Management Science

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

The Good, the Bad, and the Unhirable: Recommending Job Applicants in Online Labor Markets

Marios Kokkodis, Panagiotis G. Ipeirotis

To cite this article:
Marios Kokkodis, Panagiotis G. Ipeirotis (2023) The Good, the Bad, and the Unhirable: Recommending Job Applicants in Online Labor Markets. Management Science 69(11):6969-6987. https://doi.org/10.1287/mnsc.2023.4690

https://pubsonline.informs.org/journal/mnsc

MANAGEMENT SCIENCE
Vol. 69, No. 11, November 2023, pp. 6969–6987
ISSN 0025-1909 (print), ISSN 1526-5501 (online)

The Good, the Bad, and the Unhirable: Recommending Job Applicants in Online Labor Markets

Marios Kokkodis,[a,*] **Panagiotis G. Ipeirotis**[b]

[a] Unaffiliated; [b] New York University, New York, New York 10012

*Corresponding author

Contact: marios.kokkodis@gmail.com, https://orcid.org/0000-0002-5037-6060 (MK); panos@stern.nyu.edu, https://orcid.org/0000-0002-2966-7402 (PGI)

Received: February 24, 2019
Revised: April 26, 2020; March 21, 2021; August 26, 2022; January 5, 2023
Accepted: January 12, 2023
Published Online in Articles in Advance: March 8, 2023

https://doi.org/10.1287/mnsc.2023.4690

Abstract. Choosing job applicants to hire in online labor markets is hard. To identify the best applicant at hand, employers need to assess a heterogeneous population. Recommender systems can provide targeted job-applicant recommendations that help employers make better-informed and faster hiring choices. However, existing recommenders that rely on multiple user evaluations per recommended item (e.g., collaborative filtering) experience structural limitations in recommending job applicants: Because each job application receives only a single evaluation, these recommenders can only estimate noisy user-user and item-item similarities. On the other hand, existing recommenders that rely on classification techniques overcome this limitation. Yet, these systems ignore the hired worker's performance—and, as a result, they uniformly reinforce prior observed behavior that includes unsuccessful hiring choices—while they overlook potential sequential dependencies between consecutive choices of the same employer. This work addresses these shortcomings by building a framework that uses job-application characteristics to provide recommendations that (1) are unlikely to yield adverse outcomes (performance-aware) and (2) capture the potentially evolving hiring preferences of employers (sequence-aware). Application of this framework on hiring decisions from an online labor market shows that it recommends job applicants who are likely to get hired and perform well. A comparison with advanced alternative recommender systems illustrates the benefits of modeling performance-aware and sequence-aware recommendations. An empirical adaptation of our approach in an alternative context (restaurant recommendations) illustrates its generalizability and highlights its potential implications for users, employers, workers, and markets.

History: Accepted by Kartik Hosanagar, information systems.
Supplemental Material: The online appendix is available at https://doi.org/10.1287/mnsc.2023.4690.

Keywords: performance-aware sequential recommender systems • personalization • hiring choices in online labor markets

1. Introduction

Online labor markets, such as PeoplePerHour, Freelancer, and Upwork, facilitate global short-term contracts or freelance work. Employers can purchase services from online workers that complete diverse jobs, including web development, graphic design, accounting, sales, marketing, and data science. Like other online platforms, online labor markets grew exponentially during the past decade (Freelancers-union 2017). Upwork, for example, hosts 14 million workers and 5 million employers and reports a total annual transaction volume of $1 billion (Lauren 2017, Brier and Pearson 2018). Similarly, PeoplePerHour connects 750,000 employers to 1.5 million workers around the world (Atkins 2019). This growth is projected to continue as automation and the sharing economy structure the future of work (Sundararajan 2016, Institute of Business Value 2019).

Similar to offline settings, identifying capable workers to hire in online labor markets is hard (Klazema 2018). To make hiring decisions, employers need to assess the observed and latent characteristics of the available job applicants. The observed characteristics include the applicants' education, skills, work histories, and certifications, as listed on their resumes (Kokkodis et al. 2015). The latent characteristics are the workers' actual knowledge and abilities (when skill certifications are absent), as well as the workers' motivation, drive, and willingness to collaborate and do a good job (Geva and Saar-Tsechansky 2016). The existence of latent characteristics, the heterogeneity that appears in the observed ones (Kokkodis and Ipeirotis 2014), and the interactions between the two create an uncertain environment of information asymmetry (Akerlof 1970, Pelletier and Thomas 2018).

Besides, because job applications are free, workers often broadcast their availability widely to increase their chances of getting hired (Kokkodis et al. 2015). Large numbers of job applications increase employers' search

costs (Guo et al. 2017) and may even result in unfilled openings (Carr 2003, Snir and Hitt 2003). By some estimates, around 60% of openings in online labor markets never reach a contract (Zheng et al. 2015). As a result, increased search costs hurt both the market (through lack of revenue) and its users (employers and workers), many of whom opt to quit (Autor 2001, Guasch et al. 2003).

Information asymmetry and search costs are not unique to online labor markets. Both are present in almost every type of online market (Ba and Pavlou 2002, Chen et al. 2004, Dimoka et al. 2012). By providing targeted product or service recommendations, recommender systems are a popular solution to these issues (Fleder and Hosanagar 2009, Pathak et al. 2010, Brynjolfsson et al. 2011). Specifically, in our context, recommender systems can help employers make better-informed decisions *by ranking job applicants according to their likelihood of getting hired and performing well.*

Yet, when applied to job-applicant recommendations, existing recommender systems experience shortcomings. In particular, recommenders that rely on *many assessments* per item to provide recommendations (e.g., collaborative filtering; we refer to these systems as *many-assessment* recommenders; see Adomavicius and Tuzhilin 2005, Ricci et al. 2011, and Quadrana et al. 2018) have limited information to estimate the required user-user and item-item similarities. This is because, in online labor markets, (1) task requirements are diverse (i.e., no two jobs are identical); (2) job applicants evolve by gaining experience or learning new skills; (3) different job openings attract different pools of applicants, and, as a result, tasks have nonoverlapping choice sets; and (4) the ratio of employers to workers is significantly lower than conventional many-assessment contexts (e.g., movie recommendations). These characteristics generate virtually unique *job applications*—the focal recommended item—that are evaluated only once by a single employer. Hence, when applied in this context, many-assessment systems will underperform, as they will rely on a single observed assessment per job application to estimate noisy user-user and item-item similarities.

On the other hand, systems that rely on a *single assessment* to provide recommendations overcome these limitations (we refer to these systems as *single-assessment* recommenders; see Kokkodis et al. 2015, Mao et al. 2015, Baba et al. 2016, and Abhinav et al. 2017). However, existing single-assessment systems have two shortcomings when adapted to recommend job applicants. First, they ignore the performance of the hired applicant. Instead, they learn and uniformly reinforce previously observed behavior, including employer choices that yielded unsuccessful outcomes (Kokkodis et al. 2015, Abhinav et al. 2017). Second, they overlook potential sequential dependencies between hiring decisions of the same employers. Hence, they implicitly assume that employer hiring preferences remain the same over repeated hiring choices. As a result, they make recommendations that regress to a mean that uniformly aggregates behaviors of varying-experience and varying-ability employers.

This work identifies three principles for designing job-applicant recommenders that address these limitations of existing many-assessment and single-assessment systems: A job-applicant recommender should be a *single-assessment* system that is both *performance-aware* and *sequence-aware*. Single-assessment systems overcome the limitations of many-assessment approaches and learn to recommend items that only a single user evaluates. Performance-aware systems identify prior unsuccessful hiring choices and learn to promote job applicants who are not only hirable, but also likely to perform well. Sequence-aware systems allow repeat employers to adjust their hiring preferences and evolve independently.

We use these principles to design a classification framework that conceptualizes three discrete job-application outcomes ("No-hire," "Hire-negative," and "Hire-positive"—*performance-aware*) and captures any changes in employer hiring-preferences through a Hidden Markov Model (HMM—*sequence-aware*). Implementation of the proposed approach on hiring decisions from a major online labor market highlights the advantages of the three design principles. Across four different evaluation metrics, our framework significantly outperforms alternative job-applicant recommenders, including existing and new single-assessment systems (logistic regression, gradient boosting, random forests, support vector machines, and recurrent neural networks) and adaptations of popular many-assessment systems, such as collaborative filtering-based approaches (singular value decomposition, HMM for collaborative filtering; see Sahoo et al. 2012) and deep sequential recommenders (Kula 2018). Repeat employers benefit the most from our approach, as these employers receive personalized sequence-aware recommendations by evolving across their distinct hiring-preference paths. Application of our approach in an alternative context (restaurant recommendations) shows its generalizability in contexts where both the recommended items and the user preferences change.

This work extends research in recommender systems and online labor markets by identifying and addressing critical shortcomings of existing many-assessment and single-assessment approaches when recommending job applicants. By conceptualizing the necessary design principles, this paper is the first to directly incorporate performance outcomes into the recommendation process and allow employer hiring preferences to change. Methodologically, compared with other HMM-based recommenders (Sahoo et al. 2012, Hosseinzadeh Aghdam et al. 2015, Zhang et al. 2016) and traditional HMM approaches for classification (Murphy 2012), the proposed

HMM offers a unique structure that models choices according to observed applicant-employer-task characteristics and allows hiring preferences to evolve only after the completion of each task. As a result, because our framework provides job-applicant recommendations that lead to successful outcomes, it can benefit (1) workers to differentiate, (2) employers to make better-informed and faster (reduced search cost; see Bakos 1997) decisions, and (3) markets to increase their transaction efficiency, which, in turn, results in increased revenue and employer satisfaction.

2. Research Context

The ultimate goal of this work is to provide relevant job-applicant recommendations in the context of an online labor market.

Problem Definition. Assume a given job opening with a set of job applicants. We are interested in ranking these applicants according to their likelihood of getting hired and performing well.

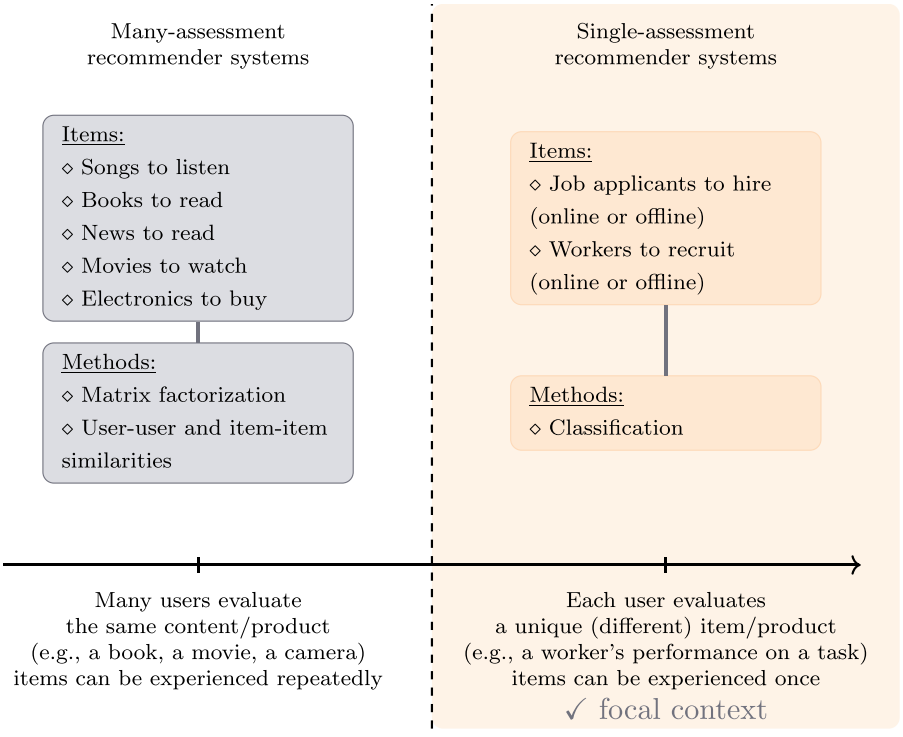
At the problem's core is a *recommender system* that ranks job applications within a given opening. Existing recommender systems (Adomavicius and Tuzhilin 2005, Kantor et al. 2011, Quadrana et al. 2018, Zhang et al. 2019) can be many-assessment or single-assessment depending on the items they recommend (Figure 1):

- Many-assessment recommenders: Many users evaluate each recommended item, as the underlying evaluated content (e.g., a book's story, a movie's finale, or a camera's chip) does not change. Examples include systems that recommend books, songs, and movies (Adomavicius and Tuzhilin 2005, Ricci et al. 2011, Quadrana et al. 2018). The availability of multiple evaluations per item allows these systems to estimate item-item and user-user similarities. Through matrix-factorization techniques, they predict user-item affinities and make recommendations (Koren et al. 2009).
- Single-assessment recommenders: Only a single user evaluates each recommended item, as these items (e.g., a worker's performance on a unique task that requires a certain combination of skills and abilities) can be experienced only once. Examples include systems that recruit workers or rank job applicants (Goswami et al. 2014, Kokkodis et al. 2015, Mao et al. 2015, Abhinav et al. 2017). These systems use classification approaches that model observed item characteristics to provide recommendations.[1]

2.1. Characteristics of the Focal Context

Our research context assumes a marketplace where workers apply to job openings, some of them get hired, and, once, they complete the required task, they receive feedback (rating) about their performance. This behavior

Figure 1. (Color online) Recommender Systems Vary According to the Uniqueness of the Focal Recommended Items



Notes. Many-assessment systems recommend items that multiple users evaluate by estimating user-user and item-item similarities. Single-assessment systems recommend unique items that a single user evaluates by modeling item characteristics through classification techniques.

suggests that some workers might get hired (and, hence, evaluated) multiple times over a period of time. Hence, one could argue that many-assessment frameworks could provide efficient solutions to our research question, as they could leverage user (employer) evaluations across multiple items (workers) to estimate the required user-user and item-item similarities.

We argue that such many-assessment modeling techniques will likely underperform because of the following characteristics of our context:

- Diverse, heterogeneous tasks: When workers get hired multiple times, they usually complete unique tasks (i.e., tasks requiring different skills and specifications). As a result, the ratings they receive represent their performance across different task requirements (Online Appendix B.1 provides relevant supportive evidence). Hence, the underlying recommended item in our context is not the worker, but, instead, a combination of worker and task characteristics—that is, a *job application*—which can only be experienced once by the employer who posted the focal task.
- Ratio of employers to workers: Our context has a low user-to-item ratio (i.e., employer-to-worker ratio). In many-assessment contexts, the user-to-item ratio is typically large: Netflix, for example, has a user-to-item ratio 27,581 times larger than the ratio of our context (Online Appendix B.1). Higher ratios generate less sparse user-item matrices and allow many-assessment recommender systems to efficiently estimate user-user and item-item similarities. When the available items are significantly more than the available users, many items do not receive any rating, which increases the sparsity of the user-item matrices. This is generally true for online labor markets, where many workers are new or have never gotten hired on the platform before applying to a focal job opening (Pallais 2014, p. 3576, table 1).
- Unique choice sets: In many-assessment contexts, the choice sets have large overlapping segments of product offerings over time. Even though new items enter the choice sets and old items exit frequently (e.g., new products on Amazon or new movies on Netflix), most items remain available for long periods. Conceptually, such overlapping choice sets are important, as they reduce sparsity and facilitate efficient matrix-completion approaches. However, in the focal context, each task attracts a unique pool of job applicants from which employers can choose. Hence, even in the scenario above, where many-assessment approaches assume that a worker is the recommended item, they will have to learn from sparse user-item matrices, which will hurt their overall performance (Online Appendix B.4).

These observations suggest that single-assessment (classification) approaches might perform better in our context, as such approaches use observed *applicant-task* characteristics (Table 1) and do not require multiple ratings or overlapping choice sets to provide meaningful candidate rankings. Online Appendix B provides additional examples and describes in detail additional structural problems that many-assessment frameworks face when applied to the focal context; Table 3 in the online appendix summarizes the similarities and differences between the focal and many-assessment contexts; Figures 10 and 15 in the online appendix show that such approaches indeed underperform in our and similar—for example, restaurant recommendations—contexts.

2.2. Limitations of Existing Recommenders

2.2.1. Do Current Single-Assessment Approaches Provide Sufficient Solutions? Existing approaches rank job applicants according to their likelihood of getting hired (Kokkodis et al. 2015, Abhinav et al. 2017). In addition, classification algorithms proposed in automated recruiters can be adapted to address the focal problem (Färber et al. 2003, Malinowski et al. 2006, Goswami et al. 2014, Mao et al. 2015). However, these current classification approaches have two shortcomings:

1. They assume binary outcomes (i.e., predict Hire or No-hire) that explicitly ignore the performance of the hired worker. As a result, these models learn to uniformly reinforce previously observed behavior that includes unsuccessful hiring choices.
2. They implicitly assume that employer hiring preferences remain the same over repeated hiring choices. However, many employers might evolve and adjust their hiring preferences over time as they gain experience by remotely managing workers and by becoming more familiar with the platform.[2]

We argue that, to achieve better performance, *single-assessment* systems should directly address these two shortcomings by being *performance-aware* and *sequence-aware*. Performance-aware systems identify previously unsuccessful hiring decisions and learn to promote job applicants who are not only hirable, but also likely to perform well. Sequence-aware systems allow employers to adjust their hiring preferences over time and offer recommendations that capture employers' current hiring preferences. Together, these design principles structure frameworks that provide personalized recommendations of job applicants who are likely to get hired and perform well.

3. Sequence-Aware and Performance-Aware Job-Applicant Recommendations

A transaction in online labor markets starts with an employer creating a job opening. The opening description reveals characteristics that the employer is looking for, such as the required set of skills and experience. Workers who are looking for opportunities observe these characteristics and self-select to submit their job applications to openings that they see fit. Employers

Table 1. Descriptive Statistics

	Mean	Median	StD	Min	Max
Variable that creates outcomes Y_{it}					
Worker performance (privately reported to the platform)	0.79	0.89	0.28	0	1
Employer characteristics ($\boldsymbol{X}_{o-1}$)					
Employer money spent after completing $o-1$ tasks	178	4	841	0	3,6806
Employer most recent outcome ($o-1$)	0.62	1	0.49	0	1
Employer total competed tasks ($o-1$)	2.3	1	6	0	131
Employer number of fixed contracts after completing $o-1$ tasks	1.5	0	4.7	0	94
Employer number of hourly contracts after completing $o-1$ tasks	0.84	0	3.1	0	120
Employer total hire-positive outcomes after completing $o-1$ tasks	1.4	0	4	0	104
Employer fixed contract jobs with hire-positive outcomes after completing $o-1$ tasks	0.96	0	3.3	0	89
Employer hourly contract jobs with hire-positive outcomes after completing $o-1$ tasks	0.46	0	2.1	0	99
Job-application characteristics (snapshots of worker profiles at the time of application, $\boldsymbol{Z}_t$)					
Applicant completed work hours	578	45	1,472	0	37,766
Skills IP	1	1	1.2	0	18
Applicant bid price	89	11	549	1	50,000
Received order of application	26	15	32	0	291
Applicant accumulated reputation score (publicly available)	4.8	4.9	0.4	1	5
Applicant completed jobs	5	0	15	0	403
Employer-applicant countries log-lift	−0.49	−0.46	0.4	−3.7	4.4
Applicant's self-reported years of experience (not verified by the platform)	4.5	4	4.1	0	30
Certifications log-lift	2.3	2.1	1.7	−0.71	7
Invited	0.15	0	0.35	0	1
Contract type	0.48	0	0.5	0	1

Notes. Statistics describe time-varying sequential observations. IP, inner product.

then assess the available job applicants and decide which ones to hire. The next paragraphs summarize a single-assessment system that provides sequence-aware and performance-aware rankings of such job applicants, hence guiding employers to make better-informed and faster decisions.

3.1. Latent Hiring Preferences and Observed Outcomes

An employer's hiring-decision process is latent. The market, however, observes the characteristics and outcomes of each job application. Specifically, for each applicant, the employer must first choose whether to hire. If the employer does not hire the applicant at hand, the market observes a No-hire outcome. However, if the employer chooses to hire the applicant at hand, the market eventually (when the task is completed) observes an outcome that describes the worker's performance: A Hire-positive outcome occurs when the performance of the hired worker is satisfactory, whereas a Hire-negative outcome occurs when the performance of the hired worker is not satisfactory. Formally, upon completion of a task, the market observes the employer's decisions on the task's job applicants, which fall in the following set of possible outcomes ($\mathcal{Y}$):[3]

$$\mathcal{Y} = \{\text{“No-hire,” “Hire-negative,” “Hire-positive”}\}. \quad (1)$$

Over time, repeat employers who hire workers on multiple tasks might adjust their hiring preferences as they get more familiar with the challenges of hiring and managing remote workers. A Hidden Markov Model can formally facilitate this possible evolution of hiring preferences. In particular, an HMM allows employers to operate from a latent state that captures their current hiring preferences. As employers hire workers over multiple tasks, they emit task-specific No-hire, Hire-positive, and Hire-negative observations. These observations reveal new information about the current employer's hiring preferences. If the employers' preferences change, the HMM allows for employers to stochastically transition to new states that better capture their updated hiring preferences. Otherwise, employers remain in the state that best describes their observed behavior.

3.2. HMM Structure

The definition of an HMM requires (1) a transition matrix T that describes the transition probabilities between states that capture different employer hiring preferences, and (2) an emission matrix E that describes the state-specific probability distributions across the set of observations $\mathcal{Y}$. In the next paragraphs, we assume that the HMM has K states, such that $\mathcal{S} = \{s_1, s_2, \ldots, s_K\}$; Section 5.3 shows the tuning process of choosing an appropriate number of states K.

3.2.1. Transition Probabilities. Employers emit observations $Y_t \in \mathcal{Y}$ for every job application they receive (Equation (1)). Conceptually, hiring preferences might only change after an employer hires a worker for a given task and observes the hired worker's performance (i.e., at the completion of the task). The HMM encodes this

by allowing state transitions to occur *only when* the employer has:

1. Chosen a job applicant to hire,
2. Chosen job applicants to not hire (No-hire observations), and
3. Observed the outcome of the hired worker (Hire-negative or Hire-positive observation).

This context-specific requirement separates the transition probability matrix of our HMM from all prior HMM designs that allow stochastic transitions to occur *after every observation* (Bishop 2006, Murphy 2012, Sahoo et al. 2012, Kokkodis 2021). (Online Appendix 5.6 shows that this constraint yields significantly better results in practice.) Formally, the task-specific transition probability of a given employer to move from state s_k to state s_l when evaluating job application t for task o after observing the outcomes of $o-1$ tasks is as follows:

$$\lambda^{s_k s_l}_{\boldsymbol{\gamma}_{kl} \boldsymbol{X}_{o-1}} := \Pr(S_t = s_l \mid S_{t-1} = s_k; \boldsymbol{\gamma}_{kl}, \boldsymbol{X}_{o-1}) = \begin{cases} 0, & \text{if } t \text{ is not the first application of task } o \text{ to be evaluated and } s_l \neq s_k \\ 1, & \text{if } t \text{ is not the first application of task } o \text{ to be evaluated and } s_l = s_k, \\ \text{softmax}(\boldsymbol{\gamma}_{kl} \boldsymbol{X}_{o-1}), & \text{if } t \text{ is the first application of task } o \text{ to be evaluated,} \end{cases} \tag{2}$$

where $\boldsymbol{X}_{o-1}$ is a vector of *employer characteristics* that captures the employer's behavior on the previously $o-1$ tasks with observed outcomes, and $\boldsymbol{\gamma}_{kl}$ is a parameter vector of state s_k that weights vector $\boldsymbol{X}_{o-1}$. An alternative way to read Equation (2) is by focusing on indexes t and o: Each job application within a task o increases index t. However, transitions to a new state can only occur when the index o increases to $o+1$, which occurs when task o is completed.

Based on Equation (2), for a job applicant t, the transition matrix has the following form:

$$T(\boldsymbol{\gamma}, \boldsymbol{X}_{o-1}) = \begin{bmatrix} \lambda^{s_1 s_1}_{\boldsymbol{\gamma}_{11} \boldsymbol{X}_{o-1}} & \lambda^{s_1 s_2}_{\boldsymbol{\gamma}_{12} \boldsymbol{X}_{o-1}} & \cdots & \lambda^{s_1 s_K}_{\boldsymbol{\gamma}_{1K} \boldsymbol{X}_{o-1}} \\ \lambda^{s_2 s_1}_{\boldsymbol{\gamma}_{21} \boldsymbol{X}_{o-1}} & \lambda^{s_2 s_2}_{\boldsymbol{\gamma}_{22} \boldsymbol{X}_{o-1}} & \cdots & \lambda^{s_2 s_K}_{\boldsymbol{\gamma}_{2K} \boldsymbol{X}_{o-1}} \\ \vdots & \vdots & \vdots & \vdots \\ \lambda^{s_K s_1}_{\boldsymbol{\gamma}_{K1} \boldsymbol{X}_{o-1}} & \lambda^{s_K s_2}_{\boldsymbol{\gamma}_{K2} \boldsymbol{X}_{o-1}} & \vdots & \lambda^{s_K s_K}_{\boldsymbol{\gamma}_{KK} \boldsymbol{X}_{o-1}} \end{bmatrix}, \tag{3}$$

where $\boldsymbol{\gamma} = [\boldsymbol{\gamma}_{11}, \boldsymbol{\gamma}_{12}, \ldots, \boldsymbol{\gamma}_{KK}]'$.

Unlike most HMMs that provide state-specific static transition matrices (Bishop 2006, Murphy 2012, Sahoo et al. 2012, Hosseinzadeh Aghdam et al. 2015, Zhang et al. 2016), the elements of the focal matrix T of Equation (3) are state-specific, employer-specific, and task-specific. Simply put, the transition probabilities of each employer are personalized, and they change according to the employer's current state and history of observed outcomes, as captured by vector $\boldsymbol{X}_{o-1}$.

3.2.2. Emission Probabilities.

The second component of the HMM framework identifies the emission probabilities across the three hiring outcomes in $\mathcal{Y}$. The HMM assumes that these probabilities are (1) state-specific, representing the current hiring preferences of the employer; and (2) job-application-specific, capturing the observed characteristics of the focal job application. In particular, for a given job application, an employer at state s_k will make a hiring choice according to the following:

$$\mu^{s_k}_{\boldsymbol{\beta}^y_k \boldsymbol{Z}_t \boldsymbol{X}_{o-1}} := \Pr(Y_t = y \mid S_t = s_k; \boldsymbol{\beta}^y_k, \boldsymbol{Z}_t, \boldsymbol{X}_{o-1}) = \text{softmax}(\boldsymbol{\beta}^y_k [\boldsymbol{Z}_t, \boldsymbol{X}_{o-1}]'), \tag{4}$$

where $\boldsymbol{Z}_t$ is a vector of *job-application characteristics*, $\boldsymbol{X}_{o-1}$ is the same vector of employer characteristics that affect transition probabilities (Equation (2)), $y \in \mathcal{Y}, s_k \in \mathcal{S}$, and $\boldsymbol{\beta}^y_k$ is a parameter vector that weights $\boldsymbol{Z}_t$ and $\boldsymbol{X}_{o-1}$ in estimating the probability to observe outcome y when being in state s_k. The emission matrix then is as follows:

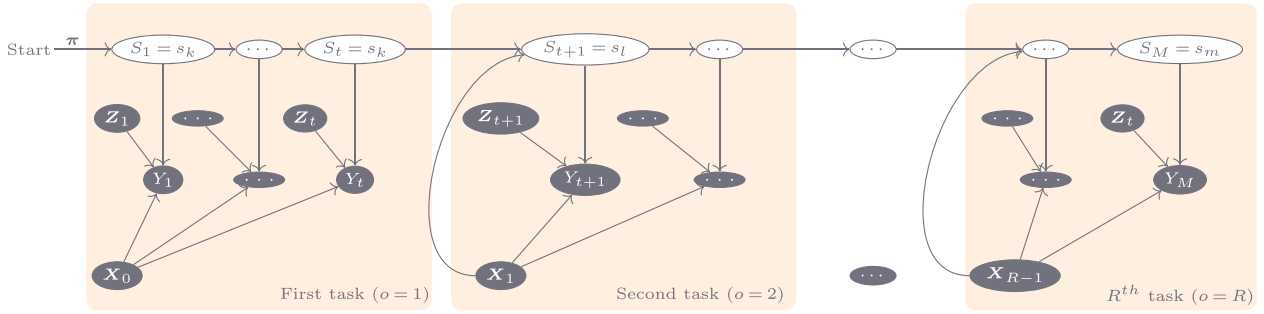
$$E(\boldsymbol{\beta}, \boldsymbol{Z}_t, \boldsymbol{X}_{o-1}) = \begin{bmatrix} \mu^{s_1}_{\boldsymbol{\beta}^{\text{No-hire}}_1 \boldsymbol{Z}_t \boldsymbol{X}_{o-1}} & \mu^{s_1}_{\boldsymbol{\beta}^{\text{Hire-negative}}_1 \boldsymbol{Z}_t \boldsymbol{X}_{o-1}} & \mu^{s_1}_{\boldsymbol{\beta}^{\text{Hire-positive}}_1 \boldsymbol{Z}_t \boldsymbol{X}_{o-1}} \\ \vdots & \vdots & \vdots \\ \mu^{s_K}_{\boldsymbol{\beta}^{\text{No-hire}}_K \boldsymbol{Z}_t \boldsymbol{X}_{o-1}} & \mu^{s_K}_{\boldsymbol{\beta}^{\text{Hire-negative}}_K \boldsymbol{Z}_t \boldsymbol{X}_{o-1}} & \mu^{s_K}_{\boldsymbol{\beta}^{\text{Hire-positive}}_K \boldsymbol{Z}_t \boldsymbol{X}_{o-1}} \end{bmatrix}, \tag{5}$$

where $\boldsymbol{\beta} = [\boldsymbol{\beta}^{\text{No-hire}}_1, \boldsymbol{\beta}^{\text{Hire-negative}}_1, \boldsymbol{\beta}^{\text{Hire-positive}}_1, \ldots, \boldsymbol{\beta}^{\text{Hire-positive}}_K]'$. Similar to the transition matrix T, and unlike popular HMM approaches that provide state-specific user-independent emission distributions (Murphy 2012, Sahoo et al. 2012, Hosseinzadeh Aghdam et al. 2015, Zhang et al. 2016), the emission probabilities of the proposed framework depend not only on the current employer state, but also on both the observed job application $\boldsymbol{Z}_t$ and employer $\boldsymbol{X}_{o-1}$ characteristics.

3.3. HMM Likelihood Derivation and Estimation

Figure 2 shows the evolution of hiring-preferences of a single employer across R tasks that attract M job applications with outcomes $Y_1, \ldots, Y_t, \ldots, Y_M$, $Y_t \in \mathcal{Y}$. Shaded ellipses identify observed outcomes and characteristics, whereas clear ellipses identify latent hiring-preference states. The figure identifies that a new employer who joins the platform starts at state $s_k \in \mathcal{S}$, according to an initial probability vector $\boldsymbol{\pi}$, and remains to that state until the completion of the first task. From that state, the employer chooses which applicant to hire for the first task. Once the task is completed, the platform observes the performance of the hired applicant and annotates observations $Y_1 \in \mathcal{Y}$ to $Y_t \in \mathcal{Y}$. The experience that the employer gained from the first task accumulates to vector $\boldsymbol{X}_1$. This vector affects the transition of the employer to a potentially new state s_l (Equation (2)), which better

Figure 2. (Color online) Interactions of the HMM Framework Across a Sequence of Completed Tasks by a Single Employer



Notes. The figure illustrates a sequence of M hiring choices (No-hire, Hire-negative, and Hire-positive) across R tasks of a single employer. During these choices, the focal employer transitions over states $S_1, S_2, \ldots, S_M$ according to the previously observed employer characteristics $\boldsymbol{X}_0, \boldsymbol{X}_1, \ldots, \boldsymbol{X}_{R-1}$ and parameters $\boldsymbol{\gamma}$ (Equation (2)). For each received job application t, the employer emits a hiring choice Y_t based on the employer's current state S_t, employer characteristics $\mathbf{X}_{o-1}$, job-application characteristics $\boldsymbol{Z}_t$, and parameters $\boldsymbol{\beta}$ (Equation (4)). The employer lands on state $S_1 \in \mathcal{S}$, according to an initial probability vector $\boldsymbol{\pi}$. Latent states form clear ellipses and observed characteristics form shaded ones (Koller and Friedman 2009).

describes the employer's hiring preferences. From that state, the employer evaluates new applications $t+1, \ldots,$ for task $o = 2$. This process continues until the completion of the R^{th} task.

Given the structure of the HMM in Figure 2, we estimate the parameter vectors $\boldsymbol{\pi}, \boldsymbol{\beta}, \boldsymbol{\gamma}$ by maximizing the conditional probability of the set of observations. For a sequence of M hiring decisions across R tasks for a given employer i (Figure 2), we observe:

$$\boldsymbol{Y}_i = Y_{i1}, Y_{i2}, \ldots, Y_{iM},\ Y_{im} \in \mathcal{Y}. \tag{6}$$

These observations correspond to a sequence of employer and job-application characteristics and to a sequence of latent states (Figure 2):

$$\boldsymbol{X}_{i0:R-1} = \boldsymbol{X}_{i0}, \boldsymbol{X}_{i1}, \ldots, \boldsymbol{X}_{iR-1}, \tag{7}$$

$$\boldsymbol{Z}_{i1:M} = \boldsymbol{Z}_{i1}, \boldsymbol{Z}_{i2}, \ldots, \boldsymbol{Z}_{iM}, \tag{8}$$

$$\boldsymbol{S}_i = S_{i1}, S_{i2}, \ldots, S_{iM},\ S_{im} \in \mathcal{S}. \tag{9}$$

Based on the structure of the graph in Figure 2, the conditional likelihood of observing $\boldsymbol{Y}_i$ is:

$$\Pr(\boldsymbol{Y}_i \mid \boldsymbol{S}_i; \boldsymbol{\beta}, \boldsymbol{Z}_{i1:M}, \boldsymbol{X}_{i1:R-1}) = \prod_{t=1}^{M} \Pr(Y_{it} \mid S_{it}; \boldsymbol{\beta}, \boldsymbol{Z}_{it}, \boldsymbol{X}_{io-1}), \tag{10}$$

where Equation (4) estimates the right-hand side.[4] Similarly, the conditional probability of observing the sequence $\boldsymbol{S}_i$ is:

$$\Pr(\boldsymbol{S}_i \mid \boldsymbol{\gamma}, \boldsymbol{X}_{i1:R-1}) = \boldsymbol{\pi}(S_1) \prod_{t=2}^{M} \Pr(S_{it} \mid S_{it-1}; \boldsymbol{\gamma}, \boldsymbol{X}_{io-1}), \tag{11}$$

where $\boldsymbol{\pi}(S_1)$ is the prior probability of being at state $S_1 \in \mathcal{S}$, and Equation (2) estimates the right-hand side.

We can now derive the likelihood of this sequence of observations for employer i from the probabilistic graphical model of Figure 2 (Koller and Friedman 2009) and Equations (10) and (11) as follows:

$$\begin{aligned} & l(\boldsymbol{Y}_i; \boldsymbol{\pi}, \boldsymbol{\beta}, \boldsymbol{\gamma}) \\ &= \Pr(\boldsymbol{Y}_i \mid \boldsymbol{\pi}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{Z}_{i1:M}, \boldsymbol{X}_{i1:R-1}, \mathbf{Y}_{i1:M-1}) \\ &= \boldsymbol{\pi}(S_1) \sum_{\forall \boldsymbol{S}_i} \Pr(\boldsymbol{Y}_i, \boldsymbol{S}_i \mid \boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{Z}_{i1:M}, \boldsymbol{X}_{i1:R-1}, \mathbf{Y}_{i1:M-1}) \\ &\overset{\text{Figure 2}}{=} \boldsymbol{\pi}(S_1) \sum_{\forall \boldsymbol{S}_i} \Pr(\boldsymbol{Y}_i \mid \boldsymbol{S}_i; \boldsymbol{\beta}, \boldsymbol{Z}_{i1:M}, \boldsymbol{X}_{i1:R-1}) \Pr(\boldsymbol{S}_i \mid \boldsymbol{\gamma}, \boldsymbol{X}_{i1:R-1}) \\ &= \boldsymbol{\pi}(S_1) Pr(\boldsymbol{Y}_{i1} \mid S_{i1}; \boldsymbol{\beta}, \boldsymbol{Z}_{i1}, \boldsymbol{X}_{i0}) \\ &\quad \cdot \sum_{\forall \boldsymbol{S}_i} \prod_{t=2}^{M} \Pr(\boldsymbol{Y}_{it} \mid S_{it}; \boldsymbol{\beta}, \boldsymbol{Z}_{it}, \boldsymbol{X}_{io-1}) \\ &\quad \cdot \Pr(S_{it} \mid S_{it-1}; \boldsymbol{\gamma}, \boldsymbol{X}_{io-1}). \end{aligned} \tag{12}$$

The complete likelihood for a data set with N employers is as follows:

$$L(\boldsymbol{\beta}, \boldsymbol{\gamma}) = \prod_{i=1}^{N} l(\boldsymbol{Y}_i; \boldsymbol{\pi}, \boldsymbol{\beta}, \boldsymbol{\gamma}). \tag{13}$$

To maximize this complete likelihood we use numerical solvers (MacDonald 2014, p. 305), such as the L-BFGS-B (Byrd et al. 1995) or the COBYLA.[5] Equation (13) assigns higher weights to repeat employers who make several hiring decisions (i.e., larger M and R values). As a result, it is expected (and desired) that the resulting HMM will perform better across repeat employers, an observation that we empirically support in Section 5.4.

4. Empirical Context

We build and evaluate the proposed framework on a set of real transactions from a major online labor market. The focal data set includes 762,802 hiring decisions by 11,461 employers that led to 45,331 completed tasks and observed performance outcomes. The data set tracks employers for 12 months and includes all their hiring decisions from the time that they joined the platform. The data set also uses the internal log of the marketplace that takes snapshots of each worker's profile at the time of a job application.

Figure 3. (Color online) Model-Free Evidence Suggests that Some Repeat Employers Adjust Their Hiring Behaviors

Notes. Performance scores are private and only observed by the platform; accumulated reputation scores are publicly available; self-reported years of experience include the self-reported experience of the worker on the skills listed (not verified by the platform). CI, confidence interval.

Figure 3 illustrates characteristics of employer behavior in the focal data set, which shows some initial model-free evidence that some employers might adjust their hiring preferences over time. Figure 3(a) shows that as employers gain experience, they hire workers who perform significantly better ($p < 0.05$). Similarly, Figure 3(b) and (c) show that, over time, employers hire workers with higher reputation scores ($p < 0.05$) and lower self-reported expertise ($p < 0.05$). These trends show how employers' average behavior changes over time. However, not all employers are average; our approach allows employers to evolve or not evolve independently, hence better capturing the current status of their hiring preferences. (Online Appendix D provides a more detailed description of the data set and the focal market.)

4.1. Outcomes, Employer, and Job-Application Characteristics

The HMM framework requires outcomes ($Y_t \in \mathcal{Y}$), as well as employer ($\mathbf{X}_{o-1}$) and job-application ($\mathbf{Z}_t$) characteristics (Figure 2).

4.1.1. Target Variable (Y_t). Equation (1) models three outcomes: Hire-positive, Hire-negative, and No-hire. The No-hire outcomes are readily available through the job applicants that employers did not choose to hire. The platform further labels a hiring outcome as Hire-positive when, upon the competition of the task, the employer privately rates the performance of the hired worker with a score greater than or equal to 80% (i.e., performance threshold = 0.8). Otherwise, the platform labels a hiring outcome as Hire-negative. (Online Appendix G illustrates the robustness of our approach across alternative performance thresholds.)

4.1.2. Employer and Job-Application Characteristics ($\mathbf{X}_{o-1}, \mathbf{Z}_t$). Previous works on job-applicant recommendations have proposed various features that capture employer and job-application characteristics to predict the likelihood of each applicant to get hired (Kokkodis et al. 2015, Abhinav et al. 2017). Online Appendix E presents the set of 22 predictive variables we consider for this analysis; Table 1 shows their descriptive statistics. These statistics represent sequential observations of the same variables over time. We log-transform variables with long tails. We normalize (min-max) all variables to accelerate the convergence speed of the numeric optimization. (Note that the variables of Table 1 are context-specific. Online Appendix C shows how alternative contexts—such as restaurant recommendations—require different variable choices.)

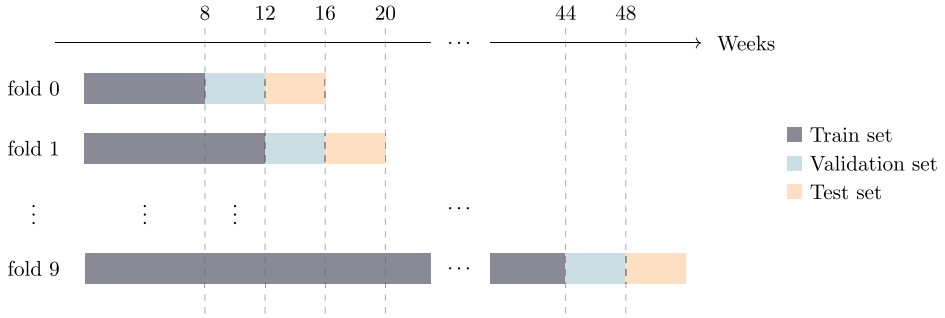
5. Framework Evaluation

The next paragraphs describe the training process of the HMM framework and compare its performance with alternative advanced recommender systems.

5.1. Nested Cross-Validation

Our data have a time component (Figure 2). To ensure that we do not use information from the future to predict the past, we evaluate all approaches through a nested cross-validation setup (see, for instance, Cochrane 2018). Figure 4 describes this process. First, we assume 10 folds. Each fold includes a training, a validation, and a test set. Each training set includes only tasks that have been completed before the beginning of the validation set; each validation set includes only tasks that have been completed before the beginning of the test set. The necessary data transformations (e.g., normalization), hyperparameter tuning, and feature selection[6] happen in each training and validation set, guaranteeing no data leakage in the previously unseen test sets. Once tuning is done, each algorithm uses both the training and validation sets of each fold to build a final model that is tested once on the previously unobserved test set.

Figure 4. (Color online) The Nested Cross-Validation Process



Notes. For all models, parameter tuning and feature selection happens in the training and validation sets. Reported performance across all metrics is estimated on the previously unseen test sets.

5.2. Alternative Recommender Systems

Alternative approaches could also generate rankings of applicants. Prior work on recommending job applicants (Kokkodis et al. 2015, Abhinav et al. 2017) along with other sequence-aware machine-learning approaches provide a variety of alternative single-assessment recommenders. To benchmark the performance of the HMM framework against such advanced alternative models, we implement and compare the following systems:

- *Current reputation:* Upon completion of each job, workers receive a publicly available rating. These ratings accumulate to form each worker's public reputation. The most straightforward and transparent recommender system ranks applicants according to their accumulated reputation scores (Kokkodis et al. 2015, Abhinav et al. 2017).
- *Single-assessment recommenders*: Classification techniques can estimate the likelihood of an applicant getting hired and completing a job successfully. These systems model the relationship:

$$\Pr(Y_t \mid \boldsymbol{Z}_t, \boldsymbol{X}_{o-1}) \sim G(\boldsymbol{Z}_t, \boldsymbol{X}_{o-1}), \tag{14}$$

where:

  - Logistic regression: G represents the logistic sigmoid.
  - Support Vector Machines (SVMs): G captures the relationships between vectors $\boldsymbol{X}_{o-1}, \boldsymbol{Z}_t$, and Y_t through Support Vector Machines.
  - Gradient-boosting classification (XGBoost): G captures the relationships between vectors $\boldsymbol{X}_{o-1}, \boldsymbol{Z}_t$, and $Y_t \in \mathcal{Y}$ through gradient-boosting classification (Chen and Guestrin 2016).
  - Random forest: G captures the relationships between vectors $\boldsymbol{X}_{o-1}, \boldsymbol{Z}_t$, and $Y_t \in \mathcal{Y}$ through a multitude of decision trees (Ho 1998).
  - Recurrent neural networks (LSTM): G captures sequence-aware relationships between vectors $\boldsymbol{X}_{o-1}, \boldsymbol{Z}_t$, and $Y_t \in \mathcal{Y}$ through Long Short-Term Memory (LSTM) networks (Hochreiter and Schmidhuber 1997).

5.3. Hyperparameter Tuning

The proposed framework and many of the alternative recommender systems require hyperparameter tuning:

- HMM: We need to identify the number of states K for each fold in the nested cross validation structure (Figure 4). We consider the following 40 combinations:

$$\text{HMM tested combinations}: \left\{ \overbrace{\{2,3,4,5\}}^{K} \times \overbrace{\{0,1,\ldots,9\}}^{folds} \right\}. \tag{15}$$

For each combination, we follow an HMM-specific step-forward feature selection process (Ferri et al. 1994) to identify the best-performing predictive variables on the validation set.

- Random forest: We use the Python package sklearn.ensemble.RandomForestClassifier. We experiment with two hyperparameters: the maximum depth of each decision tree ("max_depth") and the number of trees in the forest ("n_estimators"). We consider the following 90 combinations:

$$\text{Random forest tested combinations}: \left\{ \overbrace{\{3,10,15\}}^{\texttt{max\_depth}} \times \overbrace{\{10,50,100\}}^{\texttt{n\_estimators}} \times \overbrace{\{0,1,\ldots,9\}}^{folds} \right\}. \tag{16}$$

Similar to the HMM training process, for each one of these combinations, we follow a random-forest-specific step-forward feature-selection process to identify the best-performing predictive variables on the validation set.

- XGBoost: We use the Python package xgboost. We tune three hyperparameters: the number of trees to fit ("n_estimators"), the maximum tree depth ("max_depth"), and the subsample ratio of the training instance ("subsample"). We consider the following 180 combinations:

$$\text{XGBoost tested combinations}: \left\{ \overbrace{\{50,100,150\}}^{\texttt{n\_estimators}} \times \overbrace{\{3,10,15\}}^{\texttt{max\_depth}} \times \overbrace{\{0.8,1\}}^{\texttt{subsample}} \times \overbrace{\{0,1,\ldots,9\}}^{folds} \right\}. \tag{17}$$

For each one of these combinations, we follow an XGBoost-specific step-forward feature-selection process

to identify the best-performing predictive variables on the validation set.

- LSTM: We use the Python packages `keras.models.Sequential and keras.layers.LSTM`. To get probability estimates, we use a `softmax` activation function, and we optimize according to the `categorical_crossentropy`. We tune two hyperparameters: the number of `"epochs"` to train the model, and the number of samples per gradient update `"batch_size."` In addition, we explore stacking hidden LSTM layers, in an effort to improve performance (Brownlee 2017). We consider the following 180 combinations:

$$\text{LSTM tested combinations}: \Big\{ \overbrace{\{10, 20, 30\}}^{\texttt{epochs}} \times \overbrace{\{32, 64, 128\}}^{\texttt{batch\_size}} \times \{\text{Stacked, Not stacked}\} \times \overbrace{\{0, 1, \ldots, 9\}}^{folds} \Big\}. \quad (18)$$

To set the parameter `"units"` (i.e., the dimensionality of the output space of the hidden layers; dense-layer implementation of Keras 2021), we use the formula (Eckhardt 2018):

$$\texttt{units} = 0.67 * (n_{features} + n_{steps}), \quad (19)$$

where $n_{features}$ is the total number of predictive variables and n_{steps} is the length of the sequence (see also Online Appendix L). Our final layer has three neurons that capture the total number of output classes we predict. Similar with the previous models, for each one of these combinations, we follow an LSTM-specific step-forward feature-selection process to identify the best-performing predictive variables on the validation set.

5.4. Results

Our goal is to rank job applicants according to their likelihood of getting hired and performing well. The next paragraphs benchmark the performance of the HMM framework against alternative recommender systems across four ranking measures:

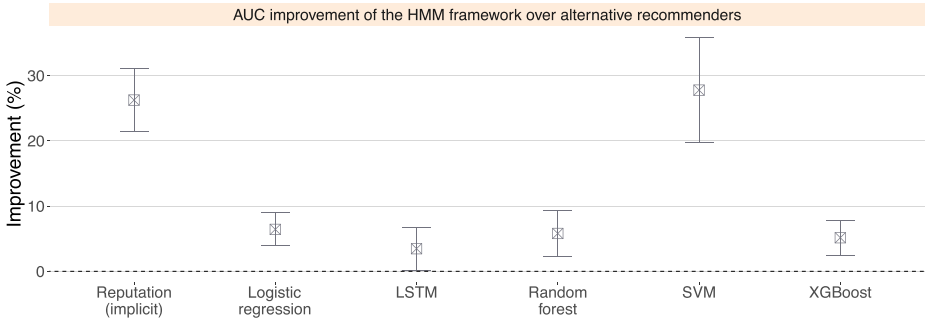
- Job-applicant rankings for all employers,
- Job-applicant rankings for repeat employers,
- Performance of top-ranked applicants,
- Performance of top-ranked applicants within tasks.

5.4.1. Job-Applicant Rankings for All Employers.

The first ranking measure that we use is the area under the receiver operating characteristic curve (AUC; see Provost and Fawcett 2001). Our AUC score of interest measures the probability that a model ranks applicants who are hirable and likely to perform well (Hire-positive) higher than applicants who are not likely to get hired (No-hire) or who are likely to get hired and perform poorly (Hire-negative; see p. 864 of Fawcett 2006). (Online Appendix F shows alternative rankings that trade-off Hire-positive with Hire-negative outcomes.)

Figure 5 compares the AUC performance of the HMM framework with that of the alternative recommender systems (Table 2 shows the average nested 10-fold cross-validated AUC scores for each approach). The y-axis shows the average percentage AUC improvement of the HMM framework compared with the x-axis alternative system (Table 4 in Online Appendix J shows the AUC scores for each fold and approach). The error bars show the nested 10-fold cross-validated 95% confidence intervals. The figure shows that the performance of the HMM framework is significantly ($p < 0.05$) better than the performance of all alternative systems. The percentage improvement over the existing single-assessment models (Kokkodis et al. 2015, Abhinav et al. 2017) ranges, on average, between 4% and 28%. The most competitive (with the HMM framework) models are the LSTM, and XGBoost, neither of which has been previously proposed for job-applicant recommendations. Yet, even against these powerful approaches, the unique structure that allows task-specific transitions of our HMM framework allows it to perform on average 4%–6% ($p < 0.05$) better. As we discuss next, the outperformance of our HMM follows an increasing trend when models recommend job applicants to repeat employers.

Figure 5. (Color online) Comparison of Alternative Recommenders: Job-Applicant Rankings for All Employers



Notes. The proposed approach ranks job applicants according to their likelihood of getting hired and performing well significantly better (at least $p < 0.1$) than the alternative systems. The y-axis shows the 10-fold nested cross-validated AUC percentage improvement of the proposed framework over the x-axis recommender systems. Error bars show 95% confidence intervals. Implicit identifies implicit-feedback systems (No-hire or Hire). Confidence intervals are estimated across the improvements of the 10 folds.

Table 2. Model Performance

Model	AUC
HMM	0.710
Logistic regression	0.668
Random forest	0.673
SVM	0.561
XGBoost	0.677
LSTM	0.688
Reputation (implicit)	0.564

Note. Average nested 10-fold cross-validated AUC scores for each approach.

5.4.2. Job-Applicant Rankings for Repeat Employers.

The real power of the HMM approach resides in modeling employers who hire workers repeatedly over different tasks, as these employers are more likely to adjust their hiring preferences and evolve across the HMM states (see also Online Appendix I, which quantifies employer transitions). To test how each approach performs in terms of such *repeat employers*, we estimate the AUC scores for hiring decisions after employers complete n or more tasks (AUC-n). Specifically, for any given n, we consider only hiring decisions that occurred after each decision-making employer has hired and evaluated $n-1$ workers. For instance, if a given employer appears in a test set with a sequence of four tasks and their respective hiring choices and outcomes, then AUC-2 would consider that employer's second, third, and fourth tasks' choices; AUC-3 would consider the third and fourth tasks' choices; and AUC-4 would consider the fourth task's choices; for $n>4$, the estimation of AUC-n would exclude this employer.

Figure 6 shows the 10-fold cross-validated AUC-n percentage improvement scores of the HMM approach over the alternative recommender approaches as employers hire workers across different job tasks. The x-axis captures the number of previously hired workers (completed collaborations) for each employer. Intuitively, as employers hire and manage more workers, the HMM should provide better recommendations, as its state structure allows employers to adjust their hiring preferences individually. As a result, the HMM improvement over the alternative recommenders should increase over time.

Figure 6 shows this expected improvement increase over time. Across all alternative recommender systems, the slope of the linear regression of percentage improvement over the number of completed jobs is positive (at least $p<0.1$; Figure 6 shows the 95% confidence intervals for the slopes of each improvement line; for the reputation baseline, the slope is positive, but not statistically significant). This increasing trend illustrates that allowing employers to evolve according to the feature vector $\mathbf{X}_{o-1}$ and Equation (2) captures employer changing hiring preferences and yields progressively better results compared with the alternative recommender systems that do not capture employer evolution as accurately. (Note that sequence-aware models such as LSTM do not encode the task-specific sequences of Equation (2); instead, the sequences these systems model identify sequential dependencies across all hiring outcomes: No-hire, Hire-negative, and Hire-positive. (Online Appendix L discusses and conceptually explains why our HMM approach outperforms the LSTM recommender.)

5.4.3. Performance of Top-Ranked Applicants.

Given a ranking of candidates, perhaps the most crucial performance metric is whether the top-ranked candidates actually get hired and perform better than the bottom-ranked ones. To evaluate this behavior, we compare all

Figure 6. (Color online) Comparison of Alternative Recommenders: Job-Applicant Rankings for Repeat Employers

Notes. As employers hire more workers, the proposed approach's improvement over the alternative recommender systems increases (positive slope). This is due to the task-specific employer transitions of Equation (2). The y-axis shows the 10-fold nested cross-validated AUC percentage improvement of the proposed framework over each alternative recommender system. The x-axis captures employer experience in terms of hired workers (completed tasks). CI stands for confidence interval. Implicit identifies implicit-feedback systems (No-hire or Hire). Confidence intervals are estimated across the improvements of the 10 folds.

alternative approaches by estimating the performance lift as follows:

$$\text{Performance lift } (p) = \frac{\#(\text{HMM "Hire-positive"} \in p) - \#(\text{Alternative "Hire-positive"} \in p)}{\#(\text{Alternative "Hire-positive"} \in p)}, \quad (20)$$

where "Alternative" captures a ranking by an alternative recommender system and $p \in \{\text{Bottom 50\%, Top 50\%}\}$.

Intuitively, as we move from the bottom-ranked to the top-ranked job applicants, the performance lift should increase. If ranked applicants by the HMM framework purely outperform ranked applicants from an alternative approach, the estimated performance lift should be negative for the bottom 50th percentile and positive for the top 50th percentile.

Figure 7 shows exactly this behavior for all alternative recommenders. The y-axis shows the nested 10-fold cross-validated performance lift; the x-axis separates applicants into the bottom and the top 50^{th} percentiles, according to their predicted likelihood of getting hired and performing well. Indeed, the observed performance lift is negative ($p < 0.05$) for all bottom-ranked job applicants and positive ($p < 0.05$) for all the top-ranked ones. Overall, compared with all alternative recommenders, the HMM framework presents significantly ($p < 0.05$) better applicants in the top-ranking positions and significantly worse applicants in the bottom ones.

5.4.4. Performance of Top-Ranked Applicants within Tasks.

The ranking (AUC, AUC-n) and performance-lift evaluations capture a model's behavior across all available tasks; they do not, however, evaluate how each algorithm performs within tasks. To do so, we rank applicants within each task, and we measure the actual performance of the top-ranked applicants as follows:

$$\text{Within-task hired-applicant performance}(k) = \frac{\#(\text{"Hire-positive"} \in \text{top–}k \text{ recommended})}{\#(\in \text{top–}k \text{ hired})} \Big/ \frac{\#(\text{"Hire-positive"} \notin \text{top–}k \text{ recommended})}{\#(\notin \text{top–}k \text{ hired})}, \quad (21)$$

where "$\in$ top–k" captures recommended applicants in the top–k who got hired and "$\notin$ top–k" captures recommended applicants not in the top–k who get hired. Conceptually, this performance ratio measures how much better the top-recommended job applicants perform compared with the rest of the non-top-ranked applicants who got hired.

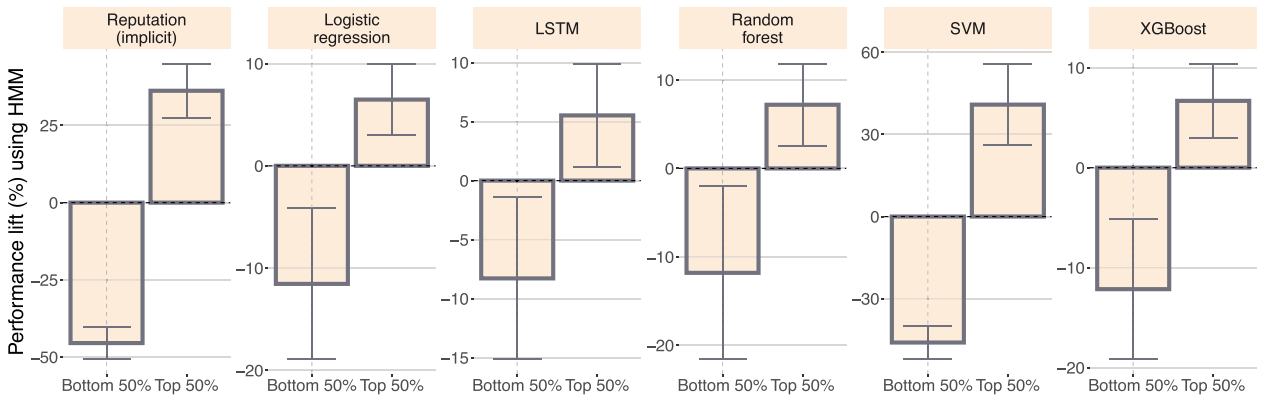
Figure 8 shows the results for $k = 3^7$ in a similar form to Figure 5: The y-axis captures the 10-fold cross-validated performance improvement of the HMM framework over the x-axis recommender. The 95% confidence intervals clearly show that the within-task performance of the HMM framework is statistically significantly ($p < 0.05$) better than all alternative recommender systems. The improvement ranges from an average of 15% over the LSTM to 43% over the SVM.

5.5. Complexity of the Proposed Framework

One potential concern about our approach is that it might be too expensive in terms of training time, as maximizing the likelihood function of Equation (13) requires the estimation of several parameters. In particular, the total number of parameters to be estimated depends on the number of states (n_{states}), the number of outcomes ($n_{outcomes}$), and the number of emission ($n_{emissions}$) and transition ($n_{transitions}$) variables:

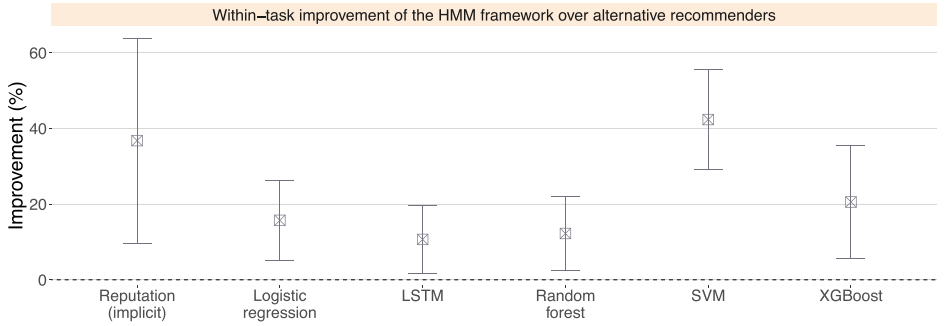
$$\text{Total number of parameters} = n_{states} * (n_{states} - 1) * n_{transitions} + n_{states} * n_{emissions} * (n_{outcomes} - 1) + n_{states}. \quad (22)$$

Figure 7. (Color online) Comparison of Alternative Recommenders: Performance of Top-Ranked Applicants



Notes. Compared with alternative recommenders, the proposed approach ranks applicants who are more likely to get hired and perform well in the top 50th percentiles, whereas it ranks applicants who are less likely to get hired and perform well in the bottom 50th ones. The y-axis shows the 10-fold nested cross-validated performance lift (Equation (20)). Error bars show 95% confidence intervals. Implicit identifies implicit-feedback systems (No-hire or Hire). Confidence intervals are estimated across the improvements of the 10 folds.

Figure 8. (Color online) Comparison of Alternative Recommenders: Performance of Top-Ranked Applicants within Tasks



Notes. The proposed approach ranks job applicants within tasks according to their likelihood of getting hired and performing well significantly better ($p < 0.05$) than the alternative systems. The y-axis shows the nested 10-fold cross-validated within-task performance (Equation (21)) improvement of the proposed framework over the x-axis recommender systems. Error bars show 95% confidence intervals. Implicit identifies implicit-feedback systems (No-hire or Hire). Confidence intervals are estimated across the improvements of the 10 folds.

Hence, the parameters that we estimate for our main models range from 42 to 410.

Estimating these parameters through the numeric optimization presented in Section 3.3 is indeed computationally tedious. In fact, directly maximizing Equation (13) takes a significant amount of time, even for small amounts of data and parameters. In practice, however, we maximize a vectorized version of this likelihood function that significantly reduces the running-time complexity of the estimation process.[8]

In particular, the run-time complexity of the vectorized algorithm is $O(N * M)$, where N is the number of employers (timelines) in our data, and M is the maximum length of a timeline (number of received job applications across completed tasks). In cases such us ours, where M is small and the average length of a timeline is significantly smaller (i.e., less than 10 tasks per employer), the amortized complexity grows almost linear with the number of timelines $\approx O(N)$. And because employer timelines are independent, running time can even be reduced to a constant $\approx O(1)$ through parallelization.[9]

5.6. Comparison of Alternative Transition Constraints

One of the distinct characteristics of our framework is that it allows transitions only after the completion of a task (Equation (2)). *Does this matter?*

We compare our task-specific approach with an HMM that allows task-independent transitions after every observation $Y_{t-1} \in \mathcal{Y}$. Specifically, we can assume that transitions have the following form:

$$\lambda^{s_k s_l}_{\gamma_{kl} \boldsymbol{X}_{o-1} Y_{t-1}} := \Pr(S_t = s_l \,|\, S_{t-1} = s_k; \gamma_{kl}, \boldsymbol{X}_{o-1}) = \text{softmax}(\gamma_{kl} \boldsymbol{X}_{o-1}). \quad (23)$$

We observe that constraining transitions after the completion of a task yields up to 4% significantly ($p < 0.001$) better results. Hence, our unique design choice to model transitions through Equation (2) significantly improves the performance of our approach.

5.7. Robustness and Generalizability

Multiple online appendices illustrate the robustness and generalizability of the HMM framework:

- **Alternative ranking mechanisms:** The proposed approach ranks job applicants according to their likelihood of getting hired and performing well; it ignores, however, the likelihood of each candidate to get hired and perform poorly (Hire-negative). Given that employers who hire poor-performing workers are likely to exit the market (Tripp and Grégoire 2011), alternative ranking mechanisms could trade off Hire-positive outcomes for fewer Hire-negative ones. Such approaches could minimize the likelihood of Hire-negative outcomes, while keeping the likelihood of observing Hire-positive outcomes at sufficient levels. Online Appendix F presents such alternative ranking approaches.
- **Robustness to alternative thresholds:** The platform's choice of performance threshold equal to 0.8 (80%) might appear ad hoc. Online Appendix G illustrates that our results are robust across alternative thresholds of separating Hire-positive from Hire-negative outcomes.
- **Generalizability:** Our approach can generalize to other single-assessment contexts (Figure 1). One such context is restaurant recommendations on reputation platforms, such as TripAdvisor and Yelp. In these platforms, user preferences evolve as reviewers grow older. At the same time, restaurants change significantly, as they go through renovations, update their menus, and hire new staff. As a result, both the users (reviewers) and the recommended items (restaurants) change. In this context, we can build recommender systems that rank restaurants within a location according to their likelihood of getting reviewed positively (i.e., a user will visit them, self-select to review them, and review them positively). Online Appendix C implements our approach and compares its

performance with alternative recommender systems on a set of TripAdvisor restaurant reviews: The proposed approach significantly outperforms ($p < 0.05$) alternative recommender systems, providing evidence that our framework generalizes in contexts where both the recommended items and the user preferences might change over time.

- **Comparison with many-assessment recommenders:** Section 2.1 illustrated the characteristics of job-applicant recommendations that make it a single-assessment context. Online Appendix B provides additional details and examples of why many-assessment systems will likely underperform in this context. Online Appendix B.3 implements eight such systems and illustrates in practice that they underperform compared with our proposed framework (Figure 10); Figure 15 shows that their underperformance extends in the alternative, restaurant -recommendation context.

6. Discussion

This work argued that job-applicant recommenders in online labor markets should be single-assessment systems that are performance-aware and sequence-aware. Based on these principles, an HMM framework modeled performance-aware emissions and allowed employer hiring preferences to change. The empirical evaluation further showed that repeat employers benefit the most from our framework, as these employers received personalized sequence-aware recommendations by following their distinct hiring-preference paths. Application of the HMM framework in a restaurant-recommendation context showed its generalizability in environments where both the recommended items and the user preferences change.

6.1. Research Contributions

Given the projected growth of the number of online workers in the coming years (Agile-1 2016, Sundararajan 2016), accurate job-applicant recommendations could be a significant factor in the ultimate reach of online work. This paper is the first to outline the limitations of existing recommender systems and explain why such systems underperform when ranking job applicants according to their likelihood of getting hired and performing well. By identifying and addressing these shortcomings, this work provides significantly enhanced recommendations of hirable and capable job applicants, especially for repeat employers.

From a design perspective, this work conceptualizes three principles that job-applicant recommender systems should have. First, they need to be *single-assessment* systems and facilitate the modeling of uniquely recommended items. Second, they need to be *performance-aware;* Our work is the first to formulate the job-application recommendation problem as a trinary classification problem that includes both implicit (the choice to hire) and explicit (performance of the hired worker) feedback. Because this formulation separates successful from unsuccessful collaborations, it does not reinforce all prior hiring decisions, but, instead, it learns from unsuccessful collaborations and identifies job applicants that have a high propensity of performing well. Third, job-applicant recommender systems should be *sequence-aware*, allowing employer hiring preferences to evolve over repeated collaborations with remote workers. Our work is the first to capture employers changing hiring preferences through task-specific customized transition probabilities (Equation (2)).

The unique design of the HMM framework extends the rich literature of existing recommender systems. Compared with previous HMM-based systems (Sahoo et al. 2012, Hosseinzadeh Aghdam et al. 2015, Zhang et al. 2016), the proposed approach allows (1) the modeling of uniquely evaluated items, (2) task-specific transition probabilities (Equation (2)), (3) history-driven stochastic transitions (affected by vector $\mathbf{X}_{t-1}$), and (4) item-driven emission probabilities (affected by vectors $\mathbf{X}_{t-1}, \mathbf{Z}_t$). Two empirical contexts (job-applicant and restaurant recommendations) show the significance of these unique characteristics (Sections 5.4 and 5.6, and Online Appendices C and K).

The conceptualizations of performance awareness and sequence awareness can transfer to other types of recommender systems in online labor markets and crowdsourcing. In particular, both automated recruiters and task recommenders could adapt to be performance-aware and sequence-aware. Automated recruiters could include observed outcomes on top of hirability requirements, while allowing both the worker's experience and the employer's hiring preferences to evolve. Similarly, task recommenders can incorporate performance when they allocate jobs to available workers, while they can also allow worker abilities to evolve. Future work on these two types of recommender systems can instill these ideas that will likely improve performance and reduce adverse outcomes.

6.2. Contributions to Practice and Generalizability

This paper provides a detailed guideline along with sample code[10] for market practitioners that are interested in developing performance-aware and sequence-aware recommender systems. Specifically, it addresses empirical challenges that include the conceptualization, modeling, and estimation of the framework:

- HMM architecture: Section 3 describes how practitioners can conceptualize and formulate a suitable structure for an HMM that allows a series of observed signals to shape the transition and emission probabilities of employers with different hiring preferences. It

further illustrates how transitions can be task-specific (Equation (2)), which is conceptually sound when modeling evolving hiring-preferences.

- Parameter estimation: Section 3.3 guides practitioners through the derivation of the global likelihood of the model and the estimation process of all the parameters. Section 5.3 presents the process of selecting an appropriate configuration for the HMM.
- Evaluation: Section 5.4 guides practitioners in generating meaningful evaluation metrics that compare the performance of various recommender systems in terms of ranking candidates according to their likelihood of performing well.

These methodological contributions generalize beyond the focal context of online work. The HMM framework can be adjusted and successfully implemented in any context where (1) recommended items change or receive very few user evaluations, (2) user preferences evolve, and (3) choice sets do not significantly overlap. Offline job-applicant recommendations form one such context: By analyzing career trajectories of LinkedIn users, sequence-aware frameworks can identify performance outcomes (e.g., through "upward trajectory" or "downward trajectory"), while allowing employer hiring preference to change as job requirements change. Recommending restaurants is another such context: Online Appendix C shows that reputation platforms such as Yelp and TripAdvisor could use a similar framework to develop performance-aware and sequence-aware restaurant recommendations. Similarly, travel platforms such as Airbnb can also use the proposed framework as both the recommended items (rented apartments) and user preferences evolve: Our approach models these sequential changes while controlling for implicit (choice of an apartment in which to stay) and explicit (rating of the chosen apartment) feedback. Other suitable contexts include recommending skills to learn (both recommended skills and users change over time) and courses to take (courses evolve based on teachers and year; students also evolve).

6.3. Implications for Platforms, Workers, Employers, and the Future of Work

Online labor markets stand to benefit through implementing our approach as job-applicant recommendations that lead to successful outcomes help (1) workers to differentiate, (2) employers to make better-informed and faster (reduced search cost; see Bakos 1997) decisions, and (3) the markets to increase their transaction efficiency, which, in turn, results in increased revenue and customer satisfaction. Through recommendations of appropriate job applicants, some low-quality workers that currently flood the market might get marginalized. This marginalization can create room for potentially high-quality workers to pursue tasks that they see fit.

Furthermore, employers who make faster decisions that lead to productive collaborations are more likely to keep using the platform (Jerath et al. 2011). More successful employers will create more job openings, which, in turn, will attract more workers and widen the reach of the online labor economy. Besides, through improved recommendations, markets will increase their transaction efficiency as more openings will reach contracts (recall that, currently, as many as 60% of job openings never reach a contract; see Zheng et al. 2015).

The performance of the proposed approach in terms of recommending candidates to repeat employers (Section 5.4 and Figure 6) is of particular importance to market managers. These employers represent a significant client segment for online labor markets. Providing them with relevant recommendations reduces the number of adverse outcomes and increases the employers' likelihood to keep participating in the marketplace (Tripp and Grégoire 2011).

To quantify this market effect of our approach, consider the analysis in Section 5.4.4 that shows a within-task performance improvement between 15% and 43%. If we regress the annual money that employers spend on the most frequent outcome that they observe in the first two months after joining the platform, we find that employers who experience mostly Hire-positive outcomes end up spending on average \$288 ($p < 0.05$) more than those who experience mostly Hire-negative outcomes. Of course, we cannot fully attribute this difference to the observed outcomes. However, for the following calculations, let us assume that 40% of this annual spending can indeed be attributed to observing positive outcomes. This suggests that the 15% improvement of our approach will yield an expected per employer annual revenue increase of $0.15 \times 0.4 \times \$288 = \$17.3$. Assuming 1 million employers (online labor markets typically have several million employers; Upwork 2014), this generates an expected additional annual revenue of \$17.3 million (M). If the improvement is instead 43%, then the expected additional annual revenue will be \$49.5M.

6.4. Further Discussion and Limitations

Next, we review how our work compares with recent works on hiring choices and worker performance in online labor markets, and we conclude by discussing some of the limitations of our approach.

6.4.1. Comparison with Recent Works. Several recent works focus on designing more efficient online markets (Horton and Chilton 2010, Moreno and Terwiesch 2014, Hong et al. 2015, Chen and Horton 2016, Kokkodis and Ipeirotis 2016, Filippas et al. 2018, Kokkodis 2019, Kokkodis et al. 2020, Kokkodis and Ipeirotis 2021, Kokkodis 2022, Kokkodis et al. 2023). Two recently published papers (Kokkodis 2021, Kokkodis and Ransbotham 2022) share methodological similarities with our work. Despite these similarities, this paper is unique, as it solves a different problem with a distinct approach. Below, we discuss

some of the most important differences between these works.[11]

Kokkodis (2021) focuses on estimating a worker's reputation across a set of arbitrary skills. On the contrary, this work provides a framework that ranks job applicants (after workers apply for a job) according to their likelihood of getting hired and performing well. The two papers focus on two distinct aspects of an online market: reputation systems (Kokkodis 2021) and recommender systems (this work). As a result, the reputation system provided in Kokkodis (2021) can generate reputation scores that can be used as features in the recommender system of our work.

Methodologically, both approaches use Hidden Markov Models. However, the two approaches are fundamentally different in terms of (1) hidden state representation (a *worker*'s true ability versus an *employer*'s unobserved hiring preferences); (2) framework structure (Kokkodis 2021 requires several HMMs, whereas the focal work uses a single HMM for all employers); (3) outcomes (Kokkodis 2021 models outcomes through continuous distributions, whereas in this work, the outcome is categorical and, as a result, is being modeled through a multinomial distribution); (4) training data (in Kokkodis 2021, the HMMs are trained only on *completed tasks* (hired workers), whereas this work requires training on every job application to provide meaningful candidate rankings); (5) state transitions (in Kokkodis 2021, transitions are unconditional, whereas in this work, transitions occur only when an employer completes a task—see Section 5.6); and (6) predictive variables (different feature engineering is needed to model worker reputation in Kokkodis 2021 compared with predicting hiring probabilities in this work).

Similarly, the recent work by Kokkodis and Ransbotham (2022) focuses on understanding how employers evolve and learn to make successful hiring choices in online labor markets over time. The paper hypothesizes and empirically shows that many employers tend to be initially overconfident. However, over time, they learn to calibrate their confidence levels, and, by relying more on worker reputation scores and pricing signals, they learn to make more successful hiring choices. On the contrary, this work focuses on ranking job applicants to facilitate all employers' (i.e., both experienced and new) hiring choices. As a result, this work does not attempt to perform a causal analysis (Kokkodis and Ransbotham 2022) and explain why employers behave the way they do. Instead, it tries to learn from previous unsuccessful and successful behaviors to provide better recommendations.

In their work, Kokkodis and Ransbotham (2022) use a panel data set and present a fixed-effects analysis along with instrumental variables, selection models, and other econometric techniques to identify the causal links of how employers learn. As a result, they only use simple HMMs as a clustering technique to place employers into "Less," "More," and "Most" successful buckets. They then use those buckets as variables in their econometric analysis. Hence, because the two works have different objectives, their respective HMMs solve different problems and are methodologically distinct. Specifically, the proposed HMM approach differs from the one in Kokkodis and Ransbotham (2022) in terms of (1) hidden state representation (employer success versus employer unobserved hiring preferences), (2) outcomes (Kokkodis and Ransbotham 2022 model outcomes through continuous distributions, whereas in this work, the outcome is categorical, and, as a result, it is being modeled through a multinomial distribution); (3) training data (in Kokkodis and Ransbotham 2022, the HMMs are trained only on *completed tasks*, whereas this work requires training on every job application to provide meaningful candidate rankings), (4) state transitions (in Kokkodis and Ransbotham 2022, transitions are unconditional, whereas in this work, transitions occur only when an employer completes a task—see Section 5.6), and (5) predictive variables (different feature engineering is needed to model employer success in Kokkodis and Ransbotham 2022 compared with predicting hiring probabilities).

Overall, neither the proposed framework of Kokkodis (2021) nor the one of Kokkodis and Ransbotham (2022) can provide relevant candidate recommendations, as they were built to solve different problems, and, hence, they are structured to work under different assumptions, with different data and different outcomes.

6.4.2. Limitations. The proposed approach primarily focuses on repeat employers, who make several hiring decisions over time. In fact, for one-time employers, the performance of our complex methodology reduces to that of logistic regression. However, on repeat employers, we expect our approach to perform better than models that are not sequence-aware, independent of the individual employer characteristics (Online Appendix H). In addition, we expect our approach to effectively identify robust and high-quality workers, hence avoiding recommending high-variance, unreliable, or low-quality ones. This expectation is in line with the fact that online labor markets tend to have an abundance of workers, so much so that Upwork recently purged 1.8 million of them.[12]

Furthermore, our approach learns to reinforce previously observed successful hiring behavior. As a result, it can miss applicants with characteristics that might be really good and valuable, but have not been previously chosen by employers (i.e., No-hire choices). Markets can explore this possibility by developing rankings that assign a higher weight to No-hire outcomes (similar to the examples presented in Online Appendix F). By

introducing more No-hire candidates into the curated top-ranked applicant recommendations, platforms can empirically measure the outcomes of hired workers who would (probabilistically) not have been hired otherwise, retrain their systems, and eventually learn to provide more holistic and better recommendations.

6.5. Conclusion

Conclusively, this work implements a single-assessment job-applicant recommendation framework that is both performance-aware and sequence-aware. Application of this framework in a large data set of hiring decisions from an online labor market shows that it provides recommendations of job applicants who are both hirable and likely to perform well. The framework's structure generalizes to other contexts where the recommended items evolve or receive very few evaluations and where user preferences change over time. As a result, its deployment in different types of online platforms could have positive impact for workers, employers, businesses, and the future of work.

Endnotes

[1] Note that both types of systems allow users to have different preferences across products. The distinction between the two types originates from whether an *item* can be experienced repeatedly (Figure 1).

[2] Employer hiring preferences might evolve over repeated hiring choices for multiple reasons. Some employers might learn through "trial and error," whereas others might adjust their confidence levels in managing and controlling workers remotely (Kokkodis and Ransbotham 2022). Identifying the exact mechanism through which employers learn or gain experience requires a deeper investigation that is outside the scope of this work. Instead, in this work, we argue that job-applicant recommenders should capture these possibilities by allowing employer hiring preferences to evolve.

[3] Section 4 discusses performance thresholds that separate Hire-positive from Hire-negative outcomes. Extensions to more granular outcomes are conceptually trivial, but increase sparsity without a clear benefit for the platform, as the platform's ultimate goal is to predict the likelihood of getting hired and being successful, which is sufficiently captured by the three classes in $\mathcal{Y}$.

[4] Recall that subscript o is related to subscript t (Figure 2).

[5] In practice, we numerically minimize the negative log-likelihood of Equation (13). See Section 5.5 for additional implementation details.

[6] For all models, we use a step-forward feature-selection process (Ferri et al. 1994). In particular, we use the Python package `mlxtend.feature_selection.SequentialFeatureSelector`, and we build each respective model by choosing the most informative predictive variables among the ones presented in Online Appendix E.

[7] Results are robust for $k = 4, 5$.

[8] See https://github.com/mkokkodis/gbu.

[9] The log of Equation (13) that forms the negative log-likelihood is a sum of individual user likelihoods. Hence, we can estimate the individual user-likelihoods in parallel and aggregate them at the end. We can do this through batches assigned in parallel cores or through a MapReduce formulation.

[10] See https://github.com/mkokkodis/gbu.

[11] Section 2 and Online Appendices A and B provide a broader overview of how this paper fits within the relevant literature.

[12] See https://en.wikipedia.org/wiki/Upwork.

References

Abhinav K, Dubey A, Jain S, Virdi G, Kass A, Mehta M (2017) Crowdadvisor: A framework for freelancer assessment in online marketplace. *2017 IEEE/ACM Internat. Conf. Software Engrg.* (IEEE, Piscataway, NJ), 93–102.

Adomavicius G, Tuzhilin A (2005) Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Trans. Knowledge Data Engrg.* 17(6):734–749.

Agile-1 (2016) Gig economy. Accessed November 28, 2020, http://www.hrotoday.com/wp-content/uploads/2016/07/Whitepaper_Agile2016-single.pdf.

Akerlof GA (1970) The market for "lemons": Quality uncertainty and the market mechanism. *Quart. J. Econom.* 84:488–500.

Atkins O (2019) PeoplePerHour plans on growing community with cross-media campaign. Accessed December 8, 2019, https://www.thedrum.com/news/2019/10/23/peopleperhour-plans-growing-community-with-cross-media-campaign.

Autor DH (2001) Wiring the labor market. *J. Econom. Perspect.* 15:25–40.

Ba S, Pavlou PA (2002) Evidence of the effect of trust building technology in electronic markets: Price premiums and buyer behavior. *MIS Quart.* 26(3):243–268.

Baba Y, Kinoshita K, Kashima H (2016) Participation recommendation system for crowdsourcing contests. *Expert Systems Appl.* 58:174–183.

Bakos Y (1997) Reducing buyer search costs: Implications for electronic marketplaces. *Management Sci.* 43:1676–1692.

Bishop CM (2006) *Pattern Recognition and Machine Learning* (Springer, New York).

Brier E, Pearson R (2018) Upwork's SVP of marketing explains what it takes to perfect an offering that relies on people. Accessed November 28, 2020, https://www.prnewswire.com/news-releases/snagajob-appoints-former-upwork-ceo-to-board-of-directors-300417689.html.

Brownlee J (2017) Stacked long short-term memory networks. Accessed November 28, 2020, https://machinelearningmastery.com/stacked-long-short-term-memory-networks/.

Brynjolfsson E, Hu Y, Simester D (2011) Goodbye pareto principle, hello long tail: The effect of search costs on the concentration of product sales. *Management Sci.* 57:1373–1386.

Byrd HR, Lu P, Nocedal J, Zhu C (1995) A limited memory algorithm for bound constrained optimization. *SIAM J. Sci. Comput.* 16:1190–1208.

Carr MS (2003) Note on online auctions with costly bid evaluation. *Management Sci.* 49:1521–1528.

Chen DL, Horton JJ (2016) Are online labor markets spot markets for tasks? A field experiment on the behavioral response to wage cuts. *Inform. Systems Res.* 27:403–423.

Chen P-Y, Shin-yi W, Yoon J (2004) The impact of online recommendations and consumer feedback on sales. *Internat. Conf. Inform. Systems*.

Chen T, Guestrin C (2016) Xgboost: A scalable tree boosting system. *Internat. Conf. Knowledge Discov. Data Mining* (Association for Computing Machinery, New York), 785–794.

Cochrane C (2018) Time series nested cross-validation. Accessed March 3, 2021, https://towardsdatascience.com/time-series-nested-cross-validation-76adba623eb9.

Dimoka A, Hong Y, Pavlou PA (2012) On product uncertainty in online markets: Theory and evidence. *MIS Quart.* 36:395–426.

Eckhardt K (2018) Choosing the right hyperparameters for a simple LSTM using Keras. Accessed November 28, 2020, https://towardsdatascience.com/choosing-the-right-hyperparameters-for-a-simple-lstm-using-keras-f8e9ed76f046.

Färber F, Weitzel T, Keim T (2003) An automated recommendation approach to selection in personnel recruitment. *Americas Conf. Inform. Systems*.

Fawcett T (2006) An introduction to ROC analysis. *Pattern Recognition Lett.* 27:861–874.

Ferri FJ, Pudil P, Hatef M, Kittler J (1994) Comparative study of techniques for large-scale feature selection. *Machine Intelligence and Pattern Recognition*, vol. 16 (Elsevier, Amsterdam), 403–413.

Filippas A, Horton JJ, Golden J (2018) Reputation inflation. *EC'18 Proc. 2018 ACM Conf. Econom. Computation* (Association for Computing Machinery, New York), 483–484.

Fleder D, Hosanagar K (2009) Blockbuster culture's next rise or fall: The impact of recommender systems on sales diversity. *Management Sci.* 55:697–712.

Freelancers-union (2017) Freelancing in America. Accessed December 2, 2019, https://s3-us-west-1.amazonaws.com/adquiro-content-prod/documents/Infographic_UP-URL_2040x1180.pdf.

Geva T, Saar-Tsechansky M (2016) Who's a good decision maker? Data-driven expert worker ranking under unobservable quality. *Internat. Conf. Inform. Systems*.

Goswami A, Hedayati F, Mohapatra P (2014) Recommendation systems for markets with two sided preferences. *Internat. Conf. Machine Learning Applications*, 282–287.

Guasch JL, Straub S, Laffont J-J (2003) Renegotiation of concession contracts in Latin America. Policy Research Working Paper, The World Bank, Washington, DC.

Guo X, Gong J, Pavlou P (2017) Call for bids to improve matching efficiency: Evidence from online labor markets. *Internat. Conf. Inform. Systems*.

Ho TK (1998) The random subspace method for constructing decision forests. *IEEE Trans. Pattern Anal. Machine Intelligence* 20:832–844.

Hochreiter S, Schmidhuber J (1997) Long short-term memory. *Neural Comput.* 9:1735–1780.

Hong Y, Wang C, Pavlou PA (2015) Comparing open and sealed bid auctions: Evidence from online labor markets. *Inform. Systems Res.* 27:49–69.

Horton JJ, Chilton LB (2010) The labor economics of paid crowdsourcing. *Conf. Electronic Commerce*, 209–218.

Hosseinzadeh Aghdam M, Hariri N, Mobasher B, Burke R (2015) Adapting recommendations to contextual changes using hierarchical hidden Markov models. *Conf. Recommender Systems*, 241–244.

Institute of Business Value (2019) The enterprise guide to closing the skills gap. Accessed December 2, 2019, https://www.ibm.com/downloads/cas/EPYMNBJA.

Jerath K, Fader PS, Hardie GSB (2011) New perspectives on customer "death" using a generalization of the pareto/NBD model. *Marketing Sci.* 30:866–880.

Kantor BP, Rokach L, Ricci F, Shapira B (2011) *Recommender Systems Handbook* (Springer, New York).

Keras API (2021) Dense layer. Accessed November 28, 2020, https://keras.io/api/layers/core_layers/dense/.

Klazema M (2018) Why is hiring the right employee so difficult? Accessed November 28, 2020, https://www.4workplaces.com/general/why-is-hiring-the-right-employee-so-difficult/.

Kokkodis M (2019) Reputation deflation through dynamic expertise assessment in online labor markets. *World Wide Web Conf.* (Association for Computing Machinery, New York), 896–905.

Kokkodis M (2021) Dynamic, multidimensional, and skillset-specific reputation systems for online work. *Inform. Systems Res.* 32:688–712.

Kokkodis M (2022) Adjusting skillset cohesion in online labor markets: Reputation gains and opportunity losses. *Inform. Systems Res.*, ePub ahead of print November 14, https://doi.org/10.1287/isre.2022.1177.

Kokkodis M, Ipeirotis PG (2014) The utility of skills in online labor markets. *Internat. Conf. Inform. Systems*.

Kokkodis M, Ipeirotis PG (2016) Reputation transferability in online labor markets. *Management Sci.* 62:1687–1706.

Kokkodis M, Ipeirotis PG (2021) Demand-aware career path recommendations: A reinforcement learning approach. *Management Sci.* 67:4362–4383.

Kokkodis M, Ransbotham S (2022) Learning to successfully hire in online labor markets. *Management Sci.*, ePub ahead of print August 26, https://doi.org/10.1287/mnsc.2022.4426.

Kokkodis M, Adamopoulos P, Ransbotham S (2023) Asymmetric reputation spillover effects from digital agencies in online markets. *MIS Quart.* Forthcoming.

Kokkodis M, Lappas T, Ransbotham S (2020) From lurkers to workers: Predicting voluntary contribution and community welfare. *Inform. Systems Res.* 31:607–626.

Kokkodis M, Papadimitriou P, Ipeirotis PG (2015) Hiring behavior models for online labor markets. *Internat. Conf. Web Search Data Mining*, 223–232.

Koller D, Friedman N (2009) *Probabilistic Graphical Models: Principles and Techniques* (MIT Press, Cambridge, MA).

Koren Y, Bell R, Volinsky C (2009) Matrix factorization techniques for recommender systems. *Computer* 42(8):30–37.

Kula M (2018) Deep recommender models using PyTorch. Accessed December 2, 2019, https://github.com/maciejkula/spotlight.

Lauren D (2017) Snagajob appoints former Upwork CEO to board of directors. Accessed November 28, 2020, https://www.prnewswire.com/news-releases/snagajob-appoints-former-upwork-ceo-to-board-of-directors-300417689.html.

MacDonald IL (2014) Numerical maximisation of likelihood: A neglected alternative to EM? *Internat. Statist. Rev.* 82:296–308.

Malinowski J, Keim T, Wendt O, Weitzel T (2006) Matching people and jobs: A bilateral recommendation approach. *Hawaii Internat. Conf. System Sci.*, vol. 6, 137c–137c.

Mao K, Yang Y, Wang Q, Jia Y, Harman M (2015) Developer recommendation for crowdsourced software development tasks. *Sympos. Service-Oriented System Engrg.*, 347–356.

Moreno A, Terwiesch C (2014) Doing business with strangers: Reputation in online service marketplaces. *Inform. Systems Res.* 25:865–886.

Murphy KP (2012) *Machine Learning: A Probabilistic Perspective* (MIT Press, Cambridge, MA).

Pallais A (2014) Inefficient hiring in entry-level labor markets. *Amer. Econom. Rev.* 104:3565–3599.

Pathak B, Garfinkel R, Gopal RD, Venkatesan R, Yin F (2010) Empirical analysis of the impact of recommender systems on sales. *J. Management Inform. Systems* 27:159–188.

Pelletier A, Thomas C (2018) Information in online labour markets. *Oxford Rev. Econom. Policy* 34:376–392.

Provost F, Fawcett T (2001) Robust classification for imprecise environments. *Machine Learning* 42:203–231.

Quadrana M, Cremonesi P, Jannach D (2018) Sequence-aware recommender systems. *ACM Comput. Surveys* 51:66:1–66:36.

Ricci F, Rokach L, Shapira B (2011) Introduction to recommender systems handbook. Ricci F, Rokach L, Shapira B, Kantor P, eds. *Recommender Systems Handbook* (Springer, Boston), 1–35.

Sahoo N, Singh PV, Mukhopadhyay T (2012) A hidden Markov model for collaborative filtering. *MIS Quart.* 36:1329–1356.

Snir EM, Hitt LM (2003) Costly bidding in online markets for it services. *Management Sci.* 49:1504–1520.

Sundararajan A (2016) *The Sharing Economy: The End of Employment and the Rise of Crowd-Based Capitalism* (MIT Press, Cambridge, MA).

Tripp MT, Grégoire Y (2011) When unhappy customers strike back on the Internet. *MIT Sloan Management Rev.* 52:37–44.

Upwork (2014) Online work report. Accessed November 28, 2020, https://web.archive.org/web/20180228011632/http://elance-odesk.com:80/online-work-report-global.

Zhang H, Ni W, Li X, Yang Y (2016) Modeling the heterogeneous duration of user interest in time-dependent recommendation: A hidden semi-Markov approach. *IEEE Trans. Systems Man, Cybernetics Systems* 48:177–194.

Zhang S, Yao L, Sun A, Tay Y (2019) Deep learning based recommender system: A survey and new perspectives. *ACM Comput. Surveys* 52:5.

Zheng A, Hong Y, Pavlou P (2015) Value uncertainty and buyer contracting: Evidence from online labor markets. *Internat. Conf. Inform. Systems*.