Algorithms for Labor Markets

John Horton
MIT & NBER

April 2023

Abstract

Labor markets are characterized by substantial search frictions, and yet, research on algorithmic approaches to overcome those frictions is not part of economics: search frictions are more discussed than reduced. One might argue that reducing search frictions is a task for private companies or computer scientists, but we argue that economics has something to offer and that approaches not steeped in economic thinking can exacerbate market matching problems. In this paper, we present a framework for organizing labor market data to support recommendations and give examples using real data from an online labor market.

Contents

1 Introduction **1**

2 What is hard about labor market matching? **2**
2.1 Both sides are highly differentiated 2
2.2 Incomplete information 2
2.3 Sellers have capacity constraints 2
2.4 Sparsity of data 3
2.5 The stigma of being unmatched and unwillingness to reveal status 3
2.6 Greater ethical considerations in making recommendations 3

3 Algorithmic approahces **4**
3.1 What would Hayek say? 5
3.2 Labor market recommendation about which worker to hire 5
3.3 How is this different from recommender systems more generally? 5

4 Desiderata for algorithm design for labor markets **6**
4.1 Legibility, interpretibility and control 6
4.2 Horizontal, not vertical 6
4.3 Hayekian 6
4.4 Easy to improve 6
4.5 Economic 6
4.6 Robust to manipulation 6
4.7 Works comfortably with text 7

5 A unified approach to labor market recommendations **7**
5.1 Formalizing this intuition 7
5.2 Similarity between a new worker and a new job 8

6 Accounting for "Choose and Be Chosen" **9**
6.1 Pseudo-Gale Shapley 9

7 Extended example **9**
7.1 Neural networks 11
7.2 Human-Capital-to-Requirements: $S^T AR$ 12
7.3 Applications are a revealed preference measure 12
7.4 Relevant workers to a new job: $rR^T A$ 13
7.5 The importance of $X^T X$ and XX^T 13

8 Using regression **15**

9 Using real data **15**

10 Create application matrix, A **15**
10.1 The $A^T A$ matrix has count of applications in common between two jobs . . . 16
10.2 Similarity measure between jobs 16
10.3 The AAT matrix has count of applications in common between two workers . 16

11 Creating Requirements Matrix **16**

12 Top Skills : Diagonal $R^T R$ **16**

13 Co-occurring skills from $R^T R$ **16**

14 Skill Clustering **16**
14.1 Speed 16

15 Find suitable workers for a new job **16**
15.1 The new job projected into "existing job" space: 17
15.2 Normalization: jobs and workers should be unit vectors 17
15.3 The new job projected into "worker" space and "human capital" space: . . . 17

16 Finding similar workers **17**
16.1 Function for making recommendations 17

17 Freelancer Skill Summary **17**

18 What was the average degree of competition a job-seeker faced? **18**

19 Inferring productivity (see Pewter memo) **18**

20 Wage bid by skill **18**

A Generative AI and Labor Markets **19**
A.1 Why signal value to writing is likely dead 19
A.2 Generative AI and synthetic data 19

1 Introduction

The idea that an information system might help us to overcome market search frictions is not novel: Resnick and Varian (1997) laid out clearly the case for recommender systems in product markets. There, recommender systems have worked remarkably well.

Given their importance of recommender systems in many product markets, one would think that they would be hugely important in labor markets as well. Over 20 years ago, Autor (2001) made the case that digitization could be important. After all, labor economics exists as a separate field, in part, because labor markets have so many difficult-to-overcome frictions. And yet the continued importance of word-of-mouth and social networking in labor market matching is partly a testament to how hard some of these matching problems are in practice.

There has been little evidence that the Internet did much to improve job search (Kuhn and Skuterud, 2004). The rise of Craigslist had large effects on other search markets, but not on labor markets (Kroft and Pope, 2014). Marinescu and Wolthoff (2020) show how important words are in matching. There is evidence that algorithmic approaches can increase match probability (**?**).

The purpose of this paper is (1) to summarize key ideas in the recommender system literature, but through the lens of labor markets, (2) to review the nascent CS literature on applying recommender system ideas to labor markets, (3) to present a unified conceptual way to think of the labor market recommendation problems, and (4) to present examples of applying these ideas to real labor market data. Our goal is to stimulate research interest in these problems. Although currently most of this happens on private platforms, innovations by researchers could spill over into practice.

Existing CS approaches to labor market recommendations. Hitsch et al. (2010); Zhu et al. (2018) estimate preferences and then examine whether realized matches are similar to what is predicted by Gale-Shapley.

Thicker markets are good, but thicker markets are also more difficult to navigate. This might be why the literature typically finds constant returns to scale to markets. The benefit of thickness is offset by the challenges. The rise of remote work has thickened markets but has not scaled up word-of-mouth. There is likely a loss of context.

2 What is hard about labor market matching?

Labor markets are matching markets, and so participants must choose and *be chosen* for a match. This means there is inherently a process of information discovery and selection on both sides. This stands in contrast to non-matching markets where one or more sides of the market are indifferent over their counterparty.

2.1 Both sides are highly differentiated

This search process can be daunting simply because of the variety on both sides. Workers and jobs are, of course, "horizontally differentiated:" there is not "labor" but rather an enormous number of different combinations of skills, experiences, knowledge, and preferences. These are mirrored by a collection of firm needs that vary dramatically in ways not captured by a simple title or job description. Workers also differ not just in their focus but their level of experience and ability. Firms also mirror these preferences, having jobs where they need very able workers and others where they are satisfied with less-than-expert workers.

2.2 Incomplete information

The number of job seekers can be vast, as can the number of job posts. Workers are not fully informed about companies, and companies are not fully informed about sellers. Nor can they be fully informed, as private information abounds: Workers are not fully informed about firms; firms are not fully informed about workers.

Firms do not know who is necessarily "on the market" and looking for more work. Even if participants want to express information truthfully, people have complex preferences that are hard to express. There is often no standard way to express attributes of jobs or job-seekers.

2.3 Sellers have capacity constraints

What workers sell in labor markets is their time, and all people have binding capacity constraints with respect to time. We cannot have every job done by one "best" worker, the same way we could give the best product to everyone. This feature of labor markets has motivated things like max flow matching and deferred acceptance algorithms. Or to deal with the indivisibility of labor with matching models. Even outside highly controlled contexts like residence match, this capacity constraint also affects the nature of the recommendations we can make.

In a competitive product market, saying a particular item is "good" and recommending it to many users likely just increases sales for that good, but probably not the price, which is pinned down by marginal cost. In a labor market, because job-seekers are capacity-constrained, giving them more job offers does not lead to them doing (much) more work: they either stock out or simply raise their prices.

2.4 Sparsity of data

In conventional product markets, the platform often observes co-purchases that, in turn, allow for good recommendations about complementary products: A person buying an oven might also buy a pizza stone; a person buying shoes might buy a matching belt; a person buying a coffee maker might buy a grinder. Note that these kinds of recommendations do not require some detailed, context-specific knowledge about the nature of the goods, but rather can just be based on statistical patterns in the observed co-purchase data.

The data we typically observe in labor markets is a particular worker matched with a particular firm, for some amount of time. We typically do not observe the consideration set for either side: which jobs a worker was considering and which workers a firm was considering.

This makes the "People who bought X also bought Y" type of recommendations difficult to make.

Job posts are ephemeral. There is typically not some job open for years on end that we can recommend many users to (though there are exceptions in some labor markets where, say, Starbucks is always hiring), and job-seekers are typically not searching for years on end. Instead, right when we have seen a job for some amount of time, it no longer needs recommendations. And if we were restricted to making recommendations solely on graph-based properties, we would be unlikely to make recommendations as it would take time for the node to get edges (as people apply). Similarly, workers are frequently exiting and entering the labor market.

2.5 The stigma of being unmatched and unwillingness to reveal status

In labor markets, being unmatched i.e., unemployed, can be a negative signal. There is good evidence that employers are less interested in workers that have been unemployed for longer periods. New entrants to a market without any history might also be regarded as less productive, which can make it harder for Cite: Tervio/Pallais

Existing employers might be upset to learn that a current employee is seeking to work elsewhere.

2.6 Greater ethical considerations in making recommendations

Concerns about fairness, legibility, and interpretability are central considerations. If some recommender system model for products learns that the color shade of luggage is associated with a price (say because "black" = more serious and professional and pink = "fun and cheeky"), if it starts recommending based on some obscured calculation of color, it is no big deal. If the same happened in labor markets—taking profile picture skin color into account—this would be undesirable, to put it mildly.

3 Algorithmic approahces

Collaborative filtering is a technique used in recommender systems to make predictions about a user's preferences or interests by leveraging the preferences or behaviors of similar users. In other words, it involves analyzing the patterns of behavior of groups of users to identify commonalities and recommend items that a user might be interested in. Collaborative filtering is based on the assumption that people who have similar preferences in the past are likely to have similar preferences in the future, and thus the system can make recommendations based on the similarities among these users.

There are two main types of collaborative filtering: user-based collaborative filtering and item-based collaborative filtering. In user-based collaborative filtering, the system recommends items to a user based on the preferences of similar users, while in item-based collaborative filtering, the system recommends items to a user based on the similarity of the items themselves. Collaborative filtering is widely used in e-commerce, social media, and content recommendation systems.

What tasks for the system?

- Recommendations that benefit hiring firms
  - Recommend workers to recruit
  - Recommend workers to hire
  - Show similar workers to a given worker
  - Recommend improvements in job titles, and descriptions
- Recommendations that benefit workers
  - Show similar jobs to a given job
  - Recommend jobs that a worker might be interested in
  - Recommend improvements in resumes and cover letters
- Help both sides understand "market" rates and provide a bird's eye view

What are the recommender system task?

1. What skills go together?
2. What workers are similar to each other, skill-wise?
3. What workers are similar to each other, application-wise?
4. What jobs are similar to each other, application-wise?
5. What jobs are similar to each other, requirements-wise?
6. What skills are needed for different job requirements?
7. What requirements go together?

3.1 What would Hayek say?

Matching algorithms like deferred acceptance start from the assumption that both sides have preferences over the other side. But most of the work in labor markets is actually in acquiring the information needed to have preferences.

The recommender system literature starts with the assumption that computing relevance is straightforward. But getting the information in the form needed to make this assessment is challenging. It also needs to be done in a way that is robust to job seekers and jobs exiting and entering the market.

3.2 Labor market recommendation about which worker to hire

Suppose two factors matter to firms: a worker's productivity, y, and the price that the worker is proposing w. The pay-off to the firm from a hire is $v = y - w$. Suppose we have access to two candidates we can recommend: Mr. Expert ($w = 2$, $y = 2$) and Mr. Novice ($w = 1$, $y = 1$). If we observe prices without abilities, we naively recommend Mr. Novice. If we observe abilities without prices, we naively recommend Mr. Expert. If we observe both w and y, then we should be indifferent in our recommendation, as both workers offer the same value. As we can see, naive recommendations made on the basis of partial information can be highly misleading.

One might argue that when one is missing, you can input w from y or vice versa. But this is a problem when w is such a mutable characteristic and the market is competitive. If you impute w from y and think the market is competitive, then your imputation will tell you the firm should be indifferent over candidates. This is because the imputation will always be $\hat{w} = y$. If you try to go the other way—impute y from w—you have put yourself in a dangerous position in that the workers can make themselves seem better simply by increasing their wage-ask. And in any event, you will still be left with the prediction that the buyer should be indifferent over workers. What if you do observe w and y at some moment in time and then recommend the "best" worker given what you know? But if the recommended worker *knows* they are recommended, they have an incentive to raise their wage ask—possibly right up until the hiring employer is indifferent again.

The implication is that labor market recommendations, trying to say who is the "best" candidate for a job algorithmically, is not likely to work that well for what are, fundamentally, economic reasons. This is a separate issue from the ethical risks inherent in saying who should get a job on the basis of a poorly-understood model.

3.3 How is this different from recommender systems more generally?

Consider the canonical collaborative filtering scenario, where n raters rate m different "items":

$$R = \begin{bmatrix} r_{1,1} & r_{1,2} & \cdots & r_{1,m} \\ r_{2,1} & r_{2,2} & \cdots & r_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ r_{n,1} & r_{n,2} & \cdots & r_{n,m} \end{bmatrix} \tag{1}$$

This matrix tends to be very sparse for product markets—users have only rated a few items, and the number of items can be vast.

Although it fluctuates, there are, on the order of 10 million active job seekers in the US and, 10 million active vacancies at a given time. The rating term matrix would have 100 *trillion* entries. But even then, this is not a very useful framework. In the same way you have to watch a movie to rate it, you need to have had the job to rate it. But a job that has been had in the past is not likely to be posted right now in the market.

Su et al. (2021)

Glassdoor works by rating a component—the company—or getting repeated ratings on a role that exists at the company that lots of people might have. This can help with assessment for a job seeker, but it not likely to be great at discovery.

4 Desiderata for algorithm design for labor markets

4.1 Legibility, interpretibility and control

One approach to fairness is to give workers control in how they are presented to the marketplace.

4.2 Horizontal, not vertical

The platform should not decide who is “good.” This is because it ignores the role of price. It also creates risk.

4.3 Hayekian

We do not know what we do not know.

There are lots of idiosyncratic reasons a person might not take a job. A person might be afraid of bridges and will not take a job that requires crossing a river—but will be happy to take a job in the “other” direction. Or maybe they are a carpenter and cannot take the carpenter job at the apiary because they are allergic to bees. We do not have to mandate matches—we need to get people in the same room. The goal should be to get information into the system that is useful to participants.

A small improvement in “performance” that comes at the expense of making recommendations entirely illegible is a bad design.

4.4 Easy to improve

Needs to adapt to changing considerations without complex re-training. It should be “set and forget.”

4.5 Economic

Needs to adapt to changing considerations without complex re-training. It should be “set and forget.”

4.6 Robust to manipulation

It should be hard for wide-scale manipulation. For example, it is hard for an individual to move a median, but it is easy for them to move a mean.

4.7 Works comfortably with text

Content-based recommendations are difficult with jobs and workers, partially because we have to make predictions across different “spaces.”

A “Developer” could mean lots of things. Or even if a job has the same titles, simple keyword matching can lead to you suggesting things like “Professor of Optometry’ jobs to “Professor of Economics” people.

5 A unified approach to labor market recommendations

Imagine a simple labor market where there are just three products, which are foods: berries, game (e.g., venison), and fish. These goods are produced, respectively, by gathering, hunting, and fishing. Suppose we have three workers in the labor market, one who specializes in gathering, Bob; another who specializes in hunting, Gabby; and another that specializes in angling (fishing), Freddie. Jobs are contract positions posted by a hungry person. Suppose there are just three jobs, one posted by BerryLover, one posted by GameLover, and another by FishLover.

Which job should we recommend to which workers? If we used our substantive knowledge of the situation, the answer is clear—we should recommend Bob to the BerryLover and Gabby to the GameLover, and Freddie to FishLover. Their skills match the product market requirements.

But suppose we do not know the productive process. We do not know that hunting produces game, gathering produces berries, and angling produces fish. This is the reality of any complex labor market: there are degrees of differentiation much finer and productive processes more complex than we can easily understand. We, as a platform, do not know which skills "go" with which production process, at least with any degree of certainty.[1]

In the absence of something else to help us, we are stuck. But now, suppose we get to observe applications where the job-seekers applied to the "right" jobs. We would learn, over time, that people who describe themselves as gathering experts tend to apply to the same jobs. If two job seekers apply to the same job, those job seekers are probably similar to each other in some sense, so long as they face some cost to applying to a job and know something about what they are applying to. If we see two jobs with a lot of overlap in applicants, they are probably similar to each other.

5.1 Formalizing this intuition

Imagine a vector h that captures the attributes of a worker. It could be counts of jobs or earnings using various human-interpretable skills. It could be the text embedding of a paragraph describing the worker's history. It could be some combination of these various things, with the vectors stacked on top of each other. We can have a corresponding vector r that captures job "requirements." It could also be an embedding of the job description and title or key attributes of desired applicants.

Now imagine we observe an active market for some amount of time, with N workers sending applications to J jobs. Let us construct a $(N \times J)$ matrix A with indicators for whether worker i applied to job j. We can have a corresponding matrix C with indicators for whether worker i was hired by j. This could either be the result of an application or

We have four important matrices, with the following dimensions:

1. Skills/human capital, H: skills $\times$ workers
2. Applications, A: workers $\times$ jobs, indicators for whether that worker applied to that job
3. Contracts, C: workers $\times$ jobs, indicators for whether that worker was hired for that particular job
4. Requirements, R: jobs $\times$ requirements

[1]Even in our simple example, consider how many words there are for fish people eat: Salmon, Tuna, Cod, Haddock, Trout, Mahi-mahi, Tilapia, Catfish, Swordfish, Halibut, Grouper, Snapper, Anchovy, Mackerel, Barramundi, Perch, Carp, Sole, Flounder, Sardines, Bass, Bluefish, Branzino, Catla, Chilean sea bass, Cobia, Corvina, Drum, Eel, Flathead, Garfish, Herring, John Dory, Kingfish, Largemouth bass, Monkfish, Orange roughy, Pike, Pollock, Pompano, Redfish, Rockfish, Scup, Skate, Smelt, Striped bass, Sturgeon, Swai, Tilefish, and Wahoo.

1. What is the count of applications per job? $\operatorname{diag} A^T A$
2. What is the count of applications per worker? $\operatorname{diag} AA^T$

5.2 Similarity between a new worker and a new job

From our historical data, how do we calculate the similarity between a new worker and a new job, not in our training data? If we have a new worker joining the market, with a human capital vector h, then the cosine similarities with the existing collection of N workers would provide a measure of similarity.

To compute the cosine similarity, we need to scale the dot products by the lengths of the constituent human capital vectors, or

$$\Sigma_H = 1/\sqrt{\operatorname{diag}(H^T H)} = \begin{bmatrix} |h_1|^{-1} & 0 & \cdots & 0 \\ 0 & |h_2|^{-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & |h_N|^{-1} \end{bmatrix}. \tag{2}$$

We can use this to compute

$$s = h/|h| H \Sigma_H \tag{3}$$

which has dimension $1 \times N$, where N is the number of jobs. These measures of worker similarity could now be used for a nearest-neighbor style recommendation of similar workers directly.

Let $\Sigma_A = 1/\operatorname{diag}(A^T A)$ be a diagonal matrix the diagonals being the reciprocal of the count of applications for that job. To obtain a measure of how a new worker is related to jobs, If we multiply x by A and normalize by application counts per job

$$x_{h \to J} = h/|h| H \Sigma_H A \Sigma_A, \tag{4}$$

which gives us the mean cosine similarity of workers applying to each of the jobs, in the collection of jobs. For the new job, r, we do the same multiplication and scaling to get the mean similarity between this new job and all existing jobs, $x_{r \to J}$.

We now find the similarity between these two vectors to get a similarity between an entirely new worker and an entirely new job, but based on historical, revealed-preference data:

$$\begin{aligned} S(h, r) &= \frac{x_{h \to J} \cdot x_{r \to J}}{|x_{h \to J}||x_{r \to J}|} && (5) \\ &= \frac{h W_A r^T}{|h||r|} && (6) \end{aligned}$$

where $W_A = (H \Sigma_H A \Sigma_A R \Sigma_R)$.

Note that despite the potentially enormous size of H, A, and R, the matrix W is only skills $\times$ requirements in dimension. Furthermore, given the clustering of skills and requirements, there is likely a low-rank approximation to W_A that reduces the computation burden. For example, an SVD of W_A with k singular values would greatly reduce the number of calculations.

Note that for entirely new jobs or workers, we have to make use of H and R as measures of similarity. But for workers that have a history and have co-applied to jobs with other workers, then AA^T offers a measure of worker-worker similarity. And by the same token, jobs that are currently receiving applicants are related to each other by $A^T A$. This is essentially going from a content-based similarity measure to a graph-based similarity measure.

Figure 1 illustrates the "movement" between various spaces and the important matrix (without any scaling).

6 Accounting for "Choose and Be Chosen"

If we want to make recommendations that represent the "choose and be chosen" aspect of the problem, we need to consider not just what workers an employer prefers, but how likely those workers are to prefer the employer. One heuristic might be to take a collection of candidate job-seekers, H' and order candidates not by $S(h, r; C)$, but by $S(h, r; C)S(h, r; A)$.

6.1 Pseudo-Gale Shapley

Another more structured approach might be to take the H' candidate workers, then draw simulated jobs, R'. Impute all the preferences for both workers and firms. Then run Gale-Shapley to get matches, $M(H', R' \cup r)$ and take the vector corresponding to r, which will give which of the H' workers was hired in that scenario. Bootstrap R' some number of times and get an estimated hire probability for each worker in h, and rank accordingly.

7 Extended example

Imagine job-seekers can be described by a d-dimensional "human capital" vector, h. There are only three skills, fishing, hunting, and gathering, in order. A job seeker that is just a hunter—Gabby—would be $h_g = (0, 1, 0)$. We can have our collection of (workers $\times$ skills) matrix as

$$H = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix},$$

corresponding to three workers focusing on gathering, hunting, and fishing. Note that the order of the columns is arbitrary. Similarly, a job post might capture the horizontal "requirements." A row that looks like (game, fish, berries)

$$R = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

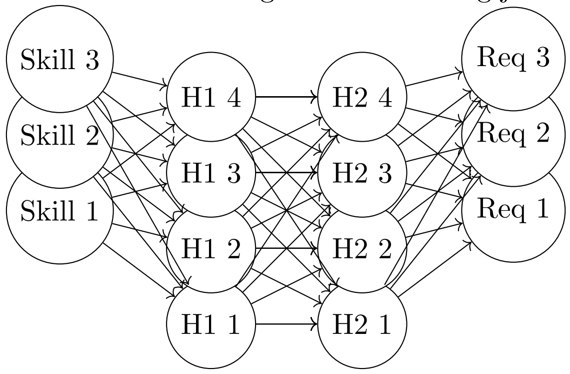
Note that even if we wanted to take a dot product here, notice that our vectors do not line up.

If we create indices for both jobs and workers, we can populate a matrix representing the application graph, "A"

Figure 1: Representation of movements between areas

$$\begin{tikzcd}
\text{Skills} \arrow[loop above, "HH^T"] \arrow[r, bend left, "H"] \arrow[d, bend left, "HA"'] \arrow[dr, bend left, "HAR"'] & \text{Workers} \arrow[loop above, "AA^T \text{ or } H^TH"] \arrow[dl, bend left, "A"] \arrow[d, bend left, "AR"] \\
\text{Jobs} \arrow[loop below, "A^TA \text{ or } R^TR"] \arrow[r, bend left, "R"] & \text{Requirements} \arrow[loop right, "RR^T"]
\end{tikzcd}$$

Figure 2: Predicting job requirements from worker skills



$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

These three matrices tell us how to move back and forth between different spaces :

$$\{H, A, R\} \rightarrow \left\{ \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \right\}$$

7.1 Neural networks

This framing of the problem restricts the modeling to a simple form—namely linear models. Of course, we can have far more complex relationships in actual data. A person might have the text “Python” and apply for jobs that list requirements like “Databases” and “Software development”—but they also might apply for jobs at a zoo, working in the reptile house, which requires “Care and feeding of snakes.” To learn these higher-level interrelationships, more complex models are required. Fortunately, the structure helps us think about the shape of the networks we want to train.

For example, in our berry-game-fish example, if we want to map worker skills to job requirements, we could construct a 3-node input layer, a 3-node output layer and some combination of hidden layers. For our training data, the inputs are the columns of S, the outputs are the rows of R, with the A matrix telling us what to link up.

Note that because of the ephemerality of jobs, we are likely *not* training models of worker-to-worker, worker-to-job, or job-to-job, or job-to-worker directly, but rather as reflected into the appropriate skills of requirements space.

7.2 Human-Capital-to-Requirements: $S^T AR$

Without any substantive knowledge about what “hunting,” and “fishing” and “gathering” actually do, the revealed application decisions show us the connections between requirements and skills. If we look at the transposed H matrix times the application matrix, times the requirements matrix, we get:

$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

If we take the full "outer product" of human capital names and requirements names, we get rows corresponding to human capital entries and columns corresponding to requirements:

$$\begin{pmatrix} \text{fishing} \rightarrow \text{venison} & \text{fishing} \rightarrow \text{fish} & \text{fishing} \rightarrow \text{berries} \\ \text{hunting} \rightarrow \text{venison} & \text{hunting} \rightarrow \text{fish} & \text{hunting} \rightarrow \text{berries} \\ \text{gathering} \rightarrow \text{venison} & \text{gathering} \rightarrow \text{fish} & \text{gathering} \rightarrow \text{berries} \end{pmatrix}$$

Many of these are zero: fishing doesn't lead to venison, fishing doesn't lead to berries and so on. If we look at the actual non-zero entries, we get: { fishing $\rightarrow$ fish, hunting $\rightarrow$ venison, gathering $\rightarrow$ berries }

The platform learned the mapping without substantive knowledge. Now, in practice, jobs are more complex and have different sets of requirements and workers are bundles of skills. But this simple example shows the basic idea that the application graph is what teaches you relevance.

7.3 Applications are a revealed preference measure

The reason this works is that job applications are costly because people do not like wasting their time. As such, the application graph is revealed preference measure of similarity. This is fundamentally different from, say, taking job post text and worker profile text try to come up with some similarity notion based on textual overlap.

A software **developer** and a condo **developer** are not "close" in the worker "space" despite having similar job titles. This may seem silly, but I get lots of recommendations for "Professor" jobs in fields like ophthalmology and veterinary science because "professor" is in my job title.

But what about the word "developer" that might show up in H and R? The nice thing about the application graph approach is that entries that are corrupted or made useless by synonymy end up mattering very little to recommendations. And to the extent we do dimensionality reduction by, say taking SVD of these various matrices (more on this later), they end up not mattering.

7.4 Relevant workers to a new job: rR^TA

Now suppose we have a new job. Which worker should we recommend? We can simply create a new requirements vector and then project it into "worker" space:

First, imagine the jobs is just a duplicate of the job from the person who likes venison:

If we did not want to make a point recommendation to Gabby, we could add more one step and multiply by H to get the new job in terms of human capital requirements: Now suppose we have a truly new job that is from someone who likes both venison and fish. We can do the same process and learn that both Gabby and Freddie might be reasonable choices (in "worker" space) and that in human capital space, we will need both fishing and hunting.

7.5 The importance of X^TX and XX^T

The "squared" version of our A, H, and R matrices have special importance and interpretable meaning. In a nutshell, they let us have some notion of distance, or similarity, between entities. To see this, let us construct a more elaborate requirements matrix that includes the "surfAndTurf" lover:

$$\begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix}$$

If we think of our three possible requirements, the "outer product" of these three are:

$$\begin{pmatrix} \text{2 venison} & \text{fish + venison} & \text{berries + venison} \\ \text{fish + venison} & \text{2 fish} & \text{berries + fish} \\ \text{berries + venison} & \text{berries +fish} & \text{2 berries} \end{pmatrix}$$

If we do R^TR, then we get

$$\begin{pmatrix} 2 & 1 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Along the diagonal, what this is showing is that we had two jobs that required venison, two that required fish and 1 that required berries. The off-diagonals show that we 1 job that required both venison and fish (the surf & turf job). It is really only the upper triangle that is relevant:

$$\begin{pmatrix} 2 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

This matrix is informative about what requirements tend to "go together. "If we do RR^T, then we get the "jobs" view as opposed to the skills view:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 2 \end{pmatrix}$$

The matrix is now showing us the overlap across jobs. For example, "berry lover" only has one skill in common with another job–namely itself, which is gathering. Note, however, that "surf & turf" has two requirements in common (hunting and fishing). It also has one in common with fishLover and one in common with deerLover. The H matrix of human capital requirements has a very similar interpretation, with both the "skill" view and the "worker" view.

For the application graph, let us make a slightly more inter scenario where both Freddie and Gabby apply to the surf & Turf job:

```
apps =
  { "berryBob" → "berryLover", "gameGabby" → "deerLover", "fishFreddie " → "fishLover ",
   "fishFreddie" → "surfAndTurf", "gameGabby" → "surfAndTurf " };
Graph[apps, VertexLabels → Automatic , GraphLayout → "BipartiteEmbedding "]
Jobs = { "berryLover" → 1, "deerLover" → 2, "fishLover" → 3, "surfAndTurf" → 4};
Workers = { "berryBob" → 1, "gameGabby" → 2, "fishFreddie" → 3}
```

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

The application graph, A, offers either a "worker view" or a "jobs view." If we start with the worker view:

$$\begin{pmatrix} 2\text{ berryBob} & \text{berryBob} + \text{gameGabby} & \text{berryBob} + \text{fishFreddie} \\ \text{berryBob} + \text{gameGabby} & 2\text{ gameGabby} & \text{fishFreddie} + \text{gameGabby} \\ \text{berryBob} + \text{fishFreddie} & \text{fishFreddie} + \text{gameGabby} & 2\text{ fishFreddie} \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 1 \\ 0 & 1 & 2 \end{pmatrix}$$

The diagonals are how many applications that worker sent. Berry Bob just went one, while Freddie and Gabby each sent two. In the off-diagonals, we can see that Freddie and Gabby overlap. This is useful, as it is indicative that they are somehow more similar to each other than they are to Bob.

If we take the jobs view:

The diagonals are how many applicants that job got. Most are 1, except surf and turf, which got two applicants. The off-diagonals capture overlap in applicants. Namely, fishLover and SurfAndTurf, deerLover and Surf & Turf.

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \end{pmatrix}$$

8 Using regression

Now suppose we have some information about our jobs like whether or not they fill, and we would like to report the fill rate by probability, or perhaps the average wage. We can easily do this using matrix calculations likely available to us in our computing environment—namely SVD.

$$\left\{\begin{pmatrix} 0 & 1. & 0 & 0 \\ -0.408248 & 0 & -0.707107 & -0.57735 \\ -0.408248 & 0 & 0.707107 & -0.57735 \\ -0.816497 & 0 & 0 & 0.57735 \end{pmatrix}, \begin{pmatrix} 1.73205 & 0 & 0 \\ 0 & 1. & 0 \\ 0 & 0 & 1. \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} -0.707107 & 0 & -0.707107 \\ -0.707107 & 0 & 0.707107 \\ 0 & 1. & 0 \end{pmatrix}\right\}$$

We can see that fishing and hunting jobs both fill 2/3rds of the time, but gathering jobs had a zero fill rate.

{ fishing $\to$ 0.666667, hunting $\to$ 0.666667, gathering $\to$ 0}

9 Using real data

We now load some real data from a large online labor market. It is just a few days with a job posts from 2017. It has data on both the applications and job openings.

10 Create application matrix, A

The rows are workers, the columns are jobs:

This is, of course, an extremely sparse matrix:

10.1 The $A^T A$ matrix has count of applications in common between two jobs

10.2 Similarity measure between jobs

This just takes the AT A matrix and scales the entry by the total number of applications to the two jobs.

$\{123502, 20327\}$

This does not rely at all on skills and content, but you'll see that the jobs are remarkably similar:

For example, here is a job for "Shopify/Squarespace designers" (index = 55). The titles alone show this approach is quite good at sussing out where job-seekers would apply:

10.3 The AAT matrix has count of applications in common between two workers

We can also take a "workers" view of of the application graph to compute notions of similarity. In this example, I have not loaded any data workers, so there is nothing like the title to compare, but I think a similar exercise with profiles would show workers that tend to apply to the same jobs tend to be similar to each other.

The jobs use in this sample only have 3 job skills each.

11 Creating Requirements Matrix

For a given job, we can see the skills it required:

We can turn this into a requirements matrix, R .

12 Top Skills : Diagonal $R^T R$

13 Co-occurring skills from $R^T R$

If we do $R^T R$, we get the count of co-appearance of skills in job posts. Visually, the diagonal is brightly colored (as we would expect) but there are also clearly some skills that very frequently go together:

14 Skill Clustering

14.1 Speed

15 Find suitable workers for a new job

Suppose we have a new job to make recommendations for. Obviously, this job does not exist in "A"—it is new. It also does not exist in R. However, we "know" how to construct new vectors in "jobs" space, which we can then project into jobs space ("these are jobs similar to yours") worker space ("these are workers who could do the job") or human capital space ("these are the attributes of workers that could do your job").

15.1 The new job projected into "existing job" space:

If we take rR^T and then look at the entries in the resulting matrix there are only three values: 0, 1 and 2. This corresponds to jobs that had no overlap with these skills, 1 skills overlapping or 2 skills overlapping.

A sample of the "1s" were job one of the skills :

15.2 Normalization: jobs and workers should be unit vectors

For simplicity, I have just kept everything here bit vectors (1/0) but it it probably makes sense to weight everything so that each worker and job makes the same "contribution." This would mean making the rows of R all be unit vectors and the rows of A all be unit vectors. Otherwise, a job-seeker that sends a ton of applications would get more weight in A, harming relevance if they apply permissively.

15.3 The new job projected into "worker" space and "human capital" space:

To project the new job into worker space, we take another "step" and compute r RT AT . I don't actually have any details on the workers, but we can look at what other jobs they have applied to. First, I find the workers with the best score. Then I get all of their apps and tally up the skills:

The result is clearly a Python-focused web developer, as expected, who has applied to some postgresql jobs. Perhaps this incidental. How do we know this person is postgresql-focused? Well, we do not, but recall our goal here is not to make pin-point matches but rather just get people in the same "room."

16 Finding similar workers

Maybe this particular worker is unavailable or is, in fact, not doing any postgresql work. How can we find similar workers? First, lets take the "row" in AA^T corresponding to our focal worker. We can see that out of our focal worker sent 33 applications:

If we tally up entries, we can see we get mostly 0s, a few 1s, a few 2s, and so on.

This gives us a local neighborhood of similar workers to potentially recommend : $\{20327, 20327\}$

16.1 Function for making recommendations

17 Freelancer Skill Summary

To get a freelancer summary of skills they focus on, we can multiply the application matrix by the requirements matrix :

18 What was the average degree of competition a job-seeker faced?

The total number of competitors a job - seeker faced was:

The average competition a freelancer faces across their application is:

19 Inferring productivity (see Pewter memo)

20 Wage bid by skill

This gets the associated wage bid:

This is the mean wage bid per job:

This gives the mean wage bid for that skill:

References

Autor, David H., "Wiring the Labor Market," *Journal of Economic Perspectives*, March 2001, *15* (1), 25–40.

Hitsch, Gunter J., Ali Hortaçsu, and Dan Ariely, "Matching and Sorting in Online Dating," *American Economic Review*, March 2010, *100* (1), 130–63.

Kroft, Kory and Devin G Pope, "Does online search crowd out traditional search and improve matching efficiency? Evidence from Craigslist," *Journal of Labor Economics*, 2014, *32* (2), 259–303.

Kuhn, Peter and Mikal Skuterud, "Internet job search and unemployment durations," *American Economic Review*, 2004, *94* (1), 218–232.

Marinescu, Ioana and Ronald Wolthoff, "Opening the Black Box of the Matching Function: The Power of Words," *Journal of Labor Economics*, 2020, *38* (2), 535–568.

Resnick, Paul and Hal R Varian, "Recommender systems," *Communications of the ACM*, 1997, *40* (3), 56–58.

Su, Yi, Magd Bayoumi, and Thorsten Joachims, "Optimizing Rankings for Recommendation in Matching Markets," 2021.

Zhu, Chen, Hengshu Zhu, Hui Xiong, Chao Ma, Fang Xie, Pengliang Ding, and Pan Li, "Person-Job Fit: Adapting the Right Talent for the Right Job with Joint Representation Learning," 2018.

A Generative AI and Labor Markets

A.1 Why signal value to writing is likely dead

With the rise of generative AI, the signal value of a cover letter or resume is likely to be diminished because these tools can create highly realistic and personalized content that mimics human writing. As a result, it becomes more challenging for recruiters and hiring managers to differentiate between genuine human-generated applications and those that have been generated by AI.

Generative AI algorithms, such as GPT-3, have demonstrated remarkable abilities to generate text that is difficult to distinguish from text written by humans. These models can analyze large amounts of data and learn to generate language thaChat

Send a message to your collaboratorst fits a particular style, tone, or purpose. They can even generate multiple versions of the same text, making it difficult for employers to determine whether a candidate has actually written a cover letter or resume themselves.

Moreover, generative AI can tailor content to the specific requirements of a job posting by analyzing the job description and incorporating relevant keywords and phrases. This capability can make AI-generated applications more compelling and targeted than those created by humans, leading to further challenges for recruiters and hiring managers in assessing candidate quality.

However, it is worth noting that while AI-generated applications may appear impressive and well-crafted, they may lack the authenticity and personal touch that comes from a human-written application. Personalized insights, anecdotes, and experiences may be difficult for an AI model to generate convincingly. Thus, candidates who can demonstrate their unique skills, experience, and personality in their applications may still have an advantage over AI-generated applications.

A.2 Generative AI and synthetic data

Generative AI can be used to create synthetic job-seekers and job openings, which can be used to generate simulations. These simulations can be useful in a variety of ways, such as:

Synthetic job-seekers can be used to simulate candidate behavior and generate realistic resumes and cover letters. This can be useful for recruiters and hiring managers to test their screening and evaluation processes and identify potential biases. Similarly, synthetic job openings can be used to test the effectiveness of job descriptions and identify areas for improvement.

Research and development: Synthetic job-seekers and job openings can be used to generate data for research purposes. For example, they can be used to simulate the impact of different hiring strategies, such as increasing diversity and inclusion, or to test the effectiveness of new recruitment tools and technologies.