Knowledge Distillation for Job Title Prediction and Project Recommendation in Open Source Communities

Xin Liu, Hang Su, and Xuesong Lu(✉)

East China Normal University, Shanghai, China
{xinliu,suhang}@stu.ecnu.edu.cn, xslu@dase.ecnu.edu.cn

Abstract. In the era of rapid digitalization, the demand for digital talents is surging and talent management in open source communities has become a crucial research area. This paper explores the application of large language models (LLMs) in two key talent management tasks within open source communities: project recommendation and job title prediction. First, we construct an evaluation dataset TM-Eval to assess the performance of LLMs on the two tasks. Second, we construct a QA dataset JA-QA from LinkedIn that describes the required APIs for each job title with job description. The dataset is used to distill knowledge pertaining to job-API correspondence of larger LLMs into smaller ones, in order to reduce computational overhead for the two tasks. We propose a hierarchical knowledge transfer method including logit-based distillation, feature-based distillation and task-specific fine-tuning with Low-Rank Adaptation. Experimental results show that larger LLMs outperform smaller ones on the two tasks. Moreover, the proposed distillation method can effectively enhance the performance of smaller LLMs, making them even surpass the original larger LLMs in some cases. This study provides a new approach for talent management in open source communities, which leverages the knowledge of LLMs to improve prediction and recommendation accuracy while reducing computational overhead. A replication package is available at https://github.com/DaSESmartEdu/KDJPPR.

Keywords: Knowledge Distillation · Large Language Models · Talent Management · Open Source Community.

1 Introduction

As the digitalization process of various industries continues to evolve, their demand for digital talents is growing rapidly. A recent report by the National Skills Coalition analyzes 43 million online job postings and finds that 92% of jobs require some type of digital skills[1]. As a gathering place for digital talents, talent management in open source communities is gaining more and more attention, including project recommendation [4,25,24], skill modeling [15,26], developer identification [2,17,8], job title prediction [16,18], and so on.

[1] http://t.newsletterext.worldbank.org/r/?id=h23ec8764,c882dba,c88543a

Existing studies mainly train deep learning models [16,24] or conduct surveys [15,27] to analyze and understand the behaviors of talents in open source communities. With the emergence of large language models (LLMs), it is very interesting to investigate the ability of LLMs on the problem. LLMs are trained with web-scale data and therefore should have a wealth of knowledge pertaining to digital talents, e.g., job titles and digital skills, as well as open source communities, e.g., software, APIs, projects and developers. As such, it is natural to mine the above knowledge of LLMs for talent management in open source communities. The advantages are that one does not have to collect and clean a bunch of data in order to train traditional models, which inevitably takes a lot of preparation time, and can interact with LLMs for talent management in a more semantic way, e.g., instructing an LLM or asking an LLM questions.

In this study, we investigate the ability of LLMs on talent management in open source communities. We particularly focus on two tasks in the literature, namely, project recommendation [25,4] and job title prediction [16,18]. The former aims to recommend suitable open source projects to talents and the latter aims to predict future job titles for talents, based on their past experience in open source communities and the job market. To achieve this, we first construct from the TOSE dataset [16] an evaluation dataset *TM-Eval* , which contains $2,000$ data points. Each data point has a context of the historical job titles and API experience in open source projects of a talent, and a prompt that asks an LLM to determine whether the talent is likely to have the target job title (job title prediction) or API experience (project recommendation) in the future. We use TM-Eval to evaluate LLMs and obtain their performance on the two tasks. Second, we hope to reduce the computational overhead of LLMs for the two tasks and hence distill the knowledge of larger LLMs into smaller ones. We construct from LinkedIn[2] a QA dataset *JA-QA* containing 20,000 pairs, where each QA pair contains a question asking about the APIs required by a job title and the corresponding answer depicting a set of APIs. We use JA-QA to distill the knowledge of a teacher LLM into a student LLM by designing three loss functions. Then we use TM-Eval to evaluate the performance of the student LLM on the above two tasks. Experimental results show that 1) larger LLMs perform better as expected than smaller LLMs on the two tasks and 2) our proposed distillation method can effectively improve the performance of smaller LLMs, making them even surpass the original larger LLMs in some cases.

2 Related Work

2.1 Talent Management in Open Source Communities

The research of talent management in open source communities aims to help developers improve their experience and career, as well as improve efficiency and quality of open source software development. Representative problems include project recommendation, job title prediction and skill modeling.

[2] https://prospeo.io/api/social-url-enrichment

Project recommendation focuses on leveraging user behaviors, content features, and social networks to suggest relevant repositories or projects tailored to developers' interests. For instance, Xu et al. [31] leverage developers' behaviors and project content, utilizing TF-IDF and cosine similarity to calculate project similarities. Sun et al. [25] integrate developer behaviors and project features for GitHub project recommendations. They use TF-IDF to extract characteristics from project descriptions and source code, and apply simulated annealing to optimize the parameters for generating top-N recommendations. Shao et al. [24] utilize text encoding and a graph convolutional network (GCN) to convert paper abstracts and GitHub repository descriptions/tags into vectors, helping users find relevant code repositories for academic papers. Dey et al. [4] employ Doc2Vec embeddings to represent developers, projects, and APIs in the skill Space. This model can predict developers' future API usage and project-joining behavior. Job title prediction aims to predict the job titles via developer expertise. For example, Liu et al. [16] explore the duality between job title prediction and API expertise prediction in open source communities, and propose a dual learning approach to predict job titles in the IT field. Montandon et al. [18] predict GitHub users' technical roles by extracting features like programming languages and project details. They employ Random Forest and Naive Bayes methods for multi-label classification. Skill modeling focuses on accurately representing developers' skill in open source software (OSS). For instance, Liang et al. [15] conduct a survey with 455 OSS contributors and identify relevant skills. They try to understand how contributors grow and share these skills, and derive design implications for incorporating skills into OSS tools and platforms. Sun et al. [26] construct a GitHub social network to integrate developers' social and development activity data, and use the heterogeneous graph neural network to learn developers' technical skill representations. Differently from existing studies, we try to mine the knowledge of LLMs for talent management in open source communities.

2.2 AI-Assisted Talent Management

In recent years, AI-driven techniques have significantly advanced talent management field [21]. Representative tasks include talent search [23,20], career mobility prediction [36,14,28], person-job fitting [33,22], job recommendation [5,30,10],and so on.

Talent search aims to identify qualified candidates by utilizing search queries defined by recruiters or hiring managers to enhance recruitment efficiency. Ozcaglar et al. [20] propose a two-level ranking system for talent search, combining Generalized Linear Mixed models with Gradient Boosted Decision Tree. Ramanath et al. [23] transform recruiters' search queries and candidate features into semantic embeddings and employ learning-to-rank method to sort and identify the suitable candidates for recruitment. Career mobility prediction aims to provide talents' career development paths by analyzing extensive career trajectory data, including work experience, job transitions, skills development and more. Li et al. [14] propose a contextual LSTM model to predict next employer and job title of talent by integrating profile context and career trajectory dynamics. Zhang et al. [37] employ an attention-based heterogeneous graph embedding framework to predict the next employer and job title of talent. Zha et al. [36] employ a trajectory hypergraph to represent career mobility patterns and use attention mechanisms to integrate market features into career trajectory modeling. Person-job fit focuses on measuring the degree of matching between a job posting and a candidate's resume. Qin et al. [22] use LSTM to model the job requirements and candidates' semantic representation and apply ability-aware attention strategies to measure the importance of matching results. Bian et al. [1] construct a job-resume graph to capture implicit correlations between jobs and resumes, and propose a multi-view co-teaching network to integrate text- and relation-based matching module. Yao et al. [33] formulate person-job fit as a graph matching problem and propose a knowledge-enhanced approach that integrates skill extraction with a graph representation learning model. Yu et al. [34] employ contrastive learning to obtain an embedding representation of resumes and jobs, and alleviate data sparsity by data augmentation methods based on EDA [29] and ChatGPT [19]. Job recommendation task aims to provide each candidate with a list of suitable job opportunities derived from the candidate's profile and job postings. Dave et al. [3] utilize job transition, job-skill, and skill co-occurrence networks to jointly learn job and skill embeddings for effective job and skill recommendation. Yu et al. [35] disentangle hierarchical skill factors and model multi-level recruitment interactions to generate a personalized list of job recommendations for candidates. Differing from existing studies, we focus on talent management in open source communities.

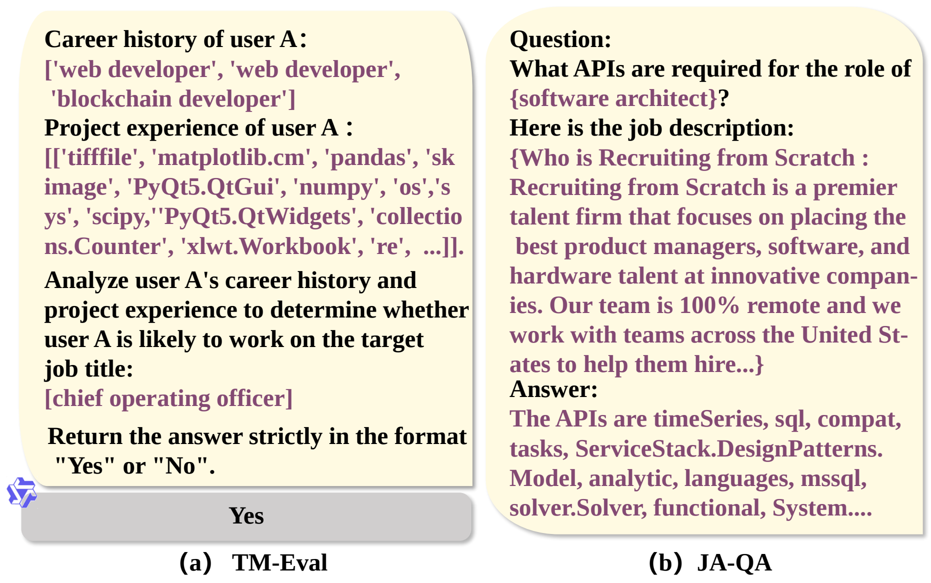
3 Preliminaries

3.1 Task Description

Given a set of historical job titles and API experience in open source projects of a talent, our aim is to predict whether the talent is likely to have a job title (job title prediction) or API experience (project recommendation) in the future. For the latter task, we follow Dey et al. [4] and regard recommending suitable open source projects as predicting the main APIs of a project that the talent works on in the future.

3.2 Dataset Construction

First, to evaluate the ability of LLMs on the above two tasks, we leverage the TOSE dataset [16] and construct an evaluation dataset TM-Eval containing $2,000$ data points. Each data point describes the historical job titles and API experience of a talent and then prompts an LLM to determine whether the talent is likely to take a target job title or use a set of target APIs. The data points are labeled with "Yes" or "No" according to the facts in TOSE. Figure 1(a) gives an example data point in TM-Eval.



Career history of user A：
['web developer', 'web developer', 'blockchain developer']
Project experience of user A :
[['tifffile', 'matplotlib.cm', 'pandas', 'skimage', 'PyQt5.QtGui', 'numpy', 'os','sys', 'scipy,''PyQt5.QtWidgets', 'collections.Counter', 'xlwt.Workbook', 're', ...]].
Analyze user A's career history and project experience to determine whether user A is likely to work on the target job title:
[chief operating officer]
Return the answer strictly in the format "Yes" or "No".

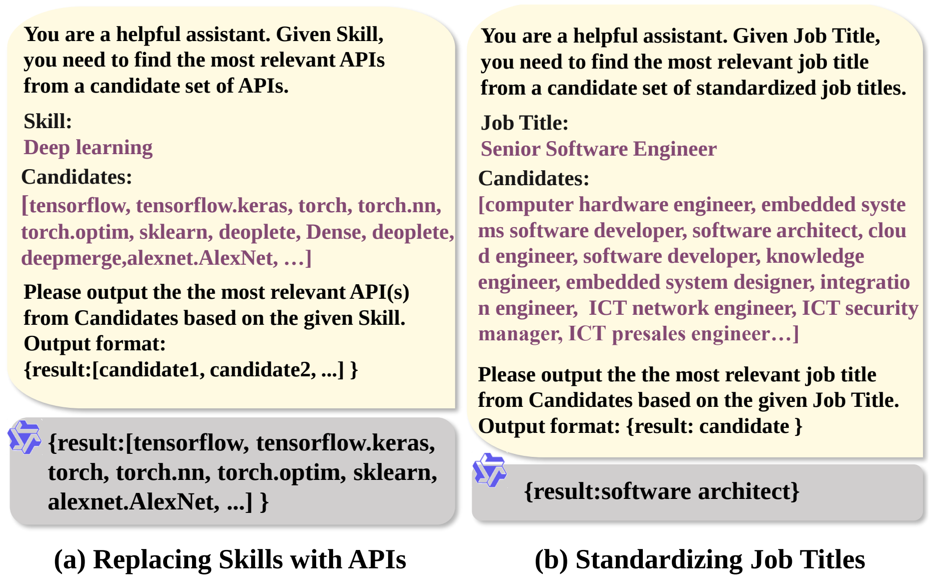
Yes

(a) TM-Eval

Question:
What APIs are required for the role of {software architect}?
Here is the job description:
{Who is Recruiting from Scratch : Recruiting from Scratch is a premier talent firm that focuses on placing the best product managers, software, and hardware talent at innovative companies. Our team is 100% remote and we work with teams across the United States to help them hire...}
Answer:
The APIs are timeSeries, sql, compat, tasks, ServiceStack.DesignPatterns.Model, analytic, languages, mssql, solver.Solver, functional, System....

(b) JA-QA

Fig. 1: The example data points in TM-Eval and JA-QA.



You are a helpful assistant. Given Skill, you need to find the most relevant APIs from a candidate set of APIs.
Skill:
Deep learning
Candidates:
[tensorflow, tensorflow.keras, torch, torch.nn, torch.optim, sklearn, deoplete, Dense, deoplete, deepmerge,alexnet.AlexNet, …]
Please output the the most relevant API(s) from Candidates based on the given Skill.
Output format:
{result:[candidate1, candidate2, ...] }

{result:[tensorflow, tensorflow.keras, torch, torch.nn, torch.optim, sklearn, alexnet.AlexNet, ...] }

(a) Replacing Skills with APIs

You are a helpful assistant. Given Job Title, you need to find the most relevant job title from a candidate set of standardized job titles.
Job Title:
Senior Software Engineer
Candidates:
[computer hardware engineer, embedded systems software developer, software architect, cloud engineer, software developer, knowledge engineer, embedded system designer, integration engineer, ICT network engineer, ICT security manager, ICT presales engineer…]
Please output the the most relevant job title from Candidates based on the given Job Title.
Output format: {result: candidate }

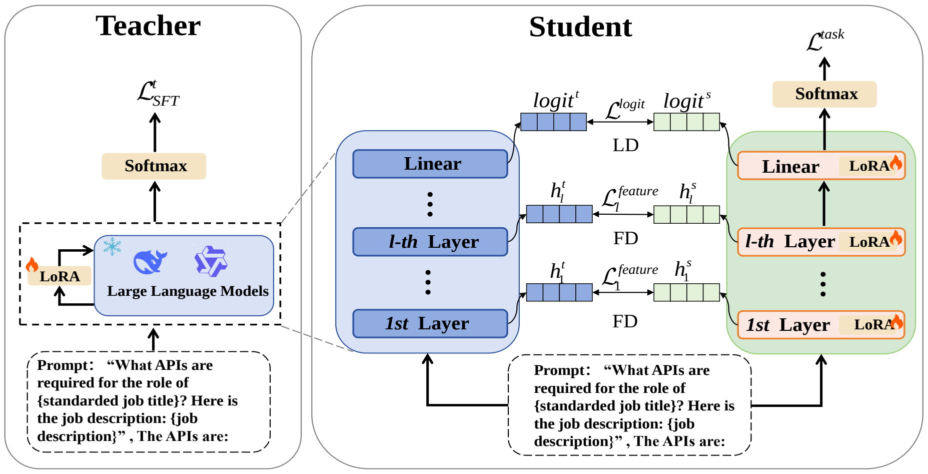
{result:software architect}

(b) Standardizing Job Titles

Fig. 2: The prompting template for replacing skills with APIs and standardizing job titles.

Second, to distill the knowledge from larger LLMs into smaller LLMs, we construct from LinkedIn a QA dataset JA-QA containing $20,000$ QA pairs. Each pair contains a question asking about the required APIs for a job title with the job description, and the corresponding answer depicting a set of APIs. Figure 1(b) gives an example QA pair in JA-QA. We use the questions as input and the answers as output to distill the knowledge in larger LLMs.

It is worth mentioning that the original requirements listed in job descriptions on LinkedIn are skills rather than APIs. In order to better connect the job market and the open source experience in knowledge distillation, we replace the skills with APIs by prompting an LLM, which is depicted in Figure 2(a). The candidate APIs have the most similar embeddings with that of the given skill, where the embeddings are obtained using gte-Qwen2-1.5B-instruct[3], a strong embedding model for semantic similarity measurement. Furthermore, the original job titles listed in job descriptions are not standardized. As such, we also prompt an LLM to standardize each job title, which is depicted in Figure 2(b). The standardized job titles are collected from the ESCO taxonomy[4]. We also use gte-Qwen2-1.5B-instruct to embed the job titles and pick the most similar standardized job titles as the candidates of the given job title.



Prompt: "What APIs are required for the role of {standarded job title}? Here is the job description: {job description}" , The APIs are:

Prompt: "What APIs are required for the role of {standarded job title}? Here is the job description: {job description}" , The APIs are:

Fig. 3: The proposed method.

[3] https://huggingface.co/Alibaba-NLP/gte-Qwen2-1.5B-instruct
[4] https://esco.ec.europa.eu/en/classification/occupation_main

4 Knowledge Distillation for Job-API Correspondence

4.1 Overview

We propose a knowledge distillation method to distill the knowledge in larger LLMs pertaining to the correspondence between jobs and APIs into smaller LLMs, which is depicted in the right part of Figure 3. We design a hierarchical knowledge transfer mechanism through (1) logit-based distillation that aligns output probability distributions between teacher and student LLMs, and (2) feature-based distillation that transfers structural patterns from intermediate transformer layers. This dual distillation mechanism enhances the student LLM's capacity for capturing nuanced job-API associations while maintaining computational efficiency. To ensure the student LLM learns more precise knowledge, we employ Low-Rank Adaptation (LoRA) [13] to fine-tune the teacher model before distillation, which is depicted in the left part of Figure 3. The fine-tune and distillation stages both use the JA-QA dataset.

4.2 Fine-tuning the Teacher LLM

To enhance the knowledge of the teacher LLM pertaining to the correspondence between jobs and APIs, we adopt supervised fine-tuning (SFT) by LoRA. This approach preserves the teacher's reasoning capabilities while integrating domain-specific knowledge.

Particularly, we use the question and job description of each data pair in the JA-QA dataset as the input and the corresponding answer as the target to fine-tune the teacher LLM. The input sequence q is formulated as follows:

> **Prompt:** "What APIs are required for the role of $\{$*job title*$\}$? Here is the job description: $\{$*job description*$\}$. The APIs are:___"

The target a is an API sequence $\{a_1, a_2, \ldots, a_n\}$, where n denotes the length of the API sequence. This structured formulation enables the teacher LLM to effectively capture the relationship between jobs and APIs.

To efficiently adapt the teacher LLM, we employ LoRA for weight matrix adaptation in the transformer layers. Instead of updating all model parameters, LoRA introduces trainable low-rank matrices $A \in \mathbb{R}^{r\times k}$ and $B \in \mathbb{R}^{d\times r}$ to adjust specific weight projections. Here, $r \ll \min(d, k)$ is the rank of the low-rank matrices, controlling the number of trainable parameters. For a given weight matrix $W \in \mathbb{R}^{d\times k}$, the adapted transformation is:

$$h = (W + \Delta W)q = (W + BA)q, \tag{1}$$

where h denotes the output of each transformer layer. At the final output layer, the teacher LLM minimizes the negative log probability for generating the target sequence a:

$$\mathcal{L}^t_{SFT} = -\sum_{i=1}^{n} \log P(a_i | a_{<i}, q; \theta_t, B, A), \tag{2}$$

where θ_t denotes the parameters of the teacher LLM and n is the API sequence length. This objective forces the teacher LLM to generate the correct API sequence while maintaining linguistic coherence.

4.3 Knowledge Distillation

While larger LLMs excel in semantic understanding, their high inference memory and latency often make them unsuitable for deployment in real-life systems. To address this issue, we propose to transfer the knowledge of a larger and more complex teacher LLM into a smaller and more efficient student LLM using knowledge distillation [7,12]. After distillation, the student LLM obtains the teacher's knowledge to map jobs to APIs while significantly reducing computational overhead, making it feasible for real-life deployment.

The student LLM shares a similar architecture with the teacher LLM and has fewer layers and smaller hidden dimensions. The input and the target output of the student LLM are the same as those of the teacher LLM, as depicted in Section 4.2. To transfer the teacher LLM's knowledge to the student LLM, we employ a multi-layer knowledge distillation approach, which distills both the low-level features and the logit distributions. Particularly, we design three loss functions for distillation, including logit-based distillation, feature-based distillation, and task-specific fine-tuning with LoRA.

Logit-based Distillation Logit-based distillation transfers the teacher's output distribution to the student by minimizing the Kullback-Leibler (KL) divergence between their logits. Let L^t and L^s denote the logits of the teacher and student LLMs, respectively. To stabilize the training process, we adopt a temperature scaling parameter τ for the logits, which softens the probability distributions. Then, the probability distributions p^t and p^s are obtained by applying the softmax function on the scaled logits. This process can be computed as:

$$p^t = \text{softmax}(\frac{L^t}{\tau}), \quad p^s = \text{softmax}(\frac{L^s}{\tau}). \tag{3}$$

To handle case where the teacher and student LLMs have different output dimensions (e.g., due to different vocabulary sizes), we introduce a padding mechanism. If the student LLM's output dimension is smaller than the teacher's, we pad the student logits with zeros to match the teacher's output dimension, and vice versa. This ensures that the KL divergence can be computed correctly. The padding operation is defined as:

$$\text{pad\_logits}(p^s, p^t) = \begin{cases} (\text{concat}(p^s, \mathbf{0}), p^t) & \text{if } \dim(p^s) < \dim(p^t), \\ (p^s, \text{concat}(p^t, \mathbf{0})) & \text{if } \dim(p^s) > \dim(p^t), \\ (p^s, p^t) & \text{otherwise}, \end{cases} \tag{4}$$

where $\mathbf{0}$ is a zero vector of appropriate size.

The logit-based distillation loss is then computed as the KL divergence between the teacher's and student's output distributions:

$$\mathrm{KL}(p^t|p^s) = p^t(a|q)\log\frac{p^t(a|q)}{p^s(a|q)}. \tag{5}$$

To ensure more stable training, the logit-based distillation loss is scaled by the square of the temperature τ and normalized by the maximum sequence length n:

$$\mathcal{L}^{logit} = \mathrm{KL}(p^t|p^s)\cdot\frac{\tau^2}{n}. \tag{6}$$

Feature-based Distillation Feature-based distillation aligns the intermediate representations (hidden states) of the teacher and student LLMs [38]. This approach ensures that the student LLM not only mimics the teacher's final output but also learns to replicate its internal representation.

Suppose that the student LLM has l intermediate layers. Feature-based distillation is conducted between the l layers and the first l layers of the teacher LLM, because the student LLM has fewer layers. Let h_i^t and h_i^s denote the hidden states of the teacher and student models at layer i, respectively. Since the teacher and student LLMs may have different sizes for the hidden states, we introduce an adaptation layer to project the student's hidden states into the teacher's representation space. At the i^{th} layer, the adaptation layer contains a trainable linear transformation W_i that maps the student's hidden state h_i^s into the teacher's hidden state space. Similarly, we apply a temperature scaling parameter τ to the hidden states. The probability distributions at layer i p_i^t and p_i^s are obtained by applying the softmax function to the temperature-scaled hidden states:

$$p_i^t = \mathrm{softmax}(\frac{h_i^t}{\tau}), \quad p_i^s = \mathrm{softmax}(\frac{W_i\cdot h_i^s}{\tau}). \tag{7}$$

The feature-based distillation loss for layer i is then computed as:

$$\mathcal{L}_i^{feature} = \mathrm{KL}(p_i^t|p_i^s) = p_i^t\log\frac{p_i^t}{p_i^s}. \tag{8}$$

The overall feature-based distillation loss is the average of the layer-wise losses:

$$\mathcal{L}^{feature} = \frac{1}{l}\sum_{i=1}^{l}\mathcal{L}_i^{feature}, \tag{9}$$

where l is the number of layers in the student LLM.

Task-specific Fine-tuning In addition to distillation, the student LLM is also fine-tuned using the JA-QA dataset like the teacher LLM does. The task-specific loss is the negative log likelihood for generating the target API sequence a:

$$\mathcal{L}^{task} = -\sum_{i=1}^{n}\log P(a_i|a_{<i}, q; \theta_s), \tag{10}$$

where θ_s denotes the parameters of the student LLM.

The overall loss for training the student LLM is a weighted combination of the logit-based distillation loss, the feature-based distillation loss, and the task-specific fine-tuning loss:

$$\mathcal{L}^{student} = \alpha\mathcal{L}^{logit} + \beta\mathcal{L}^{feature} + \mathcal{L}^{task}, \tag{11}$$

where α and β are hyperparameters. During training, the student LLM minimizes $\mathcal{L}^{student}$ using LoRA.

5 Performance Evaluation

We aim to answer the following four research questions:

- **RQ1:** How do the latest representative LLMs perform on project recommendation and job title prediction for open source talents?
- **RQ2:** How do our fine-tuning and knowledge distillation approaches improve the performance of LLMs?
- **RQ3:** How do the different components influence the LLMs' performance?
- **RQ4:** How do the hyperparameters influence the LLMs' performance?

5.1 Datasets

We use the JA-QA dataset depicted in Section 3.2 to fine-tune and distill knowledge from the teacher LLMs, which contains 20,000 QA pairs. We use TM-Eval depicted in Section 3.2 to evaluate the performance of project recommendation and job title prediction, which contains 1,000 data points for each task.

5.2 Comparative Methods

We use four traditional models as baselines, including Skill-Space [4], NeuMF [11], APJFNN [39], and WEPJF [6]. We train these models using the TOSE dataset[5] except those used to construct TM-Eval. Skill-Space embeds users and APIs in a shared space and we use the embeddings for job title prediction and project recommendation. NeuMF leverages user-job title and user-API interactions for prediction and recommendation. APJFNN uses RNNs and attention mechanisms to capture key information and semantic relationships among job titles and API sequences. WEPJF models user preferences in job title and API sequences using contrastive learning. We cannot compare with DualJE [16] because we don't have the datasets to train the API and job title models. For LLMs, we evaluate the performance of Qwen2.5-14B, -7B and -3B [32], and DeepSeek-R1-Distill-Qwen-14B, -7B and -1.5B [9][6]. Then, we fine-tune Qwen2.5-14B and R1-Distill-Qwen-14B using JA-QA and use the fine-tuned models as the teachers, i.e., Qwen2.5-14B w/SFT and R1-Distill-Qwen-14B w/SFT. Finally, we use Qwen2.5-7B and

[5] We convert the data format to adapt the data to our tasks.

[6] We omit 'DeepSeek' when necessary.

-3B, and R1-Distill-Qwen-7B and -1.5B as the students and conduct knowledge distillation. The results models are Qwen2.5-7B w/KD, Qwen2.5-3B w/KD, R1-Distill-Qwen-7B w/KD and R1-Distill-Qwen-1.5B w/KD.

5.3 Hyperparameters and Evaluation Metrics

The training process is conducted in parallel on two NVIDIA A800 80G GPUs, with a batch size of 32 per GPU. The models are trained for 3 epochs with a learning rate of 1e-5, which is adjusted using a cosine decay strategy. The optimizer is AdamW, and the distillation temperature coefficient is set to 3, which regulates the influence of the teacher LLM's soft targets on the student LLM. The loss balance factors α and β are set to 1.0 and 0.5, respectively, after tuning.

Since the predicted results are binary, e.g. Yes/No, we adopt the following four evaluation metrics: Accuracy (Acc), Precision (Pre), Recall (Rec), and F1-score (F1).

5.4 RQ1: Performance of the Latest LLMs

The first part of Table 1 reports the performance of the four traditional models. The second part reports the performance of Qwen2.5 and the fourth part reports

Table 1: Main Results.

Methods	Job Title Prediction				Project Recommendation			
	Pre	Rec	F1	Acc	Pre	Rec	F1	Acc
Skill Space [4]	55.85	54.31	55.07	54.82	52.36	54.28	53.30	53.57
NeuMF [11]	56.22	57.13	56.67	56.85	55.69	54.06	54.86	55.59
APJFNN [22]	57.76	56.68	57.21	57.39	56.15	55.47	55.81	56.24
WEPJM [6]	59.23	61.12	60.16	60.31	58.35	59.57	58.95	59.46
Qwen2.5-14B	72.49	69.94	71.19	72.36	68.98	67.79	68.38	69.25
Qwen2.5-7B	64.45	62.87	63.56	66.58	62.85	61.30	62.07	64.46
Qwen2.5-3B	61.50	60.34	60.91	61.27	58.62	59.82	59.21	61.49
Qwen2.5-14B w/SFT	**74.78** (↑ 2.29)	**72.66** (↑ 2.72)	**73.70** (↑ 2.51)	**73.95** (↑ 1.59)	**72.24** (↑ 3.26)	**71.15** (↑ 3.36)	**71.69** (↑ 3.31)	**72.67** (↑ 3.42)
Qwen2.5-7B w/KD	**69.72** (↑ 5.27)	**68.65** (↑ 5.78)	**69.18** (↑ 5.62)	**70.45** (↑ 3.87)	**66.39** (↑ 3.54)	**65.72** (↑ 4.42)	**66.05** (↑ 3.98)	**67.35** (↑ 2.89)
Qwen2.5-3B w/KD	**65.83** (↑ 4.33)	**65.39** (↑ 5.05)	**65.61** (↑ 4.70)	**66.15** (↑ 4.88)	**63.54** (↑ 4.92)	**63.89** (↑ 4.07)	**63.71** (↑ 4.50)	**64.02** (↑ 2.53)
R1-Distill-Qwen-14B	73.15	69.32	71.18	73.65	69.54	68.72	69.13	70.12
R1-Distill-Qwen-7B	66.62	64.45	65.52	67.79	63.39	64.08	63.73	65.38
R1-Distill-Qwen-1.5B	62.05	62.85	62.45	63.07	59.04	60.61	59.81	61.48
R1-Distill-Qwen-14B w/SFT	**75.09** (↑ 1.94)	**73.48** (↑ 4.16)	**74.27** (↑ 3.09)	**75.41** (↑ 1.76)	**71.43** (↑ 1.89)	**72.15** (↑ 3.43)	**71.79** (↑ 2.66)	**73.75** (↑ 3.63)
R1-Distill-Qwen-7B w/KD	**70.25** (↑ 3.63)	**69.91** (↑ 5.46)	**70.08** (↑ 4.56)	**71.32** (↑ 3.53)	**67.82** (↑ 4.43)	**67.31** (↑ 3.23)	**67.56** (↑ 3.83)	**69.03** (↑ 3.65)
R1-Distill-Qwen-1.5B w/KD	**66.34** (↑ 4.29)	**65.23** (↑ 2.38)	**65.78** (↑ 3.33)	**66.53** (↑ 3.46)	**63.52** (↑ 4.48)	**64.47** (↑ 3.86)	**63.99** (↑ 4.18)	**65.81** (↑ 4.33)

the performance of DeepSeek-R1-Distill-Qwen. We observe two points. First, the LLMs perform significantly better than the four traditional models, indicating that the LLMs have already been equipped with rich knowledge pertaining to talent management in open source communities. Second, for either Qwen2.5 or DeepSeek-R1-Distill-Qwen, the larger LLMs perform better than the smaller LLMs, indicating the former have richer knowledge. The observation is consistent with the scaling laws.

5.5 RQ2: Performance of the Fine-tuned and Distilled LLMs

The third and fifth parts of Table 1 report performance of the fine-tuned and distilled Qwen2.5 and DeepSeek-R1-Distill-Qwen, respectively. We observe that all LLMs are improved on the two tasks on all metrics, compared to their original variants before fine-tuning or distillation. First, we observe that the teacher LLMs, i.e., Qwen2.5-14B and R1-Distill-Qwen-14B, are further improved after fine-tuning, due to the injection of domain knowledge. Second, the improvements of the four student LLMs are even larger, indicating the effectiveness of the proposed distillation method. Moreover, Qwen2.5-7B w/KD and R1-Distill-Qwen-7B w/KD perform close to the original Qwen2.5-14B and R1-Distill-Qwen-14B, and Qwen2.5-3B w/KD and R1-Distill-Qwen-1.5B w/KD surpass the performance of the original Qwen2.5-7B and R1-Distill-Qwen-7B, indicating that our distillation method makes smaller LLMs outperform the original larger LLMs.

Table 2: Ablation Study

Model	Method	Job Title Prediction				Project Recommendation			
		Pre	Rec	F1	Acc	Pre	Rec	F1	Acc
Qwen2.5-7B	SFT	65.81	63.92	64.85	67.29	64.94	62.88	63.89	65.41
	KD w/ Orig Teacher	66.37	65.25	65.81	68.13	65.59	65.01	65.30	65.68
	KD w/o feature	67.15	67.49	67.32	68.89	65.78	64.85	65.31	66.34
	KD w/o logit	67.34	66.95	67.14	69.58	66.13	65.48	65.80	66.64
	Ours	**69.72**	**68.65**	**69.18**	**70.45**	**66.39**	**65.72**	**66.05**	**67.35**
Qwen2.5-3B	SFT	63.46	62.98	63.22	64.17	60.15	62.42	62.16	62.43
	KD w/ Orig Teacher	63.69	63.14	63.41	64.58	61.88	62.95	62.41	63.14
	KD w/o feature	64.33	63.96	64.14	64.91	62.58	62.75	62.66	63.43
	KD w/o logit	64.76	64.91	64.83	65.25	62.92	63.27	63.09	63.61
	Ours	**65.83**	**65.39**	**65.61**	**66.15**	**63.54**	**63.89**	**63.71**	**64.02**
R1-Distill-Qwen-7B	SFT	67.75	65.59	66.65	68.52	65.02	64.61	64.81	67.29
	KD w/ Orig Teacher	67.68	66.13	66.90	68.76	65.59	65.01	65.30	67.38
	KD w/o feature	68.44	67.53	67.98	69.45	65.87	64.94	65.40	67.61
	KD w/o logit	68.67	68.16	68.41	69.96	66.79	67.41	67.10	68.32
	Ours	**70.25**	**69.91**	**70.08**	**71.32**	**67.82**	**67.31**	**67.56**	**69.03**
R1-Distill-Qwen-1.5B	SFT	62.96	63.33	63.14	63.85	60.45	61.24	60.84	62.27
	KD w/ Orig Teacher	63.47	64.03	63.75	64.11	61.11	62.28	61.69	62.93
	KD w/o feature	63.85	63.96	63.90	64.48	61.74	62.35	62.04	63.19
	KD w/o logit	64.29	64.58	64.43	65.47	62.34	63.12	62.73	64.11
	Ours	**66.34**	**65.23**	**65.78**	**66.53**	**63.52**	**64.47**	**63.99**	**65.81**

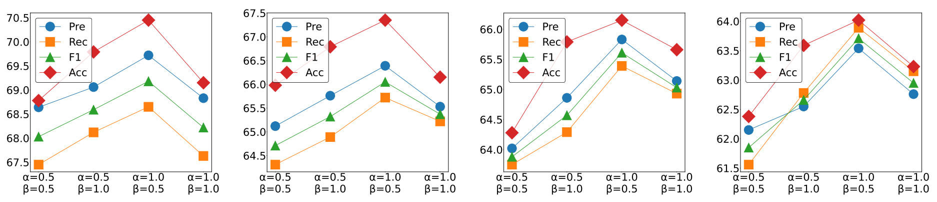
5.6 RQ3: Ablation Study

For each student LLM, we design four ablation models. First, we directly fine-tune the LLM using the JA-QA dataset, which is denoted by SFT. Second, we distill from the original teacher LLM, i.e., the teacher LLM without fine-tuning on JA-QA, which is denoted by KD w/ Orig Teacher. Third, we remove the feature-based distillation loss and the logit-based distillation loss from the total distillation loss, respectively, which are denoted by KD w/o feature and KD w/o logit.

The results are reported in Table 2. We observe that all ablation models perform worse than our distilled student model in each LLM group, which indicates the effectiveness of the components in our proposed distillation method. Particularly, the significant performance gap between SFT and the distilled student LLM shows the effect of using both feature-based distillation and logit-based distillation.

5.7 RQ4: Hyperparameter Analysis

The most important hyperparameters in our distillation method are the loss balance factors α and β. We vary their values for both student LLMs. Figure 4 shows the results of Qwen2.5 on the two tasks. We observe that the best performance is achieved when $\alpha = 1.0$ and $\beta = 0.5$, for both tasks and both LLM sizes. This may indicate that the logit-based distillation loss is more important than the feature-based distillation loss. Figure 5 shows the results of DeepSeek-R1-Distill-Qwen on the two tasks, and we observe similar results with that of Qwen2.5.

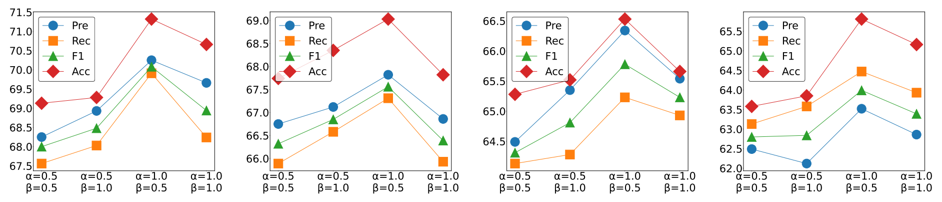


(a) Job Title Prediction (7B) (b) Project Recommendation (7B) (c) Job Title Prediction (3B) (d) Project Recommendation (3B)

Fig. 4: Varying α and β for Qwen2.5.

6 Discussion and Conclusion

This research focuses on leveraging large language models (LLMs) for talent management in open source communities, specifically in the job title prediction and project recommendation tasks. By constructing the TM-Eval and JA-QA



(a) Job Title Prediction (7B) (b) Project Recommendation (7B) (c) Job Title Prediction (1.5B) (d) Project Recommendation (1.5B)

Fig. 5: Varying α and β for DeepSeek-R1-Distill-Qwen.

datasets, we evaluate the performance of the latest representative LLMs and propose a knowledge distillation method to transfer knowledge from larger LLMs to smaller ones. The experimental results lead to several significant findings. First, LLMs demonstrate remarkable superiority over traditional models in both tasks, and larger LLMs outperform smaller ones. Second, our fine-tuning and knowledge distillation approaches are highly effective. Fine-tuning injects domain-specific knowledge into teacher LLMs, while distillation enables student LLMs to significantly improve their performance.

However, there are still areas for improvement. For example, the datasets used may not comprehensively cover all aspects of talent management in open source communities, and the distillation method could be further optimized to handle more complex scenarios, such as distilling into different LLMs. In addition, the study primarily focuses on two tasks, which may lead to a relatively narrow research scope without considering more diverse scenarios. Future research could explore expanding the datasets, improving the distillation algorithm, and integrating more advanced techniques to enhance the performance and generalization ability of LLMs in open source talent management tasks. It could be valuable to explore other talent management tasks in open source communities beyond the current focus, such as contributor profiling, skill development tracking, and the attraction and retention of open source talents. Overall, this study offers a solid foundation for future research in this field.

Acknowledgments. This work is supported by the grant from the National Natural Science Foundation of China (Grant No. 62277017 and 62137001).

References

1. Bian, S., Chen, X., Zhao, W.X., Zhou, K., Hou, Y., Song, Y., Zhang, T., Wen, J.R.: Learning to match jobs with resumes from sparse interaction data using multi-view co-teaching network. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management. pp. 65–74 (2020)
2. Bock, T., Alznauer, N., Joblin, M., Apel, S.: Automatic core-developer identification on github: A validation study. ACM Transactions on Software Engineering and Methodology **32**(6), 1–29 (2023)

3. Dave, V.S., Zhang, B., Al Hasan, M., AlJadda, K., Korayem, M.: A combined representation learning approach for better job and skill recommendation. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management. pp. 1997–2005 (2018)
4. Dey, T., Karnauch, A., Mockus, A.: Representation of developer expertise in open source software. In: 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE). pp. 995–1007. IEEE (2021)
5. Du, Y., Luo, D., Yan, R., Wang, X., Liu, H., Zhu, H., Song, Y., Zhang, J.: Enhancing job recommendation through llm-based generative adversarial networks. In: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 8363–8371 (2024)
6. Gong, Z., Song, Y., Zhang, T., Wen, J.R., Zhao, D., Yan, R.: Your career path matters in person-job fit. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 38, pp. 8427–8435 (2024)
7. Gou, J., Yu, B., Maybank, S.J., Tao, D.: Knowledge distillation: A survey. International Journal of Computer Vision **129**(6), 1789–1819 (2021)
8. Greene, G.J., Fischer, B.: Cvexplorer: Identifying candidate developers by mining and exploring their open source contributions. In: Proceedings of the 31st IEEE/ACM international conference on automated software engineering. pp. 804–809 (2016)
9. Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al.: Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. arXiv preprint arXiv:2501.12948 (2025)
10. Han, X., Zhu, C., Hu, X., Qin, C., Zhao, X., Zhu, H.: Adapting job recommendations to user preference drift with behavioral-semantic fusion learning. In: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. pp. 1004–1015 (2024)
11. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.S.: Neural collaborative filtering. In: Proceedings of the 26th international conference on world wide web. pp. 173–182 (2017)
12. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
13. Hu, E.J., yelong shen, Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: LoRA: Low-rank adaptation of large language models. In: International Conference on Learning Representations (2022)
14. Li, L., Jing, H., Tong, H., Yang, J., He, Q., Chen, B.C.: Nemo: Next career move prediction with contextual embedding. In: Proceedings of the 26th International Conference on World Wide Web Companion. pp. 505–513 (2017)
15. Liang, J.T., Zimmermann, T., Ford, D.: Understanding skills for oss communities on github. In: Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. pp. 170–182 (2022)
16. Liu, X., Wang, Y., Dong, Q., Lu, X.: Job title prediction as a dual task of expertise prediction in open source software. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 381–396. Springer (2024)
17. Montandon, J.E., Silva, L.L., Valente, M.T.: Identifying experts in software libraries and frameworks among github users. In: 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR). pp. 276–287. IEEE (2019)
18. Montandon, J.E., Valente, M.T., Silva, L.L.: Mining the technical roles of github users. Information and Software Technology **131**, 106485 (2021)

19. OpenAI: Introducing ChatGPT (2022), https://openai.com/blog/chatgpt
20. Ozcaglar, C., Geyik, S., Schmitz, B., Sharma, P., Shelkovnykov, A., Ma, Y., Buchanan, E.: Entity personalized talent search models with tree interaction features. In: The World Wide Web Conference. pp. 3116–3122 (2019)
21. Qin, C., Zhang, L., Cheng, Y., Zha, R., Shen, D., Zhang, Q., Chen, X., Sun, Y., Zhu, C., Zhu, H., et al.: A comprehensive survey of artificial intelligence techniques for talent analytics. arXiv preprint arXiv:2307.03195 (2023)
22. Qin, C., Zhu, H., Xu, T., Zhu, C., Jiang, L., Chen, E., Xiong, H.: 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. pp. 25–34 (2018)
23. Ramanath, R., Inan, H., Polatkan, G., Hu, B., Guo, Q., Ozcaglar, C., Wu, X., Kenthapadi, K., Geyik, S.C.: Towards deep and representation learning for talent search at linkedin. In: Proceedings of the 27th ACM international conference on information and knowledge management. pp. 2253–2261 (2018)
24. Shao, H., Sun, D., Wu, J., Zhang, Z., Zhang, A., Yao, S., Liu, S., Wang, T., Zhang, C., Abdelzaher, T.: paper2repo: Github repository recommendation for academic papers. In: Proceedings of The Web Conference 2020. pp. 629–639 (2020)
25. Sun, X., Xu, W., Xia, X., Chen, X., Li, B.: Personalized project recommendation on github. Science China Information Sciences **61**, 1–14 (2018)
26. Sun, Y., Wu, J., Zhao, X., Xu, H., Wang, S., Zhang, J., Zhu, Y., Huang, G.: Automatically deriving developers' technical expertise from the github social network. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. pp. 2462–2463 (2024)
27. Vadlamani, S.L., Baysal, O.: Studying software developer expertise and contributions in stack overflow and github. In: 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME). pp. 312–323. IEEE (2020)
28. Wang, C., Zhu, H., Hao, Q., Xiao, K., Xiong, H.: Variable interval time sequence modeling for career trajectory prediction: Deep collaborative perspective. In: Proceedings of the Web Conference 2021. pp. 612–623 (2021)
29. Wei, J., Zou, K.: 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). pp. 6382–6388 (2019)
30. Wu, L., Qiu, Z., Zheng, Z., Zhu, H., Chen, E.: Exploring large language model for graph data understanding in online job recommendations. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 38, pp. 9178–9186 (2024)
31. Xu, W., Sun, X., Hu, J., Li, B.: Repersp: recommending personalized software projects on github. In: 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME). pp. 648–652. IEEE (2017)
32. Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., et al.: Qwen2. 5 technical report. arXiv preprint arXiv:2412.15115 (2024)
33. Yao, K., Zhang, J., Qin, C., Wang, P., Zhu, H., Xiong, H.: Knowledge enhanced person-job fit for talent recruitment. In: 2022 IEEE 38th International Conference on Data Engineering (ICDE). pp. 3467–3480. IEEE (2022)
34. Yu, X., Zhang, J., Yu, Z.: Confit: Improving resume-job matching using data augmentation and contrastive learning. In: Proceedings of the 18th ACM Conference on Recommender Systems. pp. 601–611 (2024)

35. Yu, X., Qin, C., Zhang, Q., Zhu, C., Ma, H., Zhang, X., Zhu, H.: Disco: A hierarchical disentangled cognitive diagnosis framework for interpretable job recommendation. In: 2024 IEEE International Conference on Data Mining (ICDM). pp. 590–599 (2024)
36. Zha, R., Sun, Y., Qin, C., Zhang, L., Xu, T., Zhu, H., Chen, E.: Towards unified representation learning for career mobility analysis with trajectory hypergraph. ACM Transactions on Information Systems **42**(4), 1–28 (2024)
37. Zhang, L., Zhou, D., Zhu, H., Xu, T., Zha, R., Chen, E., Xiong, H.: Attentive heterogeneous graph embedding for job mobility prediction. In: Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining. pp. 2192–2201 (2021)
38. Zhang, L., Shi, Y., Shi, Z., Ma, K., Bao, C.: Task-oriented feature distillation. Advances in Neural Information Processing Systems **33**, 14759–14771 (2020)
39. Zhu, C., Zhu, H., Xiong, H., Ma, C., Xie, F., Ding, P., Li, P.: Person-job fit: Adapting the right talent for the right job with joint representation learning. ACM Transactions on Management Information Systems (TMIS) **9**(3), 1–17 (2018)