3. Experimental Results for additional SBR methods (Q8, Q9)

As mentioned in the rebuttal, the experimental results of combining sessions of each person in the paper and multiplying them into ‘large’ sessions were listed below, all methods in [9]&[22] had improved to a certain extent, and the model performance of STAMP had greatly improved under this settings.

	Shenzhen		Shanghai		Beijing	
	H@10	**M**@10	**H**@10	**M**@10	**H**@10	**M**@10
GRU4Rec [9]	0.3271	0.3562	0.3590	0.3268	0.3661	0.3394
STAMP [22]	0.3285	0.3627	0.3603	0.3291	0.3675	0.3434

We also conducted experiments on two additional SOTA SBR methods (HG-GNN[25] and A-PGNN[43]) in the revised version. The results are as follows:

	Shenzhen		Shanghai		Beijing	
	H@10	**M**@10	**H**@10	**M**@10	**H**@10	**M**@10
HG-GNN [25]	0.3131	0.1936	0.3637	0.2826	0.4204	0.3430
A-PGNN [43]	0.3372	0.2069	0.3876	0.3019	0.4485	0.3657
Ours	0.5598	0.6166	0.6166	0.5152	0.6247	0.5131

Reference 25 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. Reference 43 uses GNN to extract session representations for intra-session interactions and uses an attention mechanism to learn features between sessions. However, neither of the above two methods could address both preference drift and noise interaction issues at the same time, so their model performance is not as good as our proposed framework.

4. The novelty of the work (Q4)

The major novelty of this paper is addressing a novel problem of user preference drift in job recommendation.

We present BISTRO, a framework that models user preferences in real time by integrating semantic and behavioral data. Initially, we cluster users and jobs using K-Means based on resume and job requirement semantics. We address preference drift using semantic resume data, organizing user-job interactions into sessions represented by hypergraphs, which are processed using a hypergraph neural network (HGNN) to

refine user preferences. To improve efficiency, we redesign the HGNN to reduce complexity from $O(n^3)$ to $O(n\log n)$. Besides, we utilize a wavelet kernel derived from spectral graph theory to filter noise interactions among user groups efficiently. Finally, an RNN is employed at the end of our framework to capture personal preferences from interactions precisely.

In conclusion, BISTRO effectively addresses preference drift in job recommendations and has proven robust in both theoretical and practical tests.

5. Model Complexity (Q10,Q11)

Our proposed framework, BISTRO, is very 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.

(i) 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 eigen decomposition 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.

(ii) 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.

6. Length of session (Q3,Q6)

The length of each session is 28 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, as illustrated in the left part of Fig. 1. It is seen from the picture in the link on the left that the length of most sessions is between 19-37. In the picture on the right, we screen out data with different session lengths and conduct separate experiments. We compare the proposed method with the top two best methods in the baseline. 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 method is more robust than the other two methods and can better resist noise interference. We will add it to the final version of the paper.

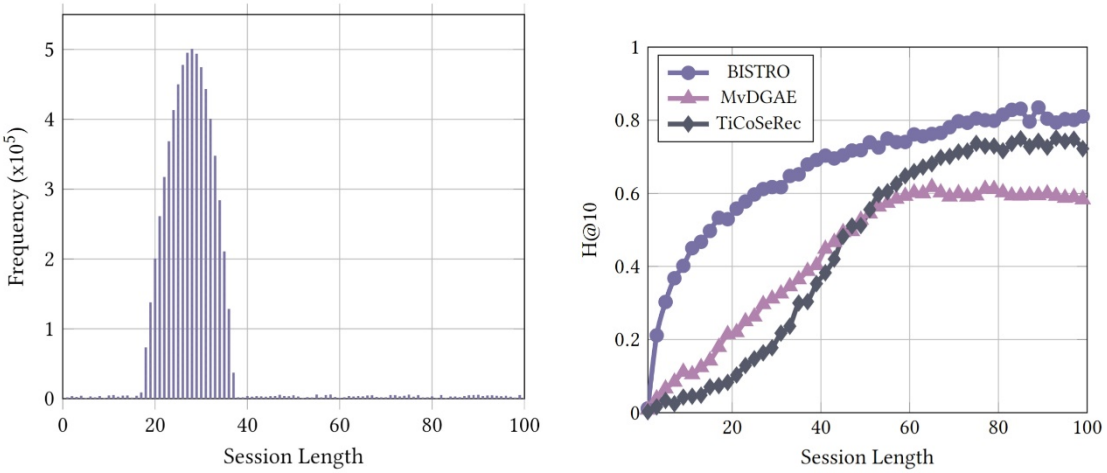

Fig. 1

Then we will answer all the other questions you mentioned.

Q5: Explain the choice of neural architecture used

A5: Thanks for your questions. All three modules in BISTRO were chosen because of their efficiency and widespread use.

Initially, because of the high efficiency of K-Means, we cluster users and jobs using it based on resume and job requirement semantics.

In job recommendation, the preference drift can be intuitively captured through the semantic information in the resumes of users. Thus, we divide the user-job interactions into multiple sessions based on resume updating. These hierarchical sessions can be represented losslessly by hypergraphs, and then a hypergraph neural network (HGNN) is deployed to learn the refined user preference.

To reduce the computational cost in HGNN, a novel Chebyshev polynomial approximation is proposed. Besides, a wavelet kernel is deployed to filter noise

interactions from different groups of users efficiently. Finally, to extract the personalized preference of a single person in the group as a classic sequence processing problem, we use a simple RNN as the last module of the framework. We will polish the explanation at the beginning of the Methodology to make it clearer.

Q7: Only HR and MRR as evaluation metrics

A7: Thanks for your concerns. Due to limited space, we did not list other metrics are not listed in Table 2.
Taking the Shenzhen data set as an example, the experimental results of our method and the top-2 methods in baselines in ***Precision***, ***Recall***, ***F1 Score***, ***Mean Reciprocal Rank***, ***Normalized Discounted Cumulative Gain***, ***Hit Ratio***, and ***Mean Average Precision*** are as follows. We will list other metrics in the revised paper.

	Precision	Recall	F1 Score	Mean Reciprocal Rank	Normalized Discounted Cumulative Gain	Hit Ratio	Mean Average Precision
BasicMF	0.1843	0.2028	0.1826	0.2837	0.1762	0.2791	0.1295
ItemKNN	0.1852	0.2052	0.1836	0.2969	0.1830	0.2479	0.1279
PureSVD	0.2614	0.2885	0.2757	0.4205	0.2699	0.4422	0.1913
SLIM	0.0262	0.0282	0.0301	0.0443	0.0144	0.0450	0.0107
DAE	0.1250	0.1264	0.1278	0.2024	0.1326	0.1867	0.0925
MultVAE	0.2503	0.2796	0.2656	0.4149	0.2486	0.4249	0.1831
EASE	0.2040	0.2123	0.2072	0.3192	0.1957	0.2891	0.1447
SLRec	0.2392	0.2516	0.2477	0.3796	0.2300	0.2767	0.1729
SGL	0.2243	0.2384	0.2448	0.3618	0.2301	0.4089	0.1711
P3a	0.2276	0.2600	0.2494	0.3748	0.2282	0.3185	0.1747
RP3b	0.1842	0.1929	0.1980	0.3017	0.1910	0.2917	0.1402
NGCF	0.1931	0.1874	0.1842	0.2859	0.1792	0.3557	0.1328
LightGCN	0.2283	0.2412	0.2357	0.3647	0.2251	0.3785	0.1742
GCCF	0.2243	0.2335	0.2317	0.3460	0.2165	0.3756	0.1588
NCL	0.2474	0.2637	0.2493	0.3917	0.2438	0.3424	0.1810
DirectAU	0.2215	0.2531	0.2410	0.3700	0.2255	0.3725	0.1688
AdaGCL	0.2789	0.3051	0.2895	0.4417	0.2738	0.3867	0.1950
MvDGAE	0.3317	0.3462	0.3374	0.5258	0.3298	0.5028	0.2317
STAMP	0.1781	0.1988	0.1840	0.2924	0.1900	0.2754	0.1268

GRU4Rec	0.2166	0.2259	0.2207	0.3449	0.2057	0.3247	0.1522
BERT4Rec	0.2359	0.2320	0.2350	0.3607	0.2262	0.3395	0.1603
CL4Rec	0.2518	0.2814	0.2690	0.4124	0.2559	0.3882	0.1822
CoScRec	0.2822	0.3043	0.2796	0.4352	0.2641	0.4096	0.1951
TiCoSeRec	0.2831	0.3137	0.3019	0.4671	0.2810	0.4397	0.1769
Ours	**0.3594**	**0.3728**	**0.3651**	**0.5467**	**0.3414**	**0.5598**	**0.2792**

The reason we selected HR and MRR as metrics is they are two common indicators in the recommendation system. Hit Ratio (HR) reflects whether the item actually clicked by the user is included in the recommendation sequence, and the Mean Reciprocal Rank (MRR) refers to whether the found item is placed in a more conspicuous position for the user. The former emphasizes the correctness of recommendations, while the latter emphasizes the order of recommendations. We will add this to the experiment section in the final revision.