPAN</th>
<th>SAYFULLINA</th>
<th>GREEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobBERT</td>
<td>k 4</td>
<td>4</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>λ 0.3</td>
<td>0.3</td>
<td>0.15</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T 0.1</td>
<td>2.0</td>
<td>10.0</td>
<td></td>
</tr>
<tr>
<td>RoBERTa</td>
<td>k 32</td>
<td>4</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td></td>
<td>λ 0.3</td>
<td>0.3</td>
<td>0.25</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T 10.0</td>
<td>0.1</td>
<td>10.0</td>
<td></td>
</tr>
<tr>
<td>JobBERT</td>
<td>k 16</td>
<td>4</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>λ 0.2</td>
<td>0.1</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T 5.0</td>
<td>10.0</td>
<td>10.0</td>
<td></td>
</tr>
</tbody>
</table>

Search Space

\[ k \in \{4, 8, 16, 32, 64, 128\} \]

\[ \lambda \in \{0.1, 0.15, 0.25, ..., 0.9\} \]

\[ T \in \{0.1, 0.5, 1.0, 2.0, 3.0, 5.0, 10.0\} \]

Table 7: Tuned Hyperparameters on Dev. These are for \{D\} + WT.

<table>
<thead>
<tr>
<th>Dataset →</th>
<th>SKILL</th>
<th>SPAN</th>
<th>SAYFULLINA</th>
<th>GREEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobBERT</td>
<td>k 32</td>
<td>4</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td></td>
<td>λ 0.3</td>
<td>0.3</td>
<td>0.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T 1.0</td>
<td>0.5</td>
<td>2.0</td>
<td></td>
</tr>
<tr>
<td>RoBERTa</td>
<td>k 128</td>
<td>128</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td></td>
<td>λ 0.35</td>
<td>0.1</td>
<td>0.25</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T 0.1</td>
<td>0.5</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>JobBERTa</td>
<td>k 32</td>
<td>8</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td></td>
<td>λ 0.15</td>
<td>0.3</td>
<td>0.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T 0.1</td>
<td>0.1</td>
<td>2.0</td>
<td></td>
</tr>
</tbody>
</table>

Search Space

\[ k \in \{4, 8, 16, 32, 64, 128\} \]

\[ \lambda \in \{0.1, 0.15, 0.25, ..., 0.9\} \]

\[ T \in \{0.1, 0.5, 1.0, 2.0, 3.0, 5.0, 10.0\} \]

Table 8: Tuned Hyperparameters on Dev. These are for \forall D.

<table>
<thead>
<tr>
<th>Trained on</th>
<th>Hyperparams.</th>
<th>SKILL</th>
<th>SPAN</th>
<th>SAYFULLINA</th>
<th>GREEN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>k</td>
<td>16</td>
<td>32</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td></td>
<td>λ</td>
<td>0.9</td>
<td>0.7</td>
<td>0.9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>0.1</td>
<td>0.5</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>SAYFULLINA</td>
<td>k</td>
<td>64</td>
<td></td>
<td>32</td>
<td></td>
</tr>
<tr>
<td></td>
<td>λ</td>
<td>0.9</td>
<td>0.1</td>
<td>0.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>GREEN</td>
<td>k</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td></td>
<td>λ</td>
<td>0.85</td>
<td>0.9</td>
<td>0.9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>0.5</td>
<td>0.1</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>ALL</td>
<td>k</td>
<td>4</td>
<td>128</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td></td>
<td>λ</td>
<td>0.25</td>
<td>0.6</td>
<td>0.65</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>1.0</td>
<td>1.0</td>
<td>0.5</td>
<td></td>
</tr>
</tbody>
</table>

Search Space

\[ k \in \{4, 8, 16, 32, 64, 128\} \]

\[ \lambda \in \{0.1, 0.15, 0.2, 0.25, ..., 0.9\} \]

\[ T \in \{0.1, 0.5, 1.0, 2.0, 3.0, 5.0, 10.0\} \]

Table 10: Results of Unseen Skills (Development Set) based on JobBERTa.

et al., 2021) and language modeling (Khandelwal et al., 2020), the inference time will be larger.

D Development Set Results

We show the dev. set results in Table 11. Overall, the patterns of improvements hold across datasets and models. We base the test set result on the best-performing setups in the development set, i.e., \{D\} + WT and \forall D + WT.

E Frequency Distribution of Skills

We show the skill frequency distribution of the datasets in Figure 8, as mentioned in Section 5.1. Here, we show evidence of the long-tail pattern in skills for each dataset. There is a cut-off at count 15 for GREEN, indicating that there are skills in the development set that occur more than 15 times.

F Further Cross-dataset Analysis

Precision and Recall Scores Cross-dataset. In Table 12, we checked the precision and recall numbers for the cross-dataset setup with \forall D + WT and JobBERTa as the backbone model. When us-
We show several qualitative results of NNOSE. In Table 13, we show a qualitative sample of using JobBERTa on SkillSpan. The current token is “IT” with gold label 0. The language model puts 0.4 softmax probability on the tag I. By retrieving the nearest neighbors, the final probability mass gets shifted towards I with probability 0.43, which is the correct tag.

In Table 14, we show a qualitative sample of using JobBERTa on SkillSpan with multi-token annotations and how this behaves. The current skill is “coding skills” with gold labels B and I respectively. Both the model and kNN puts high confidence in the correct label. Note that the nearest neighbors of “coding” are quite varied, which shows the benefit of NNOSE. Note that all the retrieved “skills” tokens are from different contexts.

In Table 15, we show a qualitative sample of using JobBERTa on SkillSpan. The current to-
Figure 8: **Frequency Distribution of Skill Occurrences in the Train Set.** We display the frequency distribution of skill occurrences in each train set. *How to read:* For instance, in the case of Sayfullina, there are over 2,000 skills that occur only **once** in the training set. We demonstrate that all skill datasets exhibit an inherent long-tail pattern.

ken is “optimistic” with gold label B. This is a so-called “soft skill”. The language model puts high confidence in the tag B, which is the correct tag. The retrieved neighbors are frequently relevant, but sometimes less. This indicates that the retrieved neighbors (all soft skills) occur in similar contexts.

In Table 16, we show a qualitative sample of using JobBERTa on **SKILLS**. The current token is “optimistic” with gold label B. This is a so-called “soft skill”. The language model puts high confidence in the tag B, which is the correct tag. The retrieved neighbors are frequently relevant, but sometimes less. This indicates that the retrieved neighbors (all soft skills) occur in similar contexts.
Table 13: **Cherry Picked Qualitative Sample NNOSE of Higher Precision.** We show a qualitative sample of using JobBERTa on SKILLSPANN. In this case, we see more weight being put on a specific tag, resulting in higher precision.

<table>
<thead>
<tr>
<th>Current token</th>
<th>Gold label</th>
<th>LM prediction probs</th>
<th>Nearest neighbors (k = 8)</th>
<th>Aggregated kNN scores</th>
<th>Final predicted probs</th>
</tr>
</thead>
<tbody>
<tr>
<td>IT</td>
<td>0</td>
<td>[0.277, 0.404, 0.319]</td>
<td>['IT', 'Software', 'Software', 'Cloud', 'Cloud', 'Database', 'Ag', 'software']</td>
<td>[0.000, 0.132, 0.868]</td>
<td>[0.221, 0.350, 0.429]</td>
</tr>
</tbody>
</table>

Table 14: **Cherry Picked Qualitative Sample NNOSE of Multiple Tokens.** We show a qualitative sample of using JobBERTa on SKILLSPANN with multi-token annotations and how this behaves.

<table>
<thead>
<tr>
<th>Current token</th>
<th>Gold label</th>
<th>LM prediction probs</th>
<th>Nearest neighbors (k = 8)</th>
<th>Aggregated kNN scores</th>
<th>Final predicted probs</th>
</tr>
</thead>
<tbody>
<tr>
<td>coding</td>
<td>B</td>
<td>[0.988, 0.000, 0.012]</td>
<td>['programming', 'coding', 'programming', 'debugging', 'scripting', 'writing', 'coding', 'programming']</td>
<td>[1.000, 0.000, 0.000]</td>
<td>[0.991, 0.000, 0.009]</td>
</tr>
<tr>
<td>skills</td>
<td>I</td>
<td>[0.000, 0.990, 0.010]</td>
<td>['skills', 'skills', 'skills', 'skills', 'skills', 'skills', 'skills', 'skills']</td>
<td>[0.000, 1.000, 0.000]</td>
<td>[0.000, 0.992, 0.008]</td>
</tr>
</tbody>
</table>

Table 15: **Cherry Picked Qualitative Sample NNOSE of Randomness.** We show a qualitative sample of using JobBERTa on SKILLSPANN. The language model puts high confidence on the tag I, which is the correct tag. Here the retrieved neighbors do not seem too relevant, which in this case is mostly random chance that it got it correctly.

<table>
<thead>
<tr>
<th>Current token</th>
<th>Gold label</th>
<th>LM prediction probs</th>
<th>Nearest neighbors (k = 8)</th>
<th>Aggregated kNN scores</th>
<th>Final predicted probs</th>
</tr>
</thead>
<tbody>
<tr>
<td>tools</td>
<td>I</td>
<td>[0.250, 0.374, 0.379]</td>
<td>['tools', 'tools', 'transport', 'transport', 'transport', 'transport', 'car', 'transport']</td>
<td>[0.124, 0.626, 0.250]</td>
<td>[0.234, 0.399, 0.366]</td>
</tr>
</tbody>
</table>

Table 16: **Cherry Picked Qualitative Sample NNOSE of Variety.** We show a qualitative sample of using JobBERTa on SKILLSPANN. The language model puts high confidence in the tag B, which is the correct tag. The retrieved neighbors are frequently relevant.

<table>
<thead>
<tr>
<th>Current token</th>
<th>Gold label</th>
<th>LM prediction probs</th>
<th>Nearest neighbors (k = 8)</th>
<th>Aggregated kNN scores</th>
<th>Final predicted probs</th>
</tr>
</thead>
<tbody>
<tr>
<td>optimistic</td>
<td>B</td>
<td>[0.958, 0.000, 0.002]</td>
<td>['proactive', 'responsible', 'holistic', 'operational', 'positive', 'open', 'professional', 'agile']</td>
<td>[1.000, 0.000, 0.000]</td>
<td>[0.995, 0.000, 0.005]</td>
</tr>
</tbody>
</table>