Adapting Job Recommendations to User Preference Drift with Behavioral-Semantic Fusion Learning

Xiao Han
hahahenha@gmail.com
City University of Hong Kong
Hong Kong, China

Chen Zhu*
zc3930155@gmail.com
Career Science Lab, BOSS Zhipin
University of Science and Technology of China
Beijing, China

Xiao Hu
huxiao@kanzhun.com
Career Science Lab, BOSS Zhipin
Beijing, China

Chuan Qin
chuanqin0426@gmail.com
Career Science Lab, BOSS Zhipin
Beijing, China

Xiangyu Zhao*
xianzhao@cityu.edu.hk
City University of Hong Kong
Hong Kong, China

Hengshu Zhu*
zhuhengshu@kanzhun.com
Career Science Lab, BOSS Zhipin
Beijing, China

ABSTRACT

Job recommender systems are crucial for aligning job opportunities with job-seekers in online job-seeking. However, users tend to adjust their job preferences to secure employment opportunities continually, which limits the performance of job recommendations. The inherent frequency of preference drift poses a challenge to promptly and precisely capture user preferences. To address this issue, we propose a novel session-based framework, BISTRO, to timely model user preference through fusion learning of semantic and behavioral information. Specifically, BISTRO is composed of three stages: 1) coarse-grained semantic clustering, 2) fine-grained job preference extraction, and 3) personalized top-k job recommendation. Initially, BISTRO segments the user interaction sequence into sessions and leverages session-based semantic clustering to achieve broad identification of person-job matching. Subsequently, we design a hypergraph wavelet learning method to capture the nuanced job preference drift. To mitigate the effect of noise in interactions caused by frequent preference drift, we innovatively propose an adaptive wavelet filtering technique to remove noisy interaction. Finally, a recurrent neural network is utilized to analyze session-based interaction for inferring personalized preferences. Extensive experiments on three real-world offline recruitment datasets demonstrate the significant performances of our framework. Significantly, BISTRO also excels in online experiments, affirming its effectiveness in live recruitment settings. This dual success underscores the robustness and adaptability of BISTRO. The source code is available at https://github.com/Applied-Machine-Learning-Lab/BISTRO.

CCS CONCEPTS

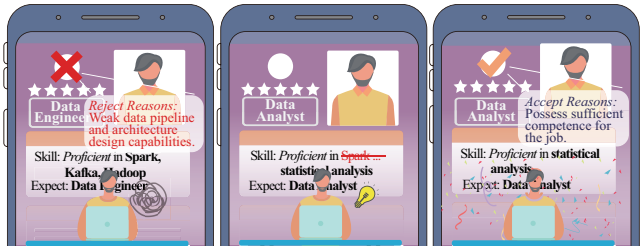
• **Information systems** → **Enterprise applications**; • **Computing methodologies** → *Neural networks*; • **Applied computing**;

KEYWORDS

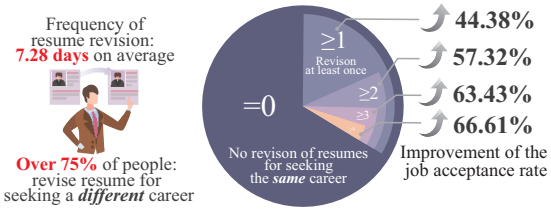
Session-based Recommendation, Interaction Hypergraph, Hypergraph Wavelet Learning, Job Recommender System

ACM Reference Format:
Xiao Han, Chen Zhu, Xiao Hu, Chuan Qin, Xiangyu Zhao, and Hengshu Zhu. 2024. 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 (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3637528.3671759

1 INTRODUCTION



(a) A scenario illustrating a user revise the resume to target a different job position



(b) Statistics on revision of resumes

Figure 1: A toy example in the online recruitment platform.

* Chen Zhu, Xiangyu Zhao, and Hengshu Zhu are the corresponding authors.
†This work was accomplished by the first author while interning at BOSS Zhipin

KDD '24, August 25–29, 2024, Barcelona, Spain

ACM ISBN 979-8-4007-0490-1/24/08.
https://doi.org/10.1145/3637528.3671759

In recent years, a clear trend has emerged where online recruitment platforms are undergoing rapid development and surpassing local job markets as the primary recruitment channel. According to the market research report [16], the global online recruitment market size, which was valued at USD 29.29 billion in 2021, is projected to expand to USD 58.16 billion by 2030, with a compound annual growth rate of 7.1% from 2023 to 2030. Consequently, the job recommender system, a primary method in online recruitment, has also underscored its significance.

However, unlike conventional recommendation settings, recruitment is inherently a bidirectional selection process [27, 45]. This dynamic necessitates that not only should the jobs align with the expectations of the job-seekers, but the job-seekers must also satisfy the requirements set by the employers. As a toy example illustrated in Figure 1(a), active users refine job preferences in order to secure employment opportunities, reflected in the user-job interactions and the resume revision. Initially, the individual seeks employment as a data engineer but encounters setbacks due to the shortage of data pipeline and architecture design capabilities. This experience led to the realization that his skills may be more aligned with data analyst positions, which shows the preference drift. Motivated by this insight, the job-seeker refines his resume to better match data analyst positions, culminating in the successful acquisition of an offer. Furthermore, Figure 1(b) provides some statistical insights about resume update behaviors collected from a prominent online recruitment platform in China over a six-month period to imply the prevalence of this phenomenon. It highlights that, on average, job-seekers tend to revise their resumes every 7.28 days when they do not secure employment and those who update their resumes frequently are at least 44% more successful in receiving job offers than the rest of those who are not proactive, underscoring the effectiveness of consistently optimizing resumes during the job-seeking process. Another interesting insight is over three-quarters of individuals would change their job-seeking objectives when refining their resumes, indicating a strong correlation between preference drift and resume refinement.

The inherent propensity for frequent preference drift implies the necessity of nuanced and timely modeling of user preference in job recommendation, which limits the effectiveness of both content and behavior-based recommendation algorithms. On the one hand, content-based job recommender systems [26, 56, 57] strive to precisely profile job-seeker capabilities for better aligning their qualifications with recommendation results. However, they fall short in capturing job-seeker nuanced job preferences. On the other hand, although behavior-based recommendation, especially session-aware recommendation [33, 35, 61], provides a solution to timely track job-seeker preferences by short-term interactions, their performance is limited by the length of session-based interactions and thus is vulnerable to noise, which is common in practical job recommender systems.

To address the above issues, we propose a novel session-based framework in this paper, namely **BISTRO** (**B**ehav**I**oral-**S**eman**T**ic fusion for job **R**ec**O**mmendation). Specifically, the framework contains three modules: 1) A coarse-grained semantic clustering module. It groups users or jobs based on semantics to facilitate the broad identification of person-job matching. 2) A fine-grained job preference extraction module. In this module, a multi-granular interaction hypergraph is constructed for capturing preference drift and a novel spectral hypergraph wavelet learning method is performed on this graph to capture preference drift and denoising preference feature. 3) A personalized top-k job recommendation module. It utilizes a recurrent neural network to analyze short-term sequential behavior and infer personalized preferences, thereby generating recommendations for the top-k jobs. In summary, our contributions are demonstrated as follows:

- To capture the job preference drift in job-seeking, we innovatively propose a session-based recommendation framework, BISTRO, which leverages behavioral-semantic fusion learning for job recommendation.
- To mitigate the effect of noise in user interactions, we develop a novel graph wavelet kernel-based hypergraph convolutional neural network, which utilizes spectral graph theory to filter noisy data adaptively.
- Extensive experiments on three real-world recruitment datasets and a half-week online algorithm deployment on an online recruitment platform demonstrate the effectiveness and efficiency of the framework BISTRO.

The remainder of this paper is organized as follows: In Section 3, a crucial definition and the problem statement are discussed. In Section 4, the proposed framework is introduced. Section 5 provides the performance evaluation. Section 2 discusses the related literature, and Section 6 concludes the paper.

2 RELATED WORK

2.1 Job Recommender System

In job recommender systems, various studies have been proposed to match job seekers with recruiters. As highlighted by [13, 23, 24, 30, 37, 38, 49, 52, 62, 63, 65], while traditional recommender systems are adept at predicting job seekers' preferences, a key to augmenting the system's overall effectiveness lies in addressing the issue of preference drift. Conventionally, this challenge has been approached through feature engineering in a certain degree, utilizing content data to capture evolving preferences [12, 21, 66]. Efforts have also been made [15, 19, 29] to integrate clustering into recommender systems, tackling this problem at the model level by grouping similar users or jobs based on minimal user/job contents. Although the user and job representations could be enhanced by these refined features, they heavily rely on the results of semantic analysis of fixed content. To overcome these limitations, we introduce a behavioral-semantic fusion framework that merges content-driven and interaction-based methodologies, offering a more comprehensive and adaptive solution to the challenge of preference drift.

2.2 Recommendation with Graphs

Graph neural networks are increasingly utilized to capture the complexity of entities and their intricate interrelations. Recent research has leveraged the graph-structured information propagation paradigm to enhance user and item embeddings, employing a variety of neighborhood aggregation techniques [14, 18, 44, 60]. For instance,

LightGCN [10] stands out as a leading graph learning-based recommendation model, celebrated for its streamlined architecture. Drawing inspiration from SGC [50], it simplifies the traditional graph convolutional neural networks by omitting transformation matrices and non-linear activation functions, focusing instead on the essential elements of graph convolution. There are also some studies combining hypergraph learning with recommender systems, including [4, 22, 47, 54], which use hypergraph to model the short-term user preference for next-item recommendations. Nonetheless, challenges such as sparsity and noise within the graph can significantly hinder effective information transfer among nodes, potentially confusing the model by prioritizing irrelevant job preferences [8, 58]. To counteract these issues, our framework incorporates a wavelet denoising filter, specifically designed to cleanse the data and ensure the accurate extraction of meaningful job preferences, thereby enhancing the overall recommendation process.

3 PRELIMINARIES

In this paper, we adopt the BISTRO framework to solve user preference drift during the job-seeking process in top-k job recommendations for users. Specifically, as shown in the above statistics, we believe a user $u \in \mathcal{U}$ would continue to refine her/his resume along with her/his job preference drift. Thus, we first segment the user interaction sequence based on the timestamps of resume refinement under the assumption that the job preferences of users remain relatively stable within a given session.

Definition 1. ***Job Preference Drift***. *It refers to the phenomenon in which users change their job preferences, which could be predominantly observed through whether the user modifies the resume rather than modeling the short-term job-seeking behaviors where user interests tend to remain stable.*

Definition 2. ***Interaction Noise***. *It stands for job-seeking behaviors that are inconsistent with user preferences, i.e., accidental clicks while a user browses jobs.*

Definition 3. ***Session-based User-Job Interactions***. *For every user u, we need to segment the sequence of the job interactions into multiple sessions $\mathcal{S}^u = \{s_1^u, s_2^u, \cdots\}$ based on whether the resume of the user has been changed, in order to obtain more accurate job preference in a specific period.*

Based on the definition above, we formulate the problem of the session-based job recommendation as follows:

Problem 1. *Given the historical user resumes D_t^u, job requirements D_t^v and interaction sessions $\mathcal{S}^u$, the objective of the job recommender system is to identify and rank a list of the top-k job vacancies that would likely appeal to a user u during a session s^u.*

4 METHODOLOGY

In this section, we detail the architecture of the proposed framework, BISTRO, illustrated in Figure 2. The framework comprises three primary modules: 1) a coarse-grained semantic clustering module, 2) a fine-grained job preference extraction module, and 3) a personalized top-k recommendation module.

Initially, the coarse-grained semantic clustering module incorporates a feature clustering approach with a probabilistic latent semantic analysis method, which facilitates the identification of broad user or job categories. The probabilistic latent semantic analysis method could efficiently summarize topics of the resume content and job requirements, and those topics could guide clustering directions. Subsequent to this, the fine-grained job preference extraction module constructs a multi-granular interaction hypergraph to deal with the data drift issue and then designs an adaptive wavelet learning algorithm for noise-robust preference extraction. In the hypergraph, we define two types of hyperedges, reflecting the intra-session and inter-session relationships, to introduce more information to the graph. Moreover, the wavelet filter in hypergraph wavelet learning is designed to detect noise in the spectral domain and further adaptively mitigate the effects of data noise. The final stage combines a recurrent neural network to discern personalized job preferences from short-term sequential interactions to generate top-k job recommendation results.

4.1 Coarse-grained Semantic Clustering

The coarse-grained semantic clustering serves as the foundational component of our framework, setting the stage for nuanced preference feature extraction. By tackling the challenge of aligning diverse and dynamic job preferences with suitable opportunities, this module highlights the core motivation behind the intricate process of facilitating effective employment matching. It utilizes semantic insights from resumes and job descriptions to broadly match job seekers with appropriate vacancies, identifying potential fits based on semantic themes related to job preference and recruitment requirements.

In our model, the conditional probability between the document content $d \in \boldsymbol{D}$ and words $w \in d$ is captured through a latent embedding z ($z = \text{Linear}(w)$ or $\text{Linear}(d)$), representing a class or topic. The model parameters, $\boldsymbol{P}(w|z)$ and $\boldsymbol{P}(z|d)$, allow for the possibility that words may associate with multiple classes and documents that may cover various topics. We assume that the distribution of words given a class, $\boldsymbol{P}(w|z)$ is conditionally independent of the document, implying $\boldsymbol{P}(w|z,d) = \boldsymbol{P}(w|z)$. Thus, the joint probability of a document d and a word w is represented as:

$$\boldsymbol{P}(w,d) = \boldsymbol{P}(d) \sum_z \boldsymbol{P}(w|z)\boldsymbol{P}(z|d). \tag{1}$$

To estimate the parameters $\boldsymbol{P}(w|z)$ and $\boldsymbol{P}(z|d)$, the Expectation-Maximization (EM) algorithm iteratively maximizes the log-likelihood function $\boldsymbol{L}$ over a training corpus $\boldsymbol{D}$:

$$\boldsymbol{L} = \sum_{d \in D} \sum_{w \in d} f(d,w) \log \boldsymbol{P}(d,w), \tag{2}$$

where $f(d,w)$ is the frequency of word w in document d. The EM process alternates between 1) the E-step, estimating the probability $\boldsymbol{P}(z|w,d)$ as:

$$\boldsymbol{P}(z|w,d) = \frac{\boldsymbol{P}(w|z)\boldsymbol{P}(z|d)}{\sum_{z'} \boldsymbol{P}(w|z')\boldsymbol{P}(z'|d)}, \tag{3}$$

and 2) the M-step, recalculating $\boldsymbol{P}(w|z)$ and $\boldsymbol{P}(z|d)$ to maximize $\boldsymbol{L}$:

$$\boldsymbol{P}(w|z) = \frac{\sum_d f(d,w)\boldsymbol{P}(z|w,d)}{\sum_{w'} \sum_d f(d,w')\boldsymbol{P}(z|w',d)}, \tag{4}$$

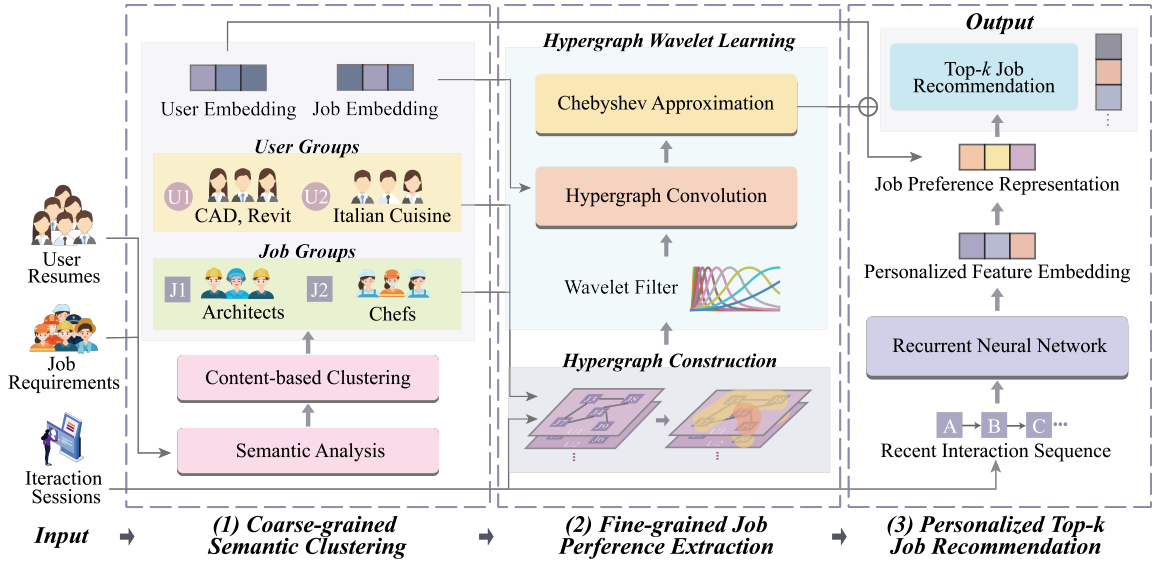


Figure 2: The framework overview of BISTRO. The framework is divided into three parts: coarse-grained semantic clustering, fine-grained job preference extraction, and personalized top-k job recommendation.

$$P(z|d) = \frac{\sum_w f(d,w)P(z|w,d)}{\sum_{z'}\sum_w f(d,w)P(z'|w,d)}. \tag{5}$$

Following training, the "folding-in" process applies the estimated $P(w|z)$ to test documents $d' \in D$, recalculating $P(z|d')$ while keeping $P(w|z)$ constant. Typically, only a few iterations of the EM algorithm are required for this process.

After semantic analysis, we combine the normalized document and word latent embeddings $\{z_1, z_2, \cdots\}$ with other normalized attributes $attr$ such as age to achieve clustering by K-Means algorithm for the whole data $C := C_{\mathcal{U}}$ or $C_{\mathcal{V}}$ due to the fact of its high efficiency, as shown in Equation (6).

$$\arg\min_{C} \sum_{i}^{K} \sum_{x \in C_i} ||x - \mu_i||_2^2, \tag{6}$$

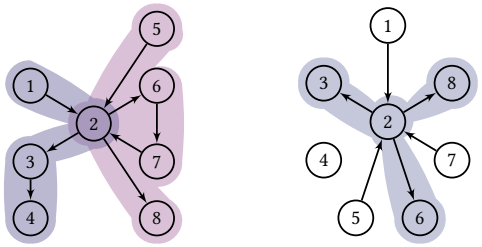
where μ_i is the mean of all data in C_i, $x = \text{concat}(z_1, z_2, \cdots, attr)$, K is the number of clusters.

4.2 Fine-grained Job Preference Extraction

Short-term interactions at a session level always encompass issues of user preference drift and noisy interactions, with each influencing the other. To tackle the dual challenges above, the fine-grained job preference extraction module utilizes a novel adaptive hypergraph wavelet learning method in a unified approach.

Initially, employing a standard graph structure to map user-job interactions often results in a proliferation of isolated vertices and edges, adversely impacting the efficacy of graph learning-based job preference extraction. In response, this paper introduces a hypergraph structure, denoted as $\mathcal{G}^{C_u} = (\mathcal{V}^{C_u}, \mathcal{E}^{C_u})$, which utilizes two specialized types of hyperedges to enhance the data with additional insights. The hypergraph for each user group C_u encompasses n job group nodes, alongside corresponding features (graph signals) $X_t^{C_u}$. The hyperedge, defined as $e = \text{link}(v_a, v_b, \cdots) \in \mathcal{E}^u$, constitutes a subset of the vertex set $\mathcal{V}$, capturing complex, high-order relationships within the graph. For illustration, consider two sessions: $s_1^{C_u} = \{v_1, v_2, v_3, v_4\}$, $s_2^{C_u} = \{v_5, v_2, v_6, v_7, v_2, v_8\}$. The introduction of two distinct hyperedge types significantly augments the data connectivity within the user-job graph, as depicted in Figure 3.

Session Hyperedges $\mathcal{E}_s^{C_u}$. The intra-session relationship is demonstrated as one of the critical factors to session-based recommendation [11]. For each user group, we link all jobs in each session to enhance the connectivity of these jobs. As for the job v_2 in Figure 3(a), we connect the session jobs $\{v_1, v_3, v_4\}$ and $\{v_5, v_6, v_7, v_8\}$ that include it with a hyperedge, respectively. It reveals the high-order correlation of jobs facilitating the interaction on v_2.



(a) Session hyperedges: The pink and grey areas are two hyperedges.

(b) The transition hyperedge: The grey area is the hyperedge for node 2.

Figure 3: Two types of hyperedges.

Transition Hyperedges $\mathcal{E}_v$. Since the first type of hyperedges cannot model the chronological order of user-job interactions, we utilize job transition hyperedges to address this issue. For example, we connect the outcoming jobs $\{v_3, v_6, v_8\}$ for job v_2 as a hyperedge in Figure 3(b), which also implies the inter-session relationship

in those two sessions . These hyperedges also tackle the issue of preference drift among sessions.

To sufficiently filter the noise while capturing the job preferences, we design a spectral-based hypergraph wavelet convolutional neural network with a graph filter in the spectral domain for graph convolutional operation.

The graph convolution of the general graph signal $\boldsymbol{x}$ with a filter $\boldsymbol{g} \in \mathbb{R}^n$ is defined as:

$$\begin{aligned} \boldsymbol{x} *_G \boldsymbol{g} &= \mathcal{F}^{-1}\left(\mathcal{F}(\boldsymbol{x}) \odot \mathcal{F}(\boldsymbol{g})\right) \\ &= \boldsymbol{U}\left(\boldsymbol{U}^{\mathrm{T}}\boldsymbol{x} \odot \boldsymbol{U}^{\mathrm{T}}\boldsymbol{g}\right), \end{aligned} \tag{7}$$

where $*_G$ stands for the convolution operator, $\odot$ denotes the element-wise product. If we denote a filter as $\boldsymbol{g}_\theta = \mathrm{diag}\left(\boldsymbol{U}^{\mathrm{T}}\boldsymbol{g}\right)$, then the spectral graph convolution can be simplified as:

$$\boldsymbol{x} *_G \boldsymbol{g}_\theta = \boldsymbol{U}\boldsymbol{g}_\theta\boldsymbol{U}^{\mathrm{T}}\boldsymbol{x}. \tag{8}$$

Almost all spectral-based graph convolutional neural networks follow the definition above, and the key difference lies in the choice of the filter. For example, Bruna *et al.* [3] take a filter $\boldsymbol{g}_\theta = \Theta_{i,j}^{(k)}$ to be a set of learnable parameters and consider graph signals having multiple channels. They define the graph convolutional layer as:

$$\boldsymbol{X}_{:,j}^{(k)} = \sigma\left(\sum_{i=1}^{d_{k-1}} \boldsymbol{U}\Theta_{i,j}^{(k)}\boldsymbol{U}^{\mathrm{T}}\boldsymbol{X}_{:,i}^{(k-1)}\right), j = 1, 2, \cdots, d_k, \tag{9}$$

where k is the layer index, $\boldsymbol{X}^{(k-1)} \in \mathbb{R}^{n\times d_{k-1}}$ is the input graph signal, $\boldsymbol{X}_{:,i}^{(0)} = \boldsymbol{x} \in \mathbb{R}^{n\times 1}$, d_{k-1} is the number of input channels and d_k is the number of output channels, $\Theta_{i,j}^{(k)}$ is a diagonal matrix filled with learnable parameters.

From Equation (9), the Laplacian eigenvectors of the hypergraph need to be precomputed to realize the mapping of hypergraph features between the vertex and spectral domains. Note that the Laplacian matrix of a hypergraph is defined as follows: $\mathcal{L} := \boldsymbol{D}_v - \boldsymbol{A} = \boldsymbol{D}_v - \boldsymbol{H}\boldsymbol{W}\boldsymbol{D}_e^{-1}\boldsymbol{H}^{\top}$, where $\boldsymbol{A}$ is the adjacency matrix of the hypergraph, $\boldsymbol{H}$ is the node-edge relationship matrix, $\boldsymbol{D}_v$ and $\boldsymbol{D}_e$ are degree matrix of nodes and hyperedges separately. Then, the eigenvectors can be obtained by eigendecomposition methods, as shown in Equation (10).

$$\boldsymbol{D}_v - \boldsymbol{H}\boldsymbol{W}\boldsymbol{D}_e^{-1}\boldsymbol{H}^{\top} = \boldsymbol{U}\Lambda\boldsymbol{U}^{\top}, \tag{10}$$

where Λ is a diagonal matrix of Laplacian eigenvalues.

In addition, the filter $\boldsymbol{g}_\theta$ utilized in Equation (9) is usually a set of learnable hyperparameters but has problems such as convergence difficulty. Therefore, we utilize the wavelet kernel to finely define a series of filters that filter in-session noise adaptively for different user groups. Similar to the hypergraph Fourier transform, the hypergraph wavelet transform projects the hypergraph signal from the vertex domain into the spectral domain. Graph wavelet transform employs a set of wavelets as bases, defined as $\psi_\kappa = \mathrm{concat}(\psi_1^\kappa, \psi_2^\kappa, \cdots, \psi_n^\kappa) \in \mathbb{R}^{n\times n}$, where each wavelet $\psi_i^\kappa \in \mathbb{R}^{1\times n}$ corresponds to a signal on graph diffused away from node i and κ is a scaling parameter, which is adapted to spectrum bounds. Mathematically, ψ_κ and ψ_κ^{-1} can be written as

$$\psi_\kappa = \boldsymbol{U}\boldsymbol{G}^{\boldsymbol{\kappa}}(\Lambda)\boldsymbol{U}^{\mathrm{T}}, \quad \psi_\kappa^{-1} = \boldsymbol{U}\boldsymbol{G}^{\boldsymbol{\kappa}}(\Lambda')\boldsymbol{U}^{\mathrm{T}}, \tag{11}$$

where $G^\kappa(\Lambda) = \mathrm{diag}[g(\kappa\lambda_1), \cdots, g(\kappa_{\lambda_n})]$ and $G^\kappa(\Lambda') = \mathrm{diag}[g^{-1}(\kappa_{\lambda_1}), \cdots, g^{-1}(\kappa_{\lambda_n})]$ are scaling matrix.

To reduce the computational overhead incurred by the inverse operation, we choose the heat kernel $g(x) := e^{-x}$ as the wavelet mother kernel in this paper, and then $g^{-1}(x) = e^{-(-x)} = g(-x)$.

However, eigen-decomposition is known to have extremely high computational overhead. In order to avoid the computational cost caused by solving the eigen-decomposition of the Laplacian matrix, we use Chebyshev polynomials to approximate the convolutional operator $\boldsymbol{x} *_G \boldsymbol{g}_\theta \approx \sum_{i=1}^{p} c_i T_i\left(\tilde{\Lambda}\right)$, where $\tilde{\Lambda} = \frac{2}{g_{\max}}\Lambda - \boldsymbol{I}_n$ and $\boldsymbol{\tau} = [\tau_1, \cdots, \tau_p]$ is a vector of Chebyshev coefficients. Note that $T_i(\tilde{\psi}_\kappa) = \boldsymbol{U}T_i(\tilde{\Lambda})\boldsymbol{U}^{\mathrm{T}}$, $\tilde{\psi}_s = \frac{2}{g_{\max}}\psi_s - \boldsymbol{I}_n$, we have:

$$\psi_\kappa \approx P_p(\psi_\kappa) = \sum_{i=1}^{p} \tau_i T_i(\tilde{\psi}_\kappa). \tag{12}$$

Similarly,

$$\psi_\kappa^{-1} \approx P_p(\psi_\kappa^{-1}) = \sum_{i=0}^{p} \tau_i' T_i(\tilde{\psi}_\kappa^{-1}), \tag{13}$$

where $\tilde{\psi}_\kappa^{-1} = \frac{2}{(g^{-1})_{\max}}\psi_\kappa^{-1} - \boldsymbol{I}_n$ and τ_i' is a vector of Chebyshev coefficients.

Therefore, the l-th hypergraph wavelet convolutional network for job preference feature extraction can be derived as follows:

$$\boldsymbol{X}_{:,j}^{l+1} = \sigma\left(\sum_{i=1}^{d_l} w_{i,j}|\mathcal{V}|^{-1}\sum_{\kappa=1}^{|\mathcal{V}|} P_p(\psi_\kappa)(\tilde{\mathcal{L}})\boldsymbol{g}_{\kappa,\theta_{(j,i)}}^{l} P_p(\psi_\kappa^{-1})(\tilde{\mathcal{L}})\boldsymbol{X}_{:,i}^{l}\right). \tag{14}$$

Generally, many studies assume the Chebyshev coefficients $\{\tau_1, \tau_2, \cdots\}$ in Equation (12) and (13) to be a set of random variables that can be trained by a Multi-Layer Perceptron (MLP). In this paper, we use Boyd's theory [2] to calculate those coefficients directly. From Boyd's theory, the Chebyshev coefficients can be expressed by:

$$\tau_j = \frac{2}{n+1}\sum_{\kappa=0}^{n} f\left(y_\kappa\right) T_j\left(y_\kappa\right), \tag{15}$$

where $y_\kappa := cos\left(\frac{(2\kappa+1)\pi}{2n+2}\right), \kappa \in [n]$ is the interpolant of degree n in the Chebyshev points of the first kind, and $f(\cdot)$ is the Chebyshev series for the function f that absolutely continuous on $[-1, 1]$, which is defined as [40, 43]:

$$f(x) = \sum_{j=0}^{\infty} b_j T_j(x), \quad b_j = \frac{2}{\pi}\int_{-1}^{1} \frac{f(x)T_j(x)}{\sqrt{1-x^2}}dx, \tag{16}$$

where the prime denotes a sum whose first term is halved and $T_j(x) = \cos\left(j\cos^{-1}x\right)$ is the Chebyshev polynomial of degree j.

The Chebyshev coefficient τ_j in Equation (15) can be computed by Fast Fourier Transform (FFT), which significantly improves computational efficiency compared to eigen-decomposition methods.

4.3 Personalized Top-k Job Recommendation

Adapting job recommendations to reflect the job preference of the current user is the final key challenge faced by job recommendation systems, which could directly affect the recommendation

performance. To address this difficulty, BISTRO employs a recurrent neural network to align recommendations with previously extracted job preferences precisely. This approach underscores our dedication to providing timely and relevant job matches, ensuring high accuracy in meeting user needs.

To refine the analysis of a user's personalized features, we consider the last T jobs with which a user has recently interacted, along with their embeddings $\boldsymbol{Emb}_t^{\text{job}}, t \in [T]$. A recurrent neural network is employed to generate the personalized feature $\boldsymbol{Y}_t$:

$$\begin{aligned} \boldsymbol{Y}_t &= \sigma(\boldsymbol{W}_a \cdot \boldsymbol{O}_t), \\ \boldsymbol{O}_t &= \sigma(\boldsymbol{W}_b \cdot \boldsymbol{Emb}_t^{\text{job}} + \boldsymbol{W}_c \cdot \boldsymbol{O}_{t-1}), \end{aligned} \tag{17}$$

where $\boldsymbol{O}_0 = \boldsymbol{0}$, σ is the activation function, $\boldsymbol{W}_a$, $\boldsymbol{W}_b$ and $\boldsymbol{W}_c$ are learnable parameters.

Then, this personalized feature enhances the job preference $\boldsymbol{X}^L$ calculated by Equation (14) for a specific user of a certain feature by the following weighted formula:

$$\boldsymbol{Y} = \text{Sigmod}\left(\text{Linear}\left(\text{concat}(\boldsymbol{Y}_T, \boldsymbol{X}^L)\right)\right). \tag{18}$$

BISTRO gives the top-k job recommendation by minimizing the loss function below:

$$Loss = \frac{1}{k}\sum_{i=1}^{k}\sum_{q=1}^{M} \chi_{iq} \log(y_{iq}), \tag{19}$$

where $\chi_{iq} \in [0, 1]$ is the ground truth, y_{iq} is the value on i-th row and q-th column in $\boldsymbol{Y}$, standing for the probability that the i-th predicted job belongs to job label q, M is the number of job positions.

5 EXPERIMENTS

In this section, we first describe the datasets used in this paper. Then, we introduce the experimental settings and compare BISTRO with representative baselines. We further present some case studies on job recommendations (*Appendix C.3*). The experiments are mainly designed to answer the research questions as follows:

- **RQ1**: Can our BISTRO recommend suitable jobs for users?
- **RQ2**: Does the clustering module effectively accommodate new jobs or users who have just revised their resumes?
- **RQ3**: How does the specially designed hypergraph wavelet learning method deal with preference drift and noise issues?
- **RQ4**: How do different settings influence the model performance?

5.1 Dataset

The datasets come from the real-world online recruitment markets of multiple cities (Shenzhen, Shanghai, and Beijing). We utilize the user-job interaction (browse, click, chat, and so on) logs, user resumes, and job requirements data on this platform from July 1, 2023 to January 31, 2024 (215 days in total). To protect the privacy of users and platform operators, all sensitive information related to users is hashed or removed, and we only keep those data in the information technology industry. The detailed statistical information of our datasets is summarized in Table 1.

Table 1: Statistics of the datasets.

	Users	Interactions	Jobs
Shenzhen	9,586	9,504,285	1,804,402
Shanghai	10,233	8,894,726	1,707,248
Beijing	15,915	13,467,252	2,012,972

5.2 Experimental Settings

Baselines We compare BISTRO with baselines from different types of recommendation methods, including conventional methods: BasicMF [20], ItemKNN [46], PureSVD [7], and SLIM [34], DAE [53], MultVAE [25], EASE [41]; Graph neural networks-based methods: SLRec [59] and SGL [51], P3a [6], RP3b [36], NGCF [48], LightGCN [10], GCCF [5], NCL [28], DirectAU [44], HG-GNN [35], A-PGNN [61], AdaGCL [18], and MvDGAE [64]; Sequential recommendation methods: STAMP [31], GRU4Rec [11], BERT4Rec [42], CL4Rec [55], CoScRec [32], and TiCoSeRec [9]. More details about these baselines are shown in *Appendix C.1*. For ablation studies, we compare the variants of BISTRO to verify the effectiveness of each component.

Evaluation metrics In this paper, two metrics commonly used in recommendation algorithms are used as evaluation metrics: hit ratio and mean reciprocal rank, and the definitions of these metrics are demonstrated as follows:

- **Hit Ratio (HR)**: It measures the proportion of successful recommended jobs out of all the recommendations made. Later, we use **H@**k to denote the value of HR when the model makes top-k recommendations.
- **Mean Reciprocal Rank (MRR)**: It is a statistical measure that focuses explicitly on the rank of the first relevant item in the list of recommendations to show the effectiveness of a recommendation method, and in this paper we use the symbol **M@**k to present this metric for simplicity.

Implementation Details We experiment with a Spark cluster for preprocessing the data and A800 GPU servers to train and infer the proposed model. It has three parts: coarse-grained semantic clustering, fine-grained job preference extraction, and personalized top-k job recommendation. 1) Coarse-grained semantic clustering: we set the ratio of the number of groups to the number of users to about 1:1000 and the ratio of the number of groups to the number of job numbers to about 1:500. User professional skills and their inherent characteristics, such as work experience, are used as user clustering characteristics. Similarly, we extract job requirements as clustering features for jobs. 2) Fine-grained job preference extraction: the sparse matrix is used to represent the structure of hypergraphs efficiently. In addition, the order of Chebyshev approximation is $p = 3$, the total degree of interpolants is $n = 50$, and the number of hypergraph convolutional layers is $L = 1$. We set the number of hidden dimensions of this network to $d_l = 128$. 4) Personalized top-k job recommendation: the number of hidden dimensions of the recurrent neural network is set to be the same as the one in hypergraph convolutional layers: $d_{\text{rnn}} = 128$. We set $k = 10$ for baseline experiments, and more experimental results under different settings of k can be found in *Appendix C.2*. As for each dataset, we split the training and validation data at a ratio

of 4:1, and we randomly sample 20% data of the whole dataset for testing. In addition, Adam is used as the optimizer for all models, and we use the default parameter for it, *i.e.,* $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, $lr = 10^{-3}$.

5.3 Overall Performance *(RQ1)*

The performance of all the baselines in three datasets is shown in Table 2, in terms of the two metrics, *i.e.,* H@10 and M@10. The performance of all methods is the average of the last 100 epochs in a total of 1000 epochs. It can be observed:

Table 2: Experimental results of different baselines.

	Shenzhen		Shanghai		Beijing	
	H@10	**M@10**	**H@10**	**M@10**	**H@10**	**M@10**
BasicMF	0.2791	0.2837	0.3528	0.2773	0.3181	0.3004
ItemKNN	0.2479	0.2969	0.3278	0.2793	0.3224	0.2931
PureSVD	0.4422	0.4205	0.4658	0.4391	0.4666	0.3827
SLIM	0.045	0.0443	0.0463	0.047	0.0539	0.04
DAE	0.1867	0.2024	0.1988	0.2125	0.2013	0.2024
MultVAE	0.4249	0.4149	0.4516	0.3499	0.5229	0.348
EASE	0.2891	0.3192	0.2683	0.2989	0.29	0.2818
SLRec	0.2767	0.3796	0.3636	0.2958	0.3968	0.3526
SGL	0.4089	0.3618	0.4375	0.397	0.4128	0.3618
P3a	0.3185	0.3748	0.3177	0.3355	0.3728	0.3489
RP3b	0.2917	0.3017	0.3362	0.2929	0.3304	0.2919
NGCF	0.3557	0.2859	0.398	0.2924	0.3851	0.2807
LightGCN	0.3785	0.3647	0.4423	0.3453	0.4232	0.3911
GCCF	0.3756	0.346	0.3901	0.2993	0.3896	0.3283
NCL	0.3424	0.3917	0.3476	0.3472	0.4111	0.3557
DirectAU	0.3725	0.37	0.4173	0.3759	0.3943	0.3643
HG-GNN	0.3131	0.1936	0.3637	0.2826	0.4204	0.3430
A-PGNN	0.3372	0.2069	0.3876	0.3019	0.4485	0.3657
AdaGCL	0.3867	0.4417	0.4044	0.3814	0.4516	0.3867
MvDGAE	0.5028	0.5258	0.5316	0.4812	0.4677	0.4845
STAMP	0.2754	0.2924	0.3031	0.2756	0.3073	0.2745
GRU4Rec	0.3247	0.3449	0.3576	0.3251	0.3623	0.3237
BERT4Rec	0.3395	0.3607	0.3738	0.3399	0.3787	0.3385
CL4Rec	0.3882	0.4124	0.4276	0.3887	0.4332	0.3871
CoScRec	0.4096	0.4352	0.4512	0.41	0.457	0.4084
TiCoSeRec	0.4397	0.4671	0.4842	0.4402	0.5324	0.4925
Ours	**0.5598**	**0.5467**	**0.6166**	**0.5152**	**0.6247**	**0.5131**
Improves	10.19%	3.82%	13.78%	6.61%	14.7%	4.01%

Bold indicates the statistically significant improvements (*i.e.,* two-sided t-test with $p < 0.05$) over the best baseline (underlined). For all metrics: the higher, the better.

- We can see that most conventional methods have poor performance, *i.e.,* BasicMF, ItemKNN, SLIM, DAE, and EASE. SLIM deploys only a linear function to model user-job interactions, which limits the ability to generalize the model. BasicMF, ItemKNN, DAE, and EASE cannot provide fine-grained modeling for user/job features. PureSVD and MultVAE offer significant enhancements in performance over other techniques, yet they require extensive computational resources. Despite their advantages, they fall short of accurately capturing the dynamics of drifted interactions.

Table 3: Results of online experiments.

	Day 1		Day 2		Day 3		Day 4	
	C	**S**	**C**	**S**	**C**	**S**	**C**	**S**
MvDGAE	0.67	0.46	0.79	0.49	0.76	0.53	0.69	0.47
TiCoSeRec	0.45	0.23	0.66	0.41	0.73	0.72	0.70	0.78
CoScRec	0.42	0.30	0.43	0.28	0.46	0.32	0.55	0.46
BISTRO	**0.82**	**0.71**	**1.00**	**0.96**	**0.98**	**1.00**	**0.97**	**0.95**

"**C**" indicates the rate of having a chat about the recommended jobs, "**S**" stands for the rate of onboarding to the recommended jobs.
* Please note that all results have been ***normalized*** to safeguard the company's trade secrets.
For all metrics: the higher, the better.

- Compared to GNN-based methods, BISTRO allows for the extraction of fine-grained user representations from user resumes and their use in preference analysis. Among these baselines, MvDGAE achieves the best performance. This is because it also uses noise reduction representation learning based on multiview graphs. However, it lacks content modeling of user resumes and job requirements, resulting in lower results than our framework.
- Sequential models can effectively learn the relationship among user-job interactions over time, but such interactions can easily be negatively impacted by spontaneous user preference drift, which would be directly reflected in interaction records. Thus, the sequence-only models do not apply to the job recommendation scenario, and their performance is justifiably worse than that of our framework.

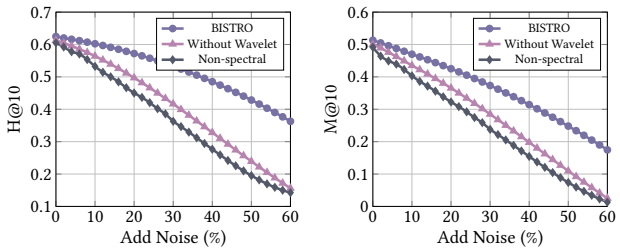
Furthermore, we extended our evaluation by deploying our model from the offline experiments to an online recruitment platform for a half-week online experiment, as shown in Table 3. In online experiments, the performance is measured on a daily basis, and we randomly select a unique 1% of active users and push the results for each model directly at the re-rank stage. It demonstrates our proposed method's superior performance and exceptional robustness compared to other baseline models.

5.4 Ablation Study *(RQ2, RQ3)*

The influence of clustering As mentioned previously, the clustering module can effectively achieve semantic matching at a coarse-grained level. Therefore, we partition the dataset into four subsets that do not overlap each other to train the model to achieve the following four tasks: 1) existing jobs for existing users, 2) existing jobs for users who have just revised resumes, 3) new jobs for existing users, and 4) new jobs for users who have just revised resumes. The comparison results are shown in Table 4.

Table 4: Results under different tasks of the dataset.

	Ours		Non-clustering	
	H@10	**M@10**	**H@10**	**M@10**
Task 1	0.7241	0.5947	0.7119	0.6012
Task 2	0.6889	0.5659	0.2215	0.1819
Task 3	0.6557	0.5385	0.2018	0.1658
Task 4	0.6247	0.5131	0.0855	0.0702



(a) The HR metric (b) The MRR metric

Figure 4: Results of model performance in relation to the proportion of noise in data.

Compared to the framework without a clustering module, BISTRO is better adapted to semantic matching: neither the HR nor the MRR suffers from a significant drop, which shows the efficiency of the clustering module.

The effect of hyperedges The construction of hyperedges in BISTRO can increase the density of the interaction graph, which could add more structural information to address the preference drift issue. To verify this idea, we use the following formula to compute the density of an undirected graph: $\textbf{Density} = \frac{2|\mathcal{E}|}{|\mathcal{V}|\times(|\mathcal{V}|-1)}$. As shown in Table 5, the density of the graph is doubled when we add all types of hyperedges and the experimental results have also been improved due to the optimization of the structure in the graph.

Table 5: Results under different graph constructions.

	Density	H@10	M@10
Without hyperedges	1.177%	0.4109	0.3375
Only session hyperedges	2.811%	0.4904	0.4028
Only transition hyperedges	2.793%	0.4927	0.4046
Both types of hyperedges	3.581%	0.6247	0.5131

The validity of hypergraph wavelet filter In BISTRO, we design a novel hypergraph wavelet learning method. In this learning method, a wavelet filter is deployed for data denoising as well as fine-grained job preference feature extraction. As shown in Figure 4, the curves illustrate the results of three models, which have different filtering settings, under different percentages of noise in the data. We can also visualize from it that our method, BISTRO, has the smoothest decrease in model performance as the proportion of noise in the data increases.

To vividly show the denoising capability of the proposed hypergraph wavelet filter, we randomly select a user who is active in a week, filter the 50 most recent interactions from three job categories, and construct an interaction graph. In this graph, each node represents a job the user has engaged with, interconnected by grey dotted lines, while the interaction sequence of the user is depicted with grey edges. On this basis, we introduce noisy jobs (marked with orange crosses) and their corresponding interactions (denoted by orange edges and dotted lines) to mimic the effect of a user accidentally clicking on unrelated job types. Given that each

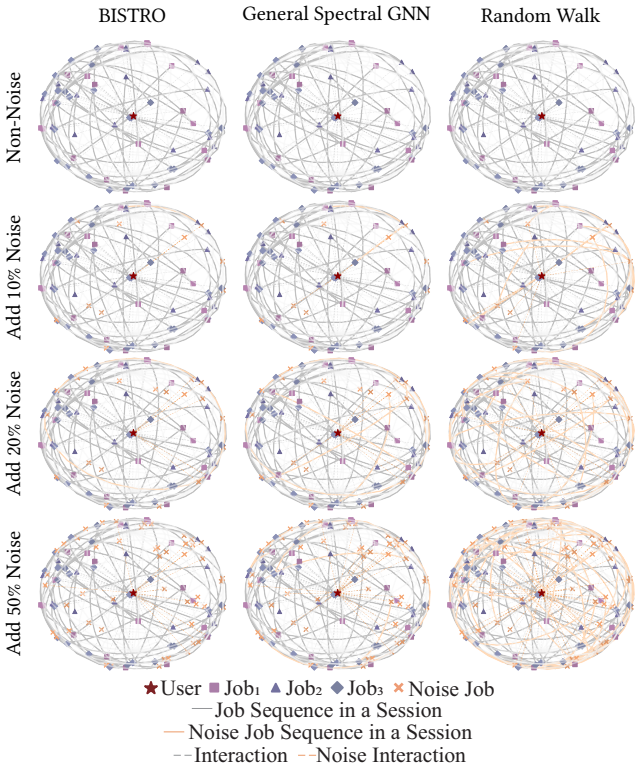
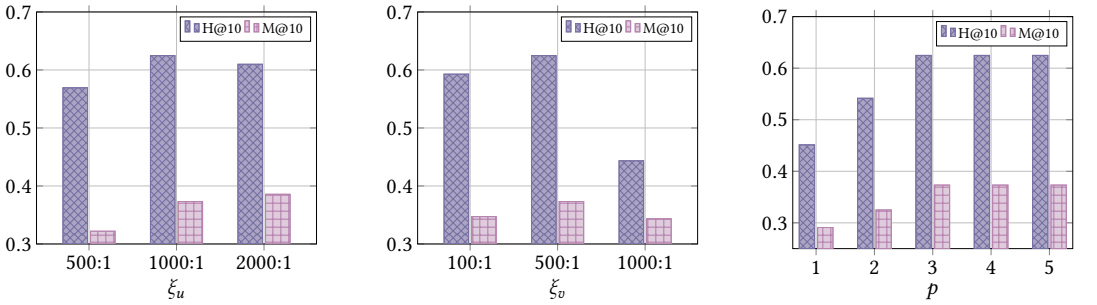


Figure 5: Results of denoising performance.

model generates job preference representations for diverse jobs, we visualize the connections between the user and jobs, as well as the relationships among jobs themselves, as shown in Figure 5. We eliminate edges whose cosine similarity between job representation pairs fell below a uniform threshold and remove links between isolated jobs and the user. Consequently, a graph with more orange lines indicates lower model performance. Notably, as data noise levels escalated, the comparative models demonstrated diminished noise filtering effectiveness relative to our proposed approach. Specifically, the random walk-based method significantly underperformed compared to the spectral GCN method, primarily due to the ability of spectral graph neural networks to filter out irrelevant interaction features. Furthermore, our approach employs a wavelet kernel to create a set of sub-filters, adeptly denoising by dynamically selecting appropriate filters for the user's evolving characteristics.

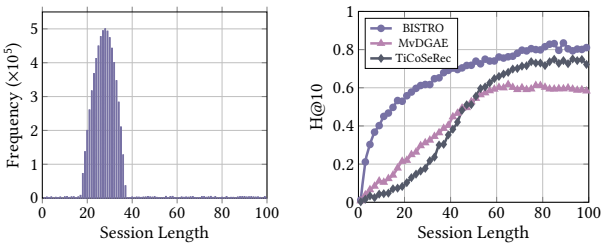
5.5 Parametric Study *(RQ4)*

The size of user (job) groups The size of user and job groups are two hyperparameters that need to be predefined. Therefore, we choose 500:1, 1000:1, and 2000:1 as the ratios of the total number of users and the number of user groups ξ_u, and 100:1, 500:1, 1000:1 as the ratios of the total number of jobs and the number of job groups ξ_v for our experiments respectively, as shown in Figure 6(a) and 6(b). We can easily observe that our model achieves best when ξ_u =1000:1 and ξ_v =500:1.



(a) Different user group sizes (b) Different job group sizes (c) Different orders of Chebyshev approximation

Figure 6: Results of different hyperparameters.



(a) Statistics of different session length (b) Model performance under different session length

Figure 7: Results under different session lengths.

The order of Chevbyshev approximation The order of Chevbyshev approximation greatly impacts the performance of hypergraph wavelet neural networks. To find the best order, we test our model with $p = 1, 2, 3, 4$ and 5, and the results are shown in Figure 6(c). We can see that the performance of the model remains constant when p is greater than or equal to 3. Notice that as p increases, the computational overhead of the model will also increase, so we choose $p = 3$ as the hyperparameter of our model.

The average length of a session The length of the session is another hyperparameter that affects the performance of the model. In Figure 7(a), we can see that the average length of each session is 19-37 on average, and such short behavioral sequences in job recommendations (In job recommendations, the average number of interactions for users to find a suitable job is more than 80 times) are easily interfered with by noisy interactions. Therefore, we further compared our proposed framework with the top-2 baselines under different session length, as illustrated in Figure 7(b). It can be seen that when the session length is relatively short, noise has a huge negative impact on the accuracy of all models. However, as the session length decreases, our framework is more robust than the other two methods and can better resist noise interference.

6 CONCLUSION

This study introduces BISTRO, an innovative framework designed to navigate the challenges of job preference drift and the subsequent data noise. The framework is structured around three modules: a coarse-grained semantic clustering module, a fine-grained job preference extraction module, and a personalized top-k job recommendation module. Specifically, a hypergraph is constructed to deal with the preference drift issue and a novel hypergraph wavelet learning method is proposed to filter the noise in interactions when extracting job preferences. The effectiveness and clarity of BISTRO are validated through experiments conducted with both offline and online environments. Looking ahead, we aim to continue refining BISTRO to enhance its applicability in broader contexts, particularly in scenarios characterized by anomalous data.

REFERENCES

[1] Shuqing Bian, Wayne Xin Zhao, Yang Song, Tao Zhang, and Ji-Rong Wen. 2019. Domain adaptation for person-job fit with transferable deep global match network. In *Proc. of EMNLP*.
[2] John P Boyd. 2001. *Chebyshev and Fourier spectral methods*. Courier Corporation.
[3] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral Networks and Locally Connected Networks on Graphs. In *Proc. of ICLR*.
[4] Jiajun Bu, Shulong Tan, Chun Chen, Can Wang, Hao Wu, Lijun Zhang, and Xiaofei He. 2010. Music recommendation by unified hypergraph: combining social media information and music content. In *Proc. of ACM MM*.
[5] Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. 2020. Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach. In *Proc. of AAAI*.
[6] Colin Cooper, Sang Hyuk Lee, Tomasz Radzik, and Yiannis Siantos. 2014. Random walks in recommender systems: exact computation and simulations. In *Proc. of WWW*.
[7] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*.
[8] Enyan Dai, Charu Aggarwal, and Suhang Wang. 2021. Nrgnn: Learning a label noise resistant graph neural network on sparsely and noisily labeled graphs. In *Proc. of KDD*.
[9] Yizhou Dang, Enneng Yang, Guibing Guo, Linying Jiang, Xingwei Wang, Xiaoxiao Xu, Qinghui Sun, and Hong Liu. 2023. Uniform Sequence Better: Time Interval Aware Data Augmentation for Sequential Recommendation. In *Proc. of AAAI*.
[10] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proc. of SIGIR*.
[11] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. In *Proc. of ICLR*.
[12] Guangneng Hu, Yu Zhang, and Qiang Yang. 2018. CoNet: Collaborative Cross Networks for Cross-Domain Recommendation. In *Proc. of CIKM*.

[13] Xiao Hu, Yuan Cheng, Zhi Zheng, Yue Wang, Xinxin Chi, and Hengshu Zhu. 2023. BOSS: A Bilateral Occupational-Suitability-Aware Recommender System for Online Recruitment. In *Proc. of KDD*.

[14] Chao Huang. 2021. Recent Advances in Heterogeneous Relation Learning for Recommendation. In *Proc. of IJCAI*.

[15] Yuzhen Huang, Xiaohan Wei, Xing Wang, Jiyan Yang, Bor-Yiing Su, Shivam Bharuka, Dhruv Choudhary, Zewei Jiang, Hai Zheng, and Jack Langman. 2021. Hierarchical training: Scaling deep recommendation models on large cpu clusters. In *Proc. of KDD*.

[16] Fortune Business Insights. [n. d.]. Online Recruitment Technology Market: Forecast Report 2030.

[17] Ling Jian, Chongzhi Rao, and Xiao Gu. 2024. Your Profile Reveals Your Traits in Talent Market: An Enhanced Person-Job Fit Representation Learning. (2024).

[18] Yangqin Jiang, Chao Huang, and Lianghao Huang. 2023. Adaptive graph contrastive learning for recommendation. In *Proc. of KDD*.

[19] Eyad Kannout, Michal Grodzki, and Marek Grzegorowski. 2023. Towards addressing item cold-start problem in collaborative filtering by embedding agglomerative clustering and FP-growth into the recommendation system. *Comput. Sci. Inf. Syst.* (2023).

[20] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* (2009).

[21] Cheng-Te Li, Chia-Tai Hsu, and Man-Kwan Shan. 2018. A Cross-Domain Recommendation Mechanism for Cold-Start Users Based on Partial Least Squares Regression. *ACM Trans. Intell. Syst. Technol.* (2018).

[22] Lei Li and Tao Li. 2013. News recommendation via hypergraph learning: encapsulation of user behavior and news content. In *Proc. of WSDM*.

[23] Muyang Li, Zijian Zhang, Xiangyu Zhao, Wanyu Wang, Minghao Zhao, Runze Wu, and Ruocheng Guo. 2023. Automlp: Automated mlp for sequential recommendations. In *Proceedings of the ACM Web Conference 2023*.

[24] Xinhang Li, Zhaopeng Qiu, Xiangyu Zhao, Zihao Wang, Yong Zhang, Chunxiao Xing, and Xian Wu. 2022. Gromov-wasserstein guided representation learning for cross-domain recommendation. In *Proc. of CIKM*.

[25] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *Proc. of WWW*.

[26] Shu-Hsien Liao, Retno Widowati, and Yu-Chieh Hsieh. 2021. Investigating online social media users' behaviors for social commerce recommendations. *Technology in Society* (2021).

[27] Hao Lin, Hengshu Zhu, Yuan Zuo, Chen Zhu, Junjie Wu, and Hui Xiong. 2017. Collaborative company profiling: Insights from an employee's perspective. In *Proc. of AAAI*.

[28] Zihan Lin, Changxin Tian, Yupeng Hou, and Wayne Xin Zhao. 2022. Improving graph collaborative filtering with neighborhood-enriched contrastive learning. In *Proceedings of the ACM Web Conference 2022*.

[29] Bulou Liu, Bing Bai, Weibang Xie, Yiwen Guo, and Hao Chen. 2022. Task-optimized user clustering based on mobile app usage for cold-start recommendations. In *Proc. of KDD*.

[30] Qidong Liu, Fan Yan, Xiangyu Zhao, Zhaocheng Du, Huifeng Guo, Ruiming Tang, and Feng Tian. 2023. Diffusion augmentation for sequential recommendation. In *Proc. of CIKM*.

[31] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: Short-Term Attention/Memory Priority Model for Session-based Recommendation. In *Proc. of KDD*.

[32] Zhiwei Liu, Yongjun Chen, Jia Li, Philip S. Yu, Julian J. McAuley, and Caiming Xiong. 2021. Contrastive Self-supervised Sequential Recommendation with Robust Augmentation. *CoRR* (2021).

[33] Ziru Liu, Jiejie Tian, Qingpeng Cai, Xiangyu Zhao, Jingtong Gao, Shuchang Liu, Dayou Chen, Tonghao He, Dong Zheng, Peng Jiang, et al. 2023. Multi-task recommendations with reinforcement learning. In *Proceedings of the ACM Web Conference 2023*.

[34] Xia Ning and George Karypis. 2011. Slim: Sparse linear methods for top-n recommender systems. In *Proc. of ICDM*.

[35] Yitong Pang, Lingfei Wu, Qi Shen, Yiming Zhang, Zhihua Wei, Fangli Xu, Ethan Chang, Bo Long, and Jian Pei. 2022. Heterogeneous global graph neural networks for personalized session-based recommendation. In *Proc. of WSDM*.

[36] Bibek Paudel, Fabian Christoffel, Chris Newell, and Abraham Bernstein. 2016. Updatable, accurate, diverse, and scalable recommendations for interactive applications. *ACM Transactions on Interactive Intelligent Systems (TiiS)* (2016).

[37] Chuan Qin, Le Zhang, Rui Zha, Dazhong Shen, Qi Zhang, Ying Sun, Chen Zhu, Hengshu Zhu, and Hui Xiong. 2023. A comprehensive survey of artificial intelligence techniques for talent analytics. *arXiv preprint arXiv:2307.03195* (2023).

[38] Michael Reusens, Wilfried Lemahieu, Bart Baesens, and Luc Sels. 2018. Evaluating recommendation and search in the labor market. *Knowledge-Based Systems* (2018).

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

[40] Ian H Sloan and William E Smith. 1980. Product integration with the Clenshaw-Curtis points: implementation and error estimates. *Numer. Math.* (1980).

[41] Harald Steck. 2019. Embarrassingly shallow autoencoders for sparse data. In *Proc. of WWW*.

[42] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proc. of CIKM*.

[43] Lloyd N Trefethen. 2008. Is Gauss quadrature better than Clenshaw–Curtis? *SIAM review* (2008).

[44] Chenyang Wang, Yuanqing Yu, Weizhi Ma, Min Zhang, Chong Chen, Yiqun Liu, and Shaoping Ma. 2022. Towards representation alignment and uniformity in collaborative filtering. In *Proc. of KDD*.

[45] Chao Wang, Hengshu Zhu, Chen Zhu, Chuan Qin, and Hui Xiong. 2020. Setrank: A setwise bayesian approach for collaborative ranking from implicit feedback. In *Proc. of AAAI*.

[46] Jun Wang, Arjen P De Vries, and Marcel JT Reinders. 2006. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proc. of SIGIR*.

[47] Jianling Wang, Kaize Ding, Liangjie Hong, Huan Liu, and James Caverlee. 2020. Next-item recommendation with sequential hypergraphs. In *Proc. of SIGIR*.

[48] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proc. of SIGIR*.

[49] Yuhao Wang, Ha Tsz Lam, Yi Wong, Ziru Liu, Xiangyu Zhao, Yichao Wang, Bo Chen, Huifeng Guo, and Ruiming Tang. 2023. Multi-Task Deep Recommender Systems: A Survey. *CoRR* (2023).

[50] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *Proc. of ICML*.

[51] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-supervised graph learning for recommendation. In *Proc. of SIGIR*.

[52] Likang Wu, Zhaopeng Qiu, Zhi Zheng, Hengshu Zhu, and Enhong Chen. 2024. Exploring large language model for graph data understanding in online job recommendations. In *Proc. of AAAI*.

[53] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *Proc. of WSDM*.

[54] Xin Xia, Hongzhi Yin, Junliang Yu, Qinyong Wang, Lizhen Cui, and Xiangliang Zhang. 2021. Self-supervised hypergraph convolutional networks for session-based recommendation. In *Proc. of AAAI*.

[55] Xu Xie, Fei Sun, Zhaoyang Liu, Shiwen Wu, Jinyang Gao, Jiandong Zhang, Bolin Ding, and Bin Cui. 2022. Contrastive learning for sequential recommendation. In *2022 IEEE 38th international conference on data engineering (ICDE)*.

[56] Tanya V Yadalam, Vaishnavi M Gowda, Vanditha Shiva Kumar, Disha Girish, and M Namratha. 2020. Career recommendation systems using content based filtering. In *2020 5th International Conference on Communication and Electronics Systems (ICCES)*.

[57] Shuo Yang, Mohammed Korayem, Khalifeh AlJadda, Trey Grainger, and Sriraam Natarajan. 2017. Combining content-based and collaborative filtering for job recommendation system: A cost-sensitive Statistical Relational Learning approach. *Knowledge-Based Systems* (2017).

[58] Yuhao Yang, Chao Huang, Lianghao Xia, and Chenliang Li. 2022. Knowledge graph contrastive learning for recommendation. In *Proc. of SIGIR*.

[59] Tiansheng Yao, Xinyang Yi, Derek Zhiyuan Cheng, Felix Yu, Ting Chen, Aditya Menon, Lichan Hong, Ed H Chi, Steve Tjoa, Jieqi Kang, et al. 2021. Self-supervised learning for large-scale item recommendations. In *Proc. of CIKM*.

[60] Chi Zhang, Rui Chen, Xiangyu Zhao, Qilong Han, and Li Li. 2023. Denoising and Prompt-Tuning for Multi-Behavior Recommendation. In *Proceedings of the ACM Web Conference 2023, WWW 2023, Austin, TX, USA, 30 April 2023 - 4 May 2023*.

[61] Mengqi Zhang, Shu Wu, Meng Gao, Xin Jiang, Ke Xu, and Liang Wang. 2020. Personalized graph neural networks with attention mechanism for session-aware recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2020).

[62] Xiangyu Zhao, Tong Xu, Qi Liu, and Hao Guo. 2016. Exploring the choice under conflict for social event participation. In *Proc. of DASFAA*.

[63] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin. 2018. Recommendations with negative feedback via pairwise deep reinforcement learning. In *Proc. of KDD*.

[64] Jiawei Zheng, Qianli Ma, Hao Gu, and Zhenjing Zheng. 2021. Multi-view denoising graph auto-encoders on heterogeneous information networks for cold-start recommendation. In *Proc. of KDD*.

[65] Zhi Zheng, Xiao Hu, Zhaopeng Qiu, Yuan Cheng, Shanshan Gao, Yang Song, Hengshu Zhu, and Hui Xiong. 2024. Bilateral Multi-Behavior Modeling for Reciprocal Recommendation in Online Recruitment. *IEEE Transactions on Knowledge and Data Engineering* (2024).

[66] Zhi Zheng, Zhaopeng Qiu, Tong Xu, Xian Wu, Xiangyu Zhao, Enhong Chen, and Hui Xiong. 2022. CBR: context bias aware recommendation for debiasing user modeling and click prediction. In *Proceedings of the ACM Web Conference 2022*.

A NOTATIONS

We summarize all notations in this paper and list them in Table 6.

Table 6: Notations in this paper.

Notation	Description
$\mathcal{S}^u$	The set of session-based interactions
$\mathcal{G}$	Interaction graph
$\mathcal{V}$	Group-based node-set, $\mathcal{V} = \{v_1, v_2, \cdots\}$
$\mathcal{E}$	The hyperedge set, $\mathcal{E} = \{e_1, e_2, \cdots\}$
$\mathcal{U}$	The user set, $\mathcal{U} = \{u_1, u_2, \cdots\}$
$\mathcal{X}$	Graph features that extracted from user/job contents
$\boldsymbol{D}^u$	The user resume set
$\boldsymbol{D}^v$	The job requirement set
$\boldsymbol{C}_u$&$\boldsymbol{C}_v$	User/Job groups
$\mathcal{L}$	Graph Laplacian matrix
$\boldsymbol{H}$	The node-edge relationship matrix of the hypergraph
$\boldsymbol{D}_v$&$\boldsymbol{D}_e$	The degree matrix of nodes and hyperedges
$\boldsymbol{U}$	Eigen-vectors of graph Laplacian matrix
$\boldsymbol{g}_\theta$	Graph filter with training parameter θ
ξ	The ratio of the users (jobs) to user (job) groups
p	The order of Chebyshev approximation

B MODEL COMPLEXITY

Our proposed framework, BISTRO, is efficient. We will analyze it from two aspects: theoretical [from $O(n^3)$ to $O(n \log n)$] and application (less than 100ms per sample) level.

Theoretical level

Among all modules in BISTRO, the graph neural network (GNN) is considered very time-consuming. Actually, graph learning-based recommender systems have computational limitations in practice. To address this issue, we cluster users (jobs) based on the semantic information in their resumes (requirements) using the K-Means algorithm and use a simple RNN to extract the personalized preference for a person in the user group. The extraction of preference features based on user (job) groups reduces the computational overhead of GNNs. Noting that eigendecomposition in GNNs is resource-intensive, we place it by using the Chebyshev polynomial estimation, and the Chebyshev coefficient in the polynomial can be computed by Fast Fourier Transform, which reduces the computational complexity from exponential complexity $O(n^3)$ to $O(n \log n)$. Therefore, our algorithm is efficient.

Application level

In practice, the training of all models is performed offline. For example, we use spark cluster to calculate the clustering center of each group and use HGNN to learn the corresponding representation of groups. For new or updated users and jobs, we assign them to the nearest group based on semantic clustering. Only the RNN module operates online, inferring personalized user representations within groups. In the online experiment, the 99% Response Time of BISTRO is less than 100ms.

C EXPERIMENT DETAIL

C.1 Baselines Detail

The details of these baselines are as follows:

• BasicMF [20]: A model that combines matrix factorization with a Multilayer Perceptron (MLP) for recommendations.
• ItemKNN [46]: A recommender that utilizes item-based collaborative filtering.
• PureSVD [7]: An approach that applies Singular Value Decomposition for recommendation tasks.
• SLIM [34]: A recommendation method known as the Sparse Linear Method.
• DAE [53]: Stands for Collaborative Denoising Auto-Encoder, used in recommendation systems.
• MultVAE [25]: A model extending Variational Autoencoders to collaborative filtering for implicit feedback.
• EASE [41]: A recommendation technique called Embarrassingly Shallow Autoencoders for Sparse Data.
• P3a [6]: A method that uses ordering rules from random walks on a user-item graph.
• RP3b [36]: A recommender that re-ranks items based on 3-hop random walk transition probabilities.
• NGCF [48]: Employs graph embedding propagation layers to generate user/item representations.
• LightGCN [10]: Utilizes neighborhood information in the user-item interaction graph.
• SLRec [59]: A method using contrastive learning among node features.
• SGL [51]: Enhances LightGCN with self-supervised contrastive learning.
• GCCF [5]: A multi-layer graph convolutional network for recommendation.
• NCL [28]: Enhances recommendation models with neighborhood-enriched contrastive learning.
• DirectAU [44]: Focuses on the quality of representation based on alignment and uniformity.
• HG-GNN [35]: Constructs a heterogeneous graph with both user nodes and item nodes and uses a graph neural network to learn the embedding of nodes as a potential representation of users or items.
• A-PGNN [61]: Uses GNN to extract session representations for intra-session interactions and uses an attention mechanism to learn features between sessions.
• AdaGCL [18]: Combines a graph generator and a graph denoising model for contrastive views.
• MvDGAE [64]: Stands for Multi-view Denoising Graph AutoEncoders.
• STAMP [31]: A model based on the attention mechanism to model user behavior sequence data.
• GRU4Rec [11]: Utilizes Gated Recurrent Units for session-based recommendations.
• BERT4Rec [42]: A model for the sequence-based recommendation that handles long user behavior sequences.
• CL4Rec [55]: An improved version of BERT4Rec with locality-sensitive hashing for faster item retrieval.
• CoScRec [32]: It explores an innovative recommendation approach that enhances sequential recommendation systems through robust data augmentation and contrastive self-supervised learning techniques.
• TiCoSeRec [9]: A method based on CoSeRec, utilizing data augmentation algorithms for sequence recommendation improvement.

Table 7: Results under different settings of k.

	$k = 5$	$k = 10$	$k = 20$
H@k	0.6059	0.6247	0.6303
M@k	0.5117	0.5131	0.5051

C.2 The number of recommended jobs

The hyperparameter, k, also has a critical impact on experimental results. We set $k = 5$, 10 and 20 to conduct experiments respectively. The experimental results are shown in Table 7. It can be seen our model performs well under all settings.

C.3 Case Study

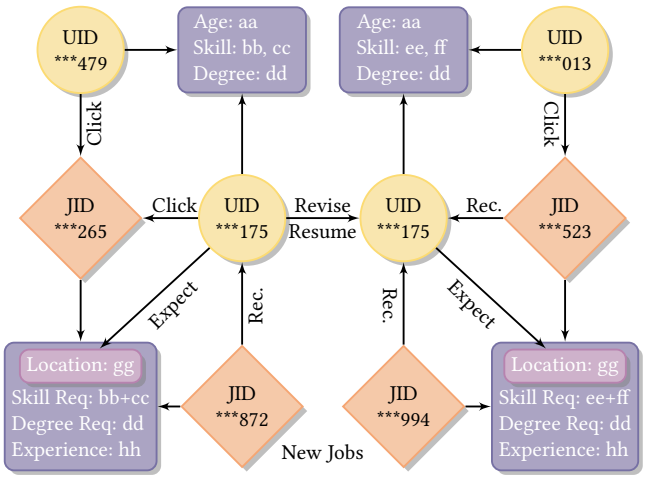


Figure 8: A real-life scenario for the job recommendation.

Beyond its effectiveness in performance, BISTRO also boasts considerable interpretability. To demonstrate how the framework mitigates both the job preference drift and data noise problems, we present a real-life scenario to illustrate the logic behind the suggestions made by BISTRO, as shown in Figure 8.

In this figure, jobs with IDs ***872 and ***994 are two job positions that are newly posted in the online recruitment system, while IDs ***265 and ***523 are two job positions that a large number of users interact with frequently. Among them, ***872 and ***265, as well as ***994 and ***523, have similar occupational demand descriptions respectively. Also, the user with ID ***175 shared a similar resume with user ID ***479 before ***175 modified the resume, and after his resume was changed, ***175 had a similar content with user ***013. Recommendations in this scenario can be divided into three examples:

Example 1 (Recommendation for a dynamically changing user) Consider the user represented by ID ***175, BISTRO addresses this challenge by deploying content-based analysis. The framework utilizes the user's social network and a set of resume attributes collected to create a composite feature profile to identify users with similar tastes. Subsequently, it recommends a job with ID ***523 favored by a like-minded user with ID ***013 to him.

Example 2 (Recommendation for a new job) A newly posted job with ID ***872 lacks any user interaction data, complicating the generation of a meaningful representation for it. BISTRO, however, overcomes this by incorporating auxiliary information such as skill requirements and working experience, and then associated tags to locate similar content. By leveraging this approach combined with the user's expectations, BISTRO acquires a rich and informative embedding for the job, enabling it to recommend the job to users who have shown an interest in comparable jobs.

Example 3 (Recommend a new job to a dynamically changing user) Combining both two situations illustrated above, BISTRO deals with this complex challenge by utilizing a wavelet graph denoising filter and graph representation method. In this way, it can recommend the latest jobs with similar job content to users with the same real-time needs as well as similar user content characteristics.

Table 8: Results of different job recommender systems.

	Shenzhen		**Shanghai**		**Beijing**	
	H@10	**M@10**	**H@10**	**M@10**	**H@10**	**M@10**
InEXIT	0.4131	0.2936	0.4611	0.3762	0.5141	0.4033
DGMN	0.4274	0.3178	0.4897	0.3992	0.5217	0.4125
APJFMF	0.4352	0.3114	0.4863	0.3910	0.5254	0.4166
Ours	**0.5598**	**0.5467**	**0.6166**	**0.5152**	**0.6247**	**0.5131**

Bold indicates the statistically significant improvements (*i.e.,* two-sided t-test with $p < 0.05$) over the best baseline (underlined). For all metrics: the higher, the better.

In addition, we also compare the proposed framework, BISTRO, with multiple job state-of-the-art recommender systems, *i.e.,* InEXIT [39], DGMN [1], and APJFMF [17]. The result can be found in Table 8. We can see that our framework acheves the best among all baselines, which verify the effectiveness of our method.

Received 8 February 2024; revised 5 April 2024; accepted 16 May 2024