Uncovering Latent Types in Sequential Choice Data Using Text Embedding Algorithm

Nadav Kunievsky

February 6, 2024

Abstract

In economic analyses of agents making a series of discrete choices, deciding what constitutes an alternative is crucial. This paper introduces a technique for categorizing similar alternatives in contexts where forward-looking agents make a series of decisions. The proposed method groups options that are equivalent from the perspective of the agents, using the renowned word2vec algorithm (Mikolov et al., 2013b, Mikolov et al., 2013a) from the Natural Language Processing literature. The paper discusses the link between the word2vec method and the underlying dynamic optimization problem of the agent.

1 Introduction

In many areas of economic research, especially those considering discrete choice problems, consolidating various options into manageable categories is essential. These groupings are crucial for tasks ranging from assessing labor market dynamics to quantifying a firm's market influence and understanding the impact of neighborhoods on subsequent life outcomes. How choice alternatives are defined and measured is pivotal for both the estimation and interpretation of the results. For instance, when evaluating labor market power, determining which firms are seen as equivalent from a worker's perspective can change the calculated average market power. Similarly, defining what constitutes a "neighborhood" is essential when assessing its effect on future outcomes.

Categorization and dimension reduction of choices alternatives are also pivotal for computational efficiency. For instance, since Rust (1987), economists have estimated dynamic discrete choice models where forward-looking agents select from multiple discrete options each period. These models are beneficial across various economic fields for welfare calculation and counterfactual computation. However, estimation is computationally expensive, with costs rising with the number of options available to the decision-maker. To address the computational challenge, researchers often group similar options using heuristic and ad-hoc methods, but these decisions are not innocuous and can affect the policy implications of the analysis.

In other cases, reducing the set of alternatives can also improve precision and reduce statistical noise. For instance, when estimating the influence of workers' employment history on wage inequality using an AKM-type regression, researcher need to control for each worker fixed effect and each possible employment history. Including a dummy variable for every possible employment history is not only computationally challenging but might offer limited observations for effect evaluation, leading to potentially imprecise and biased estimates (Bonhomme et al. (2023)). One solution is clustering similar firms, which have roughly the same effect, but setting criteria for such aggregation is non-trivial.

Traditionally, researchers group categories using predefined classifications, such as NICAS codes or the Metropolitan Statistical Area. In other cases, researcher define groups ad-hoc based on observable characteristics and heuristics. These traditional methods pose several challenges: (i) They might neglect crucial unobserved heterogeneity that's vital for decision-makers. (ii) There's potential for measurement error. (iii) Researchers could introduce bias when identifying significant heterogeneity facets. This paper suggests using the Word2Vec algorithm, common in Natural Language Processing, with K-Means, to group discrete options based on an agent's choice sequence. In the text analysis literature, the Word2Vec algorithm is used to find vector representations of words that keeps their semantic meaning. These vectors have proven useful in measuring similarity among words, where words with similar meanings are placed closer together in vector space. The idea behind the algorithm is to train a neural network to predict which words are going to appear in the text nearby. If two words are likely have the same words near them, then the algorithm will assign them a similar vector representation. We adapt this algorithm for cases where we observe a sequence of choices. The idea is that if two choices are perceived similarly by the agents, then agents are likely to make similar decisions throughout their sequence of choices. For instance, if we observe that a large proportion of workers in the High-Tech industry follow these two sequences:

$$Amazon \rightarrow Facebook \rightarrow Microsoft$$
$$Amazon \rightarrow Netflix \rightarrow Microsoft$$

Then we might infer that workers view Netflix and Facebook as equivalent, suggesting their current benefits are comparable and they play similar roles in the workers dynamic decisions. To measure their similarity, we estimate the probability that we observe Amazon or Microsoft, given that we know that someone works at Facebook. We then do the same for Netflix. If the probabilities are the same, this suggests that the two firms are equivalent from the agent's point of view. In section 2 we discuss micro-foundation of these intuitions using the dynamic optimization problem of the agents.

In addition to the literatue of word2vec (Mikolov et al. (2013b),Mikolov et al. (2013a)), this paper builds on the extensive literature on discrete choice estimation (McFadden, 1974, Hotz and Miller, 1993, Rust, 1987, Keane and Wolpin., 1997). These studies underscore the idea that an agent's actions reveal their preferences. As a result, choice data can be used to deduce an agent's utility, information set, and other relevant parameters. Additionally, this paper contributes to the emerging economics literature on grouping/clustering and uncovering latent structures (Bonhomme and Manresa., 2015, Bonhomme and Manresa, 2017, Saggio, 2012, Su and Phillips, 2016). Notably, recent works have advocated for the use of Machine Learning algorithms in firm clustering. Fogel (2022) employs a static equilibrium model of the labor market combined with a bipartite stochastic block model from the community detection literature to cluster similar workers and firms. In related papers, Vafa et al. (2023) and Vafa et al. (2022) explore the causes of wage disparities between genders, accounting for job history using a transformer-based model that learns representations of job sequences. While our work bears similarities to theirs, we employ a different algorithm and provide a unique rationale for the clustering.

The paper is organized as follows: Section 2 outlines the agent's dynamic problem, introduces the Word2Vec algorithm, and discusses its application in identifying similar choices. Section 3 conducts simulations to assess the algorithm's performance, and Section 4 provides a conclusion and summarizes the paper's findings.

2 The Dynamic Problem and the Word2Vec Algorithm

2.1 The Dynamic Problem of the Agent

Let us first start by considering an agent who faces a finite dynamic discrete choice problem. At each period the agent chooses $j \in \mathcal{J}$, from a finite set of available options. The Bellman equation for the agent problem at period $t < T$ is given by

$$V_t(H_{t-1}) = \max_{j \in \mathcal{J}} u(H_{t-1}, j, \epsilon_{j,t}) + \beta \mathrm{E}\left[V_{t+1}(H_{t-1}, j) | H_t^i, j)\right]$$

and for period T, the value function is given by

$$V_T(H_{T-1}^i) = \max_{j \in \mathcal{J}} u(H_{T-1}^i, j, \epsilon_{j,T})$$

where $\epsilon_{j,t}$ is a preference shock for choice j, at period t, and $H_t^i = (J_1, , , , J_{t-1})$ is a $t-1$ vector, containing the $t-1$ choices the agent made up until period t. This classic formulation of the discrete choice problem captures a wide variety of economic problems and is used extensively in the Macro, Industrial Organization and labor literature.

I assume that there is a small finite number of distinct types $\theta \in \mathcal{T}$, where $|\mathcal{T}| << |\mathcal{J}|$ such that the agent is indifferent between any two options with the same type, given their choice sequence until period t. Specifically, let $G : j \to \theta$, then the agent's preference shock is over types, $\theta \in \mathcal{T}$, and not over $j \in \mathcal{J}$, i.e. $G(j) = G(k) \iff \epsilon_{j,t} = \epsilon_{k,t} = \epsilon_{\theta,t}$ [1]. Also, I assume that if two options have the same type then

$$u(H_{t-1}, j, \epsilon_{j,t}) + \beta \mathrm{E}\left[V_{t+1}(H_{t-1}, j, \epsilon_{j,t}) | H_{t-1}, j\right) = u(H_{t-1}, k, \epsilon_{k,t}) + \beta \mathrm{E}\left[V_{t+1}(H_{t-1}, k, \epsilon_{k,t}) | H_{t-1}, k\right)$$

The assumptions concerning types suggest that when faced with two options of the same type, workers display indifference. Consequently, when presented with two options from the same type, agents would choose one of the options at random. To illustrate, consider a worker in the High-Tech sector deciding on an employer. The worker might view Facebook and Netflix as equivalent because both offer similar wages and have comparable impacts on future career prospects. In this

[1] We can relax this assumption, and allow for shocks over the specific js, that are uncorrelated with $\epsilon_{\theta_i,t}$

instance, we would categorize both firms as indistinguishable choices belonging to the same type. In the neighborhood context, when individuals contemplate relocating after having children, ”types” would represent areas offering similar utility values to those with analogous residency histories.

2.2 The Word2Vec Algorithm

Word2Vec (Mikolov et al., 2013b, Mikolov et al., 2013a) is an immensely popular algorithm aimed at converting words, which are distinct in nature, into vector representations that carry semantic meaning. This algorithm has been used extensively in tasks requiring natural language processing and has shown great success (Goldberg, 2017). As we'll see below, the general idea behind this algorithm is to build and train a predictive model such that the model parameters are used as the vector representation of the word.

Before diving into the mechanics of the algorithm, let's first consider an example. Assume we have the following sentence in our data:

$$\underbrace{\text{Economic research}}_{\textit{Context}}\ \underbrace{\text{changes}}_{\textit{Target}}\ \underbrace{\text{the world}}_{\textit{Context}}$$

To represent the words in that sentence as vectors, we will train a neural network with a single hidden layer, to predict the probability of observing the context words, given the target word. x‘The resulting probabilities will inform us how likely it is to find each word in our data next to the target word. Therefore, in our example, we are going to train the neural network using the word $\{changes\}$ as input and the context words $\{Economic, research, the, world\}$ as output. After iterating over all the words in our data, the neural network will provide predictions on how likely it is to find a particular word given our target word. For instance, if our neural network performs well, we would expect to find that ”Research” has a higher probability of appearing next to ”Economics” than the word ”potatoes”, i.e., $\hat{\mathrm{P}}(Research|Economics) > \hat{\mathrm{P}}(Potatoes|Economics)$. After training the neural network, we won't use the model predictions but will instead use the neural network parameters to represent our words as vectors, as we'll detail below.[2].

[2] While not formally stated, the intuition behind the algorithm's success in practice is based on the ”Distributional Hypothesis”, which suggests that ”words that occur in the same contexts tend to have similar meanings” (Sahlgren, 2008)

Now, let's discuss the algorithm more formally. Let $\mathcal{C}$ be the "Corpus", the set of unique words in our data. Let w be the window size around the target word (this defines the number of words we consider as context; for instance, in the sentence above, the window size would be 2). Let $x_i \in \mathbb{R}^{|C|}$ be an indicator vector, where each component corresponds to a unique word in our corpus. This vector has a value of 1 at the position of word i and 0 elsewhere. Let n be the size of the vector representation. Let $\mathcal{W} \in \mathbb{R}^{|C|\times n}$ and $\mathcal{V} \in \mathbb{R}^{n\times|C|}$ be two matrices. The figure below, taken from Mikolov et al., 2013b, illustrates the architecture of the Word2Vec algorithm. The neural network takes the input vector, x_i, and projects it into a lower-dimensional vector of size n using $\mathcal{W}$. We then project back into a vector of size $|C|$ using $\mathcal{V}$ to generate the predicted probabilities.

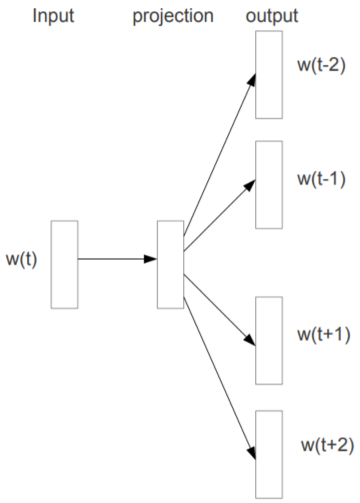


Figure 1: The Word2Vec Neural Network Architecture

The algorithm can be described using the following steps:

1. For each word in the data, generate an indicator vector $x_i \in R^C$. For example, in the aforementioned sentence, the vector has a 1 at the position of the word "changes" and 0 everywhere else.

2. Multiply the vector by $\mathcal{W}$ to generate the vector $x\mathcal{W} = w_i \in R^n$.

3. Multiply $w_i \times \mathcal{V} = S \in \mathbb{R}^V$ to generate the score vector.

4. Convert the score vector into a probability vector, $\hat{y} = (\hat{y}_{c-w}, \hat{y}_{c-w+1}, ..., \hat{y}_{c+w-1}, \hat{y}_{c+w})$, using the logistic function

$$\hat{y}_{c-k} = \frac{exp(w_i^T \mathcal{V}_k)}{\sum_{o=-w}^{w} exp(w_i^T \mathcal{V}_k))}$$

where $\mathcal{V}_k$ corresponds to the context word k.

5. Generate the output vector $y = (y_{c-w}, y_{c-w+1}, ..., y_{c+w-1}, y_{c+w})$, where y_i is an indicator vector that takes the value 1 at the position of word i. In the example sentence above, y_{-2} takes a 1 for the position of the word ”economic” and 0 elsewhere.

6. Multiply the probability vector by y to derive the cross-entropy objective function:

$$H(\hat{y}, y) = \sum_{k=c-w}^{c+w} y_k log(\hat{y}_k)$$

7. Update the model parameters (the matrices $\mathcal{W}$ and $\mathcal{V}$) using back-propagation.

There are several noteworthy points. First, the cross-entropy objective function closely resembles the regular maximum likelihood estimator, especially when estimating a multinomial logit model. This allows us to interpret the neural network as a Maximum Likelihood Estimator aiming to maximize the probability of observing a sentence, or minimize the Kullback–Leibler distance to the true underlying distribution. Second, at step 2, we multiply the indicator vector by $\mathcal{W}$, which gives us row i of the matrix. This means that each row in $\mathcal{W}$ corresponds to a unique word, and this serves as our vector representation for that word. If two words have identical adjacent words given a fixed $\mathcal{V}$, their cross-entropy objective functions would match. Hence, the neural network would assign them identical values, as these row vectors would be multiplied by matrix $\mathcal{V}$ to yield similar probabilities. The number of similar words adjacent to a word provides a significant distance metric between words.

From a computational perspective, step 4, which involves using the logistic function to normalize the output over a large vocabulary, can be particularly resource-intensive. This vocabulary can encompass millions of words, making the training process slow. To alleviate this, we use ”Negative Sampling”. In negative sampling we use ”positive” and ”negative” pairs. A positive pair consists of a word and its genuine context word observed in the data. Conversely, a negative pair is a word combined with a random word that isn't its genuine context in the data. The idea behind negative sampling is to restrict the updates to only a subset of word vectors during training. Specifically, for each positive word-context pair, a few ”negative” word-context pairs are selected at random. The negative pairs are chosen based on word frequencies in the corpus, utilizing a distribution in which word frequency is raised to the power of 3/4. This method ensures a balance between sampling frequently occurring and rarer words. The primary goal is to adjust the model parameters to increase the probability for genuine (positive) pairs and decrease it for the randomly selected (negative) pairs. Such selective updating dramatically reduces the computational burden, making training with expansive vocabularies both feasible and efficient.

Finally, word2vec algorithm described above version is termed Skip-Gram. Another version called Continuous Bag-Of-Words (CBOW) exists, where the input and output of the neural network are switched, trying to predict a target word based on adjacent words. The focus hereafter will be on the Skip-Gram version of the Word2Vec algorithm.

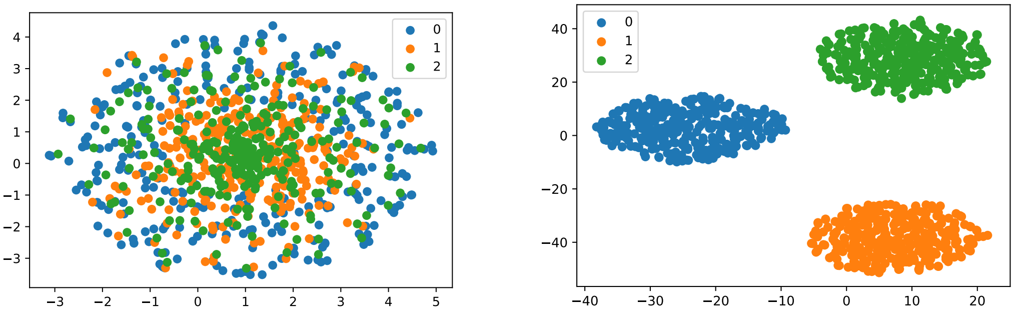
2.3 Using the Word2Vec Algorithm to uncover the Latent Discrete States

After reviewing the Word2Vec algorithm and the agent's dynamic problem, we can explore how the two can be integrated to uncover latent types. We begin by creating a vector representation for the agent's choices. To achieve this, it's helpful to consider the agent's potential choices as ”words” and a sequence of choices as a ”sentence”. We can then compile a ”corpus” of all possible choices our agents might make and input these choice sequences into our neural network. This allows us to generate vector representations of the choices within a lower-dimensional vector space. Subsequently, we can introduce these representations into a clustering algorithm, such as K-Means or DBSCAN. These algorithms will cluster different choices based on their proximity in this vector space.

It's worth noting that this clustering approach will be effective if the neural network representation assigns similar values to analogous choices and distinct values to dissimilar choices. This principle is illustrated in Figure 2[3]. Each dot in this figure represents an option available to an agent, with colors indicating the types of choice. In 2a, we observe that the Word2Vec algorithm struggled to differentiate the three groups. Under these circumstances, it's improbable that even an advanced clustering algorithm could discern the latent structure effectively. Conversely, in Figure 2*b*, the

[3] For graphical visualization, I projected the output of the Word2Vec into a 2-dimensional space using the T-SNE algorithm

algorithm adeptly clustered options of the same type while keeping those of different types separate. As a result, nearly any clustering algorithm would be proficient at segregating the options into distinct groups.



(a) The algorithm is not doing well

(b) The algorithm is doing well

2.4 When could we expect the algorithm to perform well

It is paramount to consider under which conditions the algorithm will create a representation that captures the similarity between options. Following our discussion in section 2.2, we understand that the algorithm should produce two similar representations for two words if the context words are alike. More formally, assume that for two words i and k we have $\Pr(w_j|w_i) \approx \Pr(w_j|w_k)$ for every word j, then we can't improve upon assigning them the same representation[4] Thus, if we implement the Word2Vec algorithm on the agent problem, we need options of the same type to be positioned in a sequence, adjacent to similar choices. In other words, for the algorithm to function effectively, the probability of choosing option j, given the target choice i, $\Pr(j|i) \approx \Pr(j|k)$ should hold if $G(i) = G(k)$ for all $j \in \mathcal{J}$.

This might not always hold true, especially if the difference between types is subtle and can be discerned only when observing the granular joint distribution of choices. For instance, there might

[4] Bear in mind that there's typically no guarantee of unique representations, or that two items from the same category will share a representation. However, when matrix $\mathcal{V}$ is full rank, which is a plausible assumption for randomly initialized parameters, and the corpus size exceeds n, then items with different conditional probability vectors $P = P(j|i)_j$ will have distinct lower-dimensional representations. This is because there's only one solution, up to scale, to $P = softMax(Vvec)$. It's also worth noting that scale isn't a concern, as in most applications vectors are normalized to unit length.

be situations in our data where two types exhibit the same $\Pr(j|i)_j$, but there exists an option k for which $P(k,j|i)$ differs. Therefore, we can differentiate the types only if they produce enough variation that the pairwise joint distributions capture. We thus introduce the following assumption:

Assumption 1 (sufficient variation)**.** *For any two options i and k where $G(i) \neq G(k)$, there exists at least one j such that $P(j|i) \neq P(j|k)$.*

Under which conditions would this be the case? Consider a simplified example of the agent problem, discussed in section 2.1 where the taste shocks are additive $u_\theta(H_{t-1},\theta,\epsilon_j) = u(H_{t-1},\theta) + \epsilon_\theta$. Additionally, assume that the current payoff function is separable in its random component, $\epsilon_{G(i),t}$, and consider a window of size 1 (thus, we examine what option the agent chose before and after the current choice). In this context, we know that the algorithm would assign two options similar values if their conditional probabilities align. Let $P(j_t|i)$ denote the probability that the agent chose j at period t, given that they chose i in the current period. Then, the the conditional probability of selecting option j given option i is

$$\Pr(j|i) = P(j_{+1}|i) + P(j_{-1}|i)$$

Writing explicitly $\Pr(j_{+1}|i)$

$$\Pr(j_{+1}|i) = \sum_t \Pr(i_t) \sum_{H_{t-1}} \Pr(j_{t+1}|i_t, H_{t-1}) \Pr(H_{t-1}|i_t)$$

where

$$\begin{aligned}\Pr(j_{t+1}|i_t, H_{t-1}) = \Pr(j|G(j)) \Pr \Big(& u(H_{t-1}, G(i), G(j)) - u(H_{t-1}, G(i), G(k)) \\ & + \beta \mathrm{E}\left[V_{t+1}(H_{t-1}, G(i), G(j)\right] - \beta \mathrm{E}\left[V_{t+1}(H_{t-1}, G(i), G(k)\right] \\ & > \epsilon_{G(k),t} - \epsilon_{G(j),t}, \ \forall k | H_{t-1} \Big)\end{aligned}$$

where $\Pr(i_t)$ represents the probability that option i is chosen at period t. The expression for $\Pr(j_{-1}|i)$ is analogous. Expressing the probabilities in this manner indicates that we perceive two options as equivalent if they yield the same level of flow utility and influence future choices similarly, given a choice history.[5]

. Furthermore, it's evident that the algorithm identifies two options as identical if the agent chooses at random choose between option of the same type. Therefore, the algorithm would do better if the conditional probability vectors for each type differ significantly.

The above discussion also reveals that the size of the window used should correlate with the impact of the current choice on subsequent choices. For instance, if choices follow a Markovian process where the current period choice depends only on the previous period choice, then a concise window of one period effectively captures all the information needed to differentiate between types. However, while introducing a broader window might add more distinguishing information, it can also introduce noise. Specifically, assume we opt for a two-period window instead of a one-period window; we then have:

$$P(j|i) = P(j_{+2}|i) + P(j_{+1}|i) + P(j_{-1}|i) + P(j_{-2}|i) \tag{1}$$

In this scenario, $P(j_{+2}|i)$ is defined as:

$$\Pr(j_{+2}|i) = \sum_{j} \Pr(j_{+2}|j_{+1}) P(j_{+1}|i) \tag{2}$$

therefore, the differences in $P(j|i)$ across types is driven solely by changes in the first first period and it's sufficient to use a short, one period window. Introducing larger window would add noise that can worsen the ability of the algorithm to group similar alternatives together. A practical heuristic for determining the window's size could involve assessing the Markovian nature of choices by regressing past options against future ones, provided the data allows for this.[6]

It is also worth considering what happens in the case where there are no "types" and options are not equivalent but rather, only similar. Consider the extreme case where history does not affect

[5] Notice that for two firms of the same type we have

$$\Pr(H_{t-1}|i) = \frac{\Pr(i|H_{t-1})\Pr(H_{t-1})}{Pr(i_t)} = \frac{Pr(G(i)_t|H_{t-1})Pr(H_{t-1})}{Pr(G(i)_t)} = \Pr(\Pr(H_{t-1}|k_t),\ \forall k \in \{k : G(k) = G(i)\}$$

where we rely on the idea that options with the same type behave the same

[6] If the corpus is too large, one can sample options at varying intervals and observe how past choices predict future choices within each interval.

choices (we can get this by assuming the history effect is separable, then it would cancel out in $\Pr(j_{t+1}|i_t, H_{t-1})$), then in that case, the algorithm would place closer together options, which have similar flow utility and similar effect on future choices. In other words, closer options would be those, that if we switch between the two options, the decision maker utility does not change by much.

When might the algorithm underperform? This scenario could arise if substantial heterogeneity exists among agents. First, note that if agent heterogeneity results from individual fixed effects and the payoff function is separable in these individual fixed effects, this won't affect the choice probabilities, and the same intuitions we discussed previously apply.Second, we've consistently assumed that the agent is concerned with only one objective, which aligns with the researcher's focus. However, it's plausible that agents prioritize two components. This divergence could increase the number of firm types, even if the researcher's primary outcome might categorize fewer types. For instance, consider workers valuing both wages and amenities. If firms vary across these criteria, then a firm type could be defined by any unique combination of wages and amenities. Should the researcher focus solely on wages, not all types will prove relevant. This can pose challenges, especially with limited data, but if the data include also data on wages, this might be controlled for by naturally extending the word2vec algorithm to take wages into account as additional input.

Another potential complication stemming from heterogeneity is when different agents perceive certain firms as equivalent in contrasting ways. Consider young workers viewing McDonald's as a dead-end, lower-tier job, whereas older workers might envision roles at McDonald's executive level, classifying it as a high-tier firm. This disparity affects $\Pr(H_{t-1}|i^t)$, positioning McDonald's vector representation distant from both high-type and low-type firms.[7] If a researcher believes they can account for this heterogeneity, it might prove beneficial to execute the algorithm based on specific observables. For instance, running the algorithm separately on young and old workers' choices can reveal distinct types from each group's perspective.

[7] This dilemma mirrors challenges in representing words with multiple meanings. For example, the word "bank" holds distinct meanings in "I deposit my money at the bank" versus "I sat by the river bank."

3 Monte Carlo Simulations

3.1 The Data Generating Process

For the Monte Carlo simulation, I consider the problem of workers who need to choose a place of work. I consider two potential models - the first is a Dynamic Discrete Choice problem, where in each period the workers choose where to work. The second is a Job-Ladder model, where on each period a worker gets a job offer with some probability and chooses whether to accept or reject.

3.1.1 A Dynamic Discrete Choice Model (DC)

I consider a problem of a worker that has worked for five years. There are $|\mathcal{J}|$ number of firms, which are divided between three types of firms. Workers can affect wages through two channels. The first, is through "Current-Effect", $\alpha_{G(i)}$, which is the effect of currently working at firm i, of type $G(i)$. The second effect is a "History-Effect",$\beta_{G(i)}$, which is the effect that working in the past at firm i, that has type $G(i)$, has on the worker's current wage [8].

In this setup, each worker solves the following dynamic problem[9]

$$
\begin{aligned}
V_t(H_t^i) &= \max_j \frac{\sum_{k=1}^{t-1} b_{G(i(t))}}{t-1} + \alpha_{G(j),t} + \epsilon_{G(j),t} + \beta E[V_{t+1}(H, G(j))] \\
V_5(H) &= \max_j \frac{\sum_{k=1}^{4} b_{G(i(t))}}{4} + \alpha_{G(j),t} + \epsilon_{G(j),t}
\end{aligned}
$$

where $i(t)$ is the firm the worker chose to work at year t and $\epsilon_{G(i)}$ is distributed with type 1 extreme value for all firms types. Table 1 shows the value used for the history and current wage effects.

[8] This is a reduced form interpretation of many different economic mechanisms. One example for a History Effect can be seen, if some firms do a better job of improving the worker's human capital and skills, thereby affecting and improving the worker's future wages. Another channel under the History Effect that can affect the worker's current wage is Signaling. If their employment history carries some information on worker quality and firms pay based on expected marginal productivity, then working at different firms would translate to different wages in the future

[9] In both DGPs I abstract from worker fixed effect

3.1.2 A Job Ladder Model (JL)

I consider a Job Ladder model, which is a simplified model of the one discussed in Sorkin, 2017. This is a partial equilibrium model, with an exogenous, undirected, search, where workers also work for 5 periods and there are three types of firms and $|\mathcal{J}|$ firms. At each period, workers get a job offer with probability λ. They then decide whether to move to the new firm or stay at the current firm. I also include an exogenous separation shock, where, with probability δ, workers must accept the job offered to them (sometimes called the "Godfather Shock"), even if it means lower wages. The following Bellman equation summarizes workers behavior

$$\begin{aligned}
V_t(H,j) = {} & \underbrace{\frac{\sum_{k=1}^{t-1} b_{G(i(t))}}{t-1} + \alpha_{G(j),t} + \epsilon_{G(j),t}}_{\text{Current Period Payoff}} + \beta \Bigg[\underbrace{\lambda\delta \left(\frac{\sum_i V_{t+1}(H,j,i)}{3}) \right)}_{\text{Exogenous Moving between firms}} \\
& + \underbrace{\lambda(1-\delta) \left(\frac{\sum_i \max\{V_{t+1}(H,j,j), V_{t+1}(H,j,i)\}}{3} \right)}_{\text{Endogenous Moving between firms}} \\
& + (1-\lambda) V_{t+1}(H,j,j) \Bigg] \\
V_5(H) = {} & \underbrace{\frac{\sum_{k=1}^{t-1} b_{G(i(t))}}{t-1} + \alpha_{G(j),t} + \epsilon_{G(j),t}}_{\text{Current Period Payoff}}
\end{aligned}$$

where $\epsilon_{G(i)}$ is distributed with type 1 extreme value for all firms types. As implied by the Bellman equation, I use the same payoff function used in the dynamic discrete choice DGP. Notice that in this setup that if the probability of λ decreases, or the probability of δ increases, the choices made by the agents are less revealing on type of the firm. In the extream case where $\lambda = 0$ the worker would not switch jobs at all and the choice sequence would not contain any information on firms type. In that case we would expect for the Word2Vec algorithm to preform badly. Table 1 shows the parameters used in the simulations.

	Discrete Choice	Job Ladder
β_1	0	0
β_2	1	1
β_3	-1	-1
α_1	0.5	0.5
α_2	0.7	0.7
α_3	3	3
λ		0.2
δ		0.1

Table 1: DGP Parameters

3.1.3 Simulating Panels

For both DGPs I solve for the value functions using backwards induction, and simulate data for each worker as follows

1. Draw $\epsilon_{\theta,0}$ for each firm type
2. Let the worker choose between the types of firms, given the value functions, the shocks and whether the worker can move
3. Calculate the current worker wage
4. repeat for five periods

After simulating a panel of workers and firm types, draw $|J|$ firm IDs and assign them at random for different worker spells, to generate a panel with workers, firm IDs and wages.

3.2 Evaluation Criteria

To demonstrate the performance of the method, I consider three accuracy criteria. In the first criterion, I calculate the share of firms-pairs who were correctly classified. More formally, Let $\mathcal{P} = \{\{i, j\}\}$, be the set of $\binom{|\mathcal{J}|}{2}$ pairs of firms, then for each pair I check whether the classifier included these two firms at the same class or not and report the average success rate. More explicitly, the first accuracy criterion is given by

$$S_1 = \frac{1}{|\mathcal{P}|}\sum_{\mathcal{P}}\left(\mathbb{1}\{G(i) = G(j) \text{ and } \hat{t}(i) = \hat{t}(j)\} + \mathbb{1}\{t(i) \neq t(j) \text{ and } \hat{t}(i) \neq \hat{t}(j)\}\right)$$

The second criteria measure the share of correctly classified pairs, condition on a pair being of the same type. Specifically, let $\mathcal{P}_1 = \{\{i, j\}|G(i) = G(j)\}$, then the second accuracy criterion is given by

$$S_2 = \frac{1}{|\mathcal{P}_2|}\sum_{\mathcal{P}_2}\mathbb{1}\{\hat{G}(i) = \hat{G}(j)\}$$

Last, I consider a similar measure of success, where we ask what the share of pairs who were assigned the same type, are correctly classified. Let $\mathcal{P}_3 = \{\{i, j\}|\hat{G}(i) = \hat{G}(j)\}$, then our last accuracy criterion is given by

$$S_3 = \frac{1}{|\mathcal{P}_3|}\sum_{\mathcal{P}_3}\mathbb{1}\{G(i) = G(j)\}$$

3.3 Results

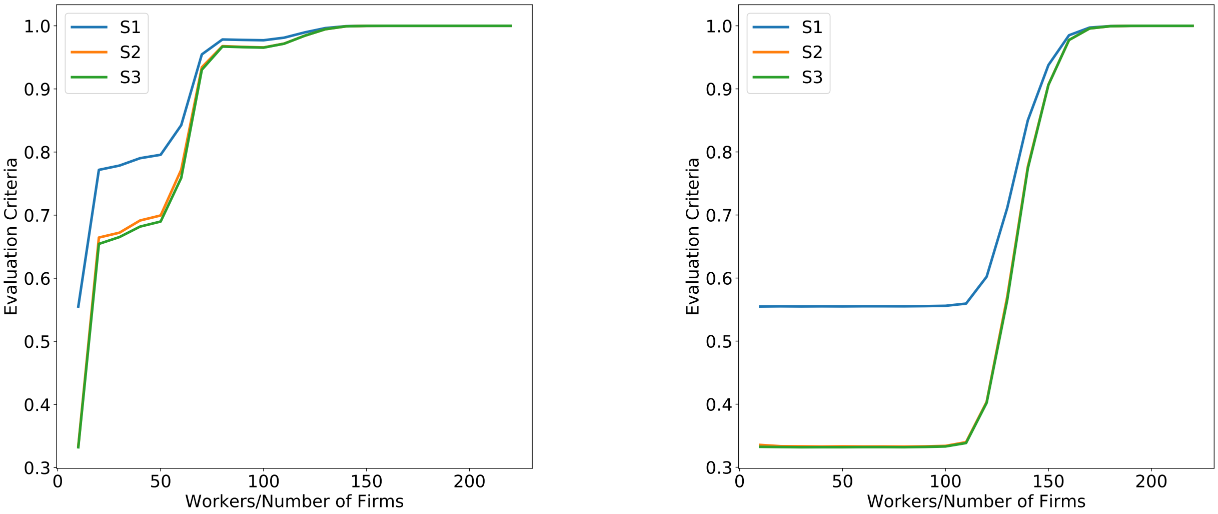
I simulate the data using the two DGPs as detailed in section 3.1.3. I use the Gensim package in Python to produce vector representations, while $K - mean$ clustering from the Sklearn package in Python to cluster firms. For the Discrete Choice model, I simulate a panel of $63,000$ workers and 900 firms, distributed equally over three types. I applied a window of 2, set the representation vector size to 100, and trained the neural net for 10 epochs. For the Job Ladder model, the behavior of $126,000$ workers was simulated. Other parameters remained consistent with the Discrete Choice model. For K-mean clustering, 3 clusters were used, with cosine similarity measuring the distance between two vectors. I used 10 starting values and repeated each exercise 50 times, reporting the average success rates[10].

Figures 3a and 3b demonstrate how the three evaluation criteria vary with an increased ratio of workers to the number of firms. For smaller ratios, clustering is suboptimal, and classification is almost random. In the DC scenario with a ratio of 10, the proportion of firm-pairs of the same

[10] 50 repetitions were due to time constraints. The results remained consistent across the Monte Carlo simulations

type classified similarly by the algorithm is around 33%. Given three clusters, this suggests almost random firm grouping. Increasing the worker-to-firm ratio significantly improves algorithmic results. In the DC model, a ratio of around 70 results in nearly 100% accurate classification by all evaluation criteria. In contrast, the JL model requires a ratio close to 150. Both ratios are considerably high, and in a Card and Kline, 2013 paper, the average worker count per firm is approximately 16. Given the DGPs considered, the algorithm would perform poorly at this ratio.

Next, we examine how the algorithm's performance changes when we adjust some of its parameters. We start with training time. Figure 6 in the appendix displays the results of extending training time (epochs)[11]. As expected, a longer runtime improves the algorithm's job separations into the correct groups. However, minimal improvement is observed after increasing epochs beyond 20.



(a) Discrete Choice DGP

(b) Job Ladder DGP

Figure 3: A figure with two subfigures

[11] Epochs represent the number of times the algorithm goes over the entire sample, adjusting the parameters to minimize the objective function. An epoch of 10 means the Word2Vec algorithm used each observation 10 times to update model parameters

Vector Size	10	50	100	300	600
S1	0.72	0.78	0.95	0.99	0.99
S2	0.83	0.67	0.92	0.99	0.99
S3	0.55	0.67	0.92	0.99	0.99
Window Size	1	2	3	4	5
S1	0.998	0.95	0.80	0.79	0.78
S2	0.997	0.93	0.70	0.68	0.67
S3	0.997	0.92	0.70	0.68	0.67
Number of Clusters	3	6	9	15	30
S1	0.95	0.75	0.71	0.69	0.68
S2	0.93	0.39	0.24	0.14	0.07
S3	0.92	0.72	0.69	0.67	0.66

Table 2: Sensitivity test - Discrete Choice Model

Vector Size	10	50	100	300	600
S1	0.56	0.61	0.85	0.68	0.62
S2	0.33	0.41	0.78	0.53	0.43
S3	0.33	0.41	0.77	0.52	0.42
Window Size	1	2	3	4	5
S1	0.56	0.86	0.99	1.00	1.00
S2	0.33	0.79	0.99	1.00	1.00
S3	0.33	0.78	0.99	1.00	1.00
Number of Clusters	3	6	9	15	30
S1	0.85	0.71	0.68	0.67	0.66
S2	0.77	0.33	0.19	0.10	0.05
S3	0.77	0.64	0.55	0.50	0.46

Table 3: Sensitivity test - Job Ladder Model

Tables 2 and 3 illustrate the model's ability to uncover the types based on different model parameters. Through testing the algorithm, it's clear that the worker/firm ratio is a crucial data feature. If this ratio is sufficiently large, the algorithm consistently identifies the underlying types. To assess the model's sensitivity to this ratio, I use a ratio of 70:1 for the DC DGP and a ratio of 140:1 for the JL DGP. Both ratios result in good, though not perfect, clustering. In 2, for the DC scenario, enlarging the representation vector (n) enhances the algorithm's accuracy. When the size grows from 50 to 100, the likelihood of a firm-pair being classified correctly increases from about 77% to 94%. Likewise, the chance that two firms classified as the same type by the algorithm are genuinely of the same type rises from 66% to approximately 91%. In contrast, table 3 shows that increasing the vector size doesn't always yield a linear boost in accuracy. There's an optimal size close to 100, possibly due to over-fitting, where similar firms are positioned distantly in the vector space.

We then assess the impact of window size on model accuracy. In the DC DGP, increasing the window size doesn't enhance classification; instead, it degrades it. When we consider two selected firms, the classification accuracy by the algorithm drops from 99.8% for a window size of 1 to about 78% for a window size of 5. This decline is likely because in the DC model, history ”cancels out” in choice decisions, rendering a window of size 1 as the optimal choice. A larger window seems to introduce noise, thereby reducing the algorithm's performance. Conversely, table 3 portrays a different trend. Here, a larger window size aids in better firm classification. The S1 criteria shifts from a low 0.55%, indicating a near-random type classification, to an impressive 99.9% when the window size is expanded to 5. This suggests that the entire sequence is taken into account for each target word. Such an observation aligns with our understanding: in the JL model, history assumes a non-additive role and does not "cancel out." Thus, a longer window provides more information about the type than a mere 1-period window. The importance of longer sequences in the JL model is all also driven by the fact that the worker's choice is largely deterministic. Only about $0.2 \times 0.9 = 18\%$ of the instances involve them actively making a choice. This suggests that a brief glance at the immediate environment of each choice doesn't offer substantial insight into the underlying types.

Next, we can explore how the the accuracy of the model changes if we increase the number of clusters and keep the number of firms fixed. The results in table 2 and table 3 imply that in both DGPs, increasing the number of clusters does not result in better accuracy[12]. In the DC DGP, S1 accuracy falls from 95% to 68%, as we increase the number of clusters from 3 to 30. Similarly, in the JL DGP the S1 criteria falls from 85% to 66%.

Last, we can consider how the model accuracy changes with respect to changes in λ in the JL DGP. As discussed earlier, as λ increase, the agent choice sequence is more revealing on the true types. For example, consider the case where $\lambda = 0$, in that case, workers choose their first job and do not move at all. In this case the choice sequence is not going be revealing at all on the underlying types. Figure 4 explores the model accuracy as function of λ. As we can see, when $\lambda < 0.15$ the model classification is as good as random, where S1 criteria is around 55% and the S3 criteria is at 33%. Once we reach 0.2, which implies at around 18% of the time workers can choose whether to move or not, the algorithm performance start to increase, where when λ is at 0.3 (workers are allowed to choose where to work $0.3 \times 0.9 = 27\%$ of the time), the model is able to better uncover the underlying types.

[12] Notice that S2 falls sharply, but this is expected, as a larger number of clusters implies that the probability of two firms of the same type, to be in the same cluster decreases mechanically

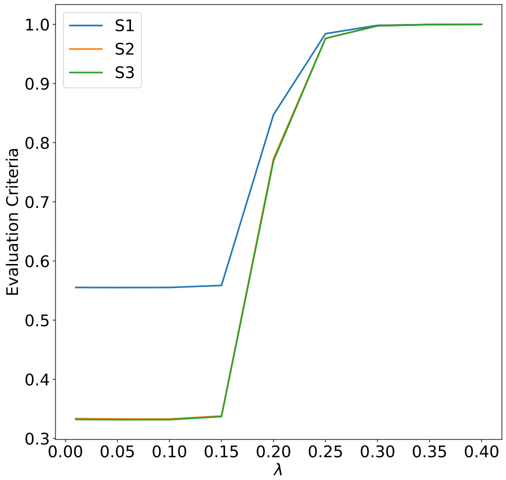


Figure 4: Accuracy in the Job Ladder Model with respect to λ

3.4 Application to AKM

In this section we want to explore whether the model could improve estimation of the AKM (Abowd and Margolis, 1999) type specification of the JL model. Specifically, I use the data generate from the JL model, and run a regression with firm fixed effect on wages in the fifth year. [13]

$$w_{i,5} = \phi_{j(i)} + \epsilon_{i,5}$$

I then calculate the variance and the firms fixed effect and compare it the true variance (which is given the by $(\text{Var}(\alpha_{G(i)}))$. I then run a second regression, where I add control for the type of firms the worker worked in years prior to year 5.

$$w_{i,5} = \phi_{j(i)} + \beta_1\hat{\theta}_1 + \beta_2\hat{\theta}_2 + \hat{\theta}_3\theta_3 + \epsilon_{i,5}$$

where $\hat{\theta}_i$ is the number of years the worker worked at firm of type i, where I use the method discussed in this paper to generate the types. Seeing as we know that in the JL model employment history plays

[13] As mentioned before, I abstract from worker heterogeneity

a significant role in determining wage levels, we would expect that the standard fixed effect regression would provide bias estimates of the variance fixed effect. On the other hand, if our algorithm is doing a good job to undercover the types, then controling for the employment history would allow us to recover the true variance of firms fixed effect.

Figure 5 shows the the average FE variance from 50 repetitions of the Monte Carlo exercise. Looking at the first figure on the left, we can see that as λ is low, the variance of the FE estimator is biased downwards and controlling for the types of firms the worker worked for does not improve over the baseline standard FE estimation. As we've seen in the previous section, this is due to the fact that when λ is very low, the Word2Vec algorithm does not preform well. Once λ reaches a critical point, and the Word2Vec algorithm preforms as expected, we can see that the estimated variance of the fixed effect matches the true variance. We can see that we get similar results, while increasing the ratio between the number of workers and the number of firms.

Last, we see that increasing the number of clusters in the K-Means algorithm decreases the quality of the control, as the algorithm accuracy falls, as we've seen in the previous section.

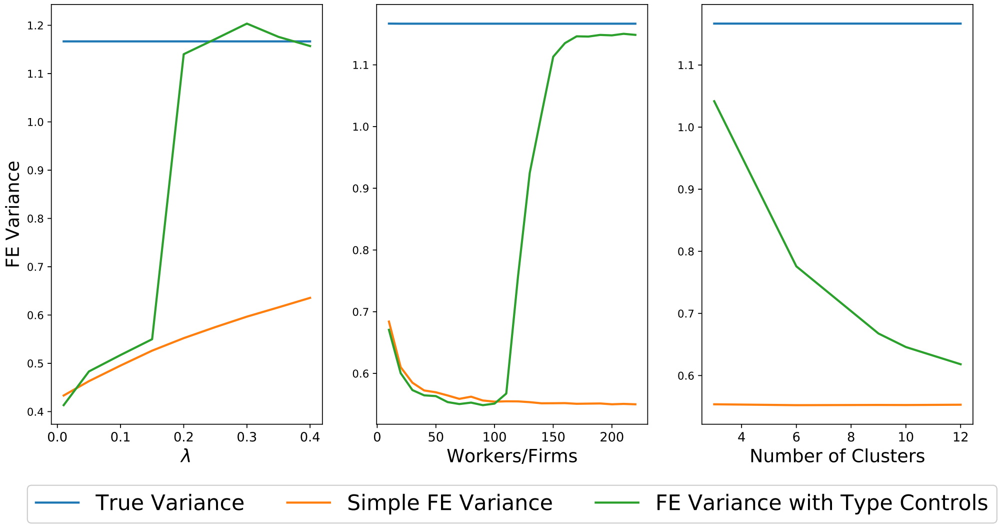


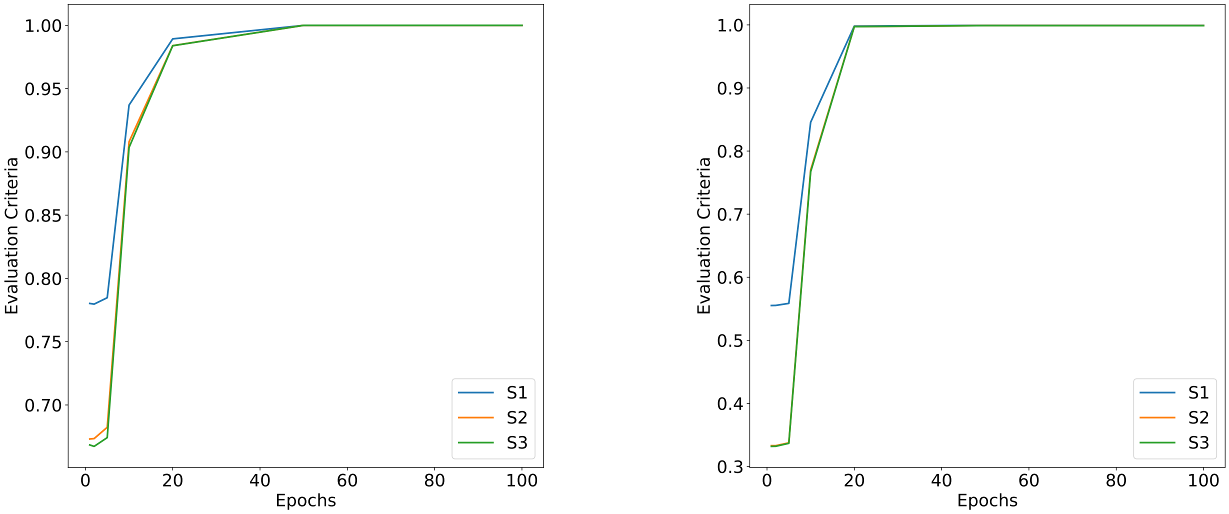
Figure 5: Firm FE Variance

4 Conclusions

In this paper, I suggest a new method for clustering based on sequences of choices. I suggest utilizing the Word2Vec algorithm, a popular algorithm that originated from Natural Language Processing research and has demonstrated success in numerous applications. The main advantages of the method suggested here are twofold. First, it provides a measure of distance in a vector space between two discrete options grounded in economic reasoning. Second, it offers a means to reduce the dimensionality of complex problems to a more manageable level, again, rooted in economic reasoning.

The primary advantage of employing the word2vec algorithm, despite the availability of newer algorithms, is its transparency and simplicity. As discussed in section 2, word2vec seeks to find a representation where two options with identical conditional probabilities have the same vector representation. This characteristic establishes a direct connection between the algorithm and numerous data generating processes (DGPs) prevalent in economics. This paper emphasizes the application of word2vec to sequential choice problems. Comparable arguments can be made for scenarios where agents in different setups make multiple choices. For instance, analyzing the assortment of products purchased by customers using the method described in this paper can offer insights into product equivalencies and substitution patterns. The algorithm's clarity and its alignment with the DGP not only set it apart from other machine learning techniques but also highlight its strengths and limitations. Understanding the algorithm's functional boundaries, namely the specific conditions and contexts where it excels, is vital in economics, where producing accurate out-of-sample predictions or counterfactuals is a primary objective. Using algorithms without accounting for the inherent DGP might deliver acceptable results with existing data but may preform worse out-sample.

A Figures



(a) Discrete Choice Model

(b) Job Ladder Modeo

Figure 6: Number of Epochs

References

Abowd, John M., F. K. and Margolis, D. N. (1999). High wage workers and high wage firms. *Econometrica*, 67(2):251–333.

Bonhomme, S., Holzheu, K., Lamadon, T., Manresa, E., Mogstad, M., and Setzler, B. (2023). How much should we trust estimates of firm effects and worker sorting? *Journal of Labor Economics*, 41(2):291–322.

Bonhomme, S. and Manresa., E. (2015). Grouped patterns of heterogeneity in panel data. *Econometrica*, 83(3):1137–1184.

Bonhomme, Stéphane, T. L. and Manresa, E. (2017). Discretizing unobserved heterogeneity. *University of Chicago, Becker Friedman Institute for Economics Working Paper 2019*, 16.

Card, David, J. H. and Kline, P. (2013). Workplace heterogeneity and the rise of west german wage inequality. *The quarterly journal of economics*, 128(3):967–1015.

Fogel, J. (2022). *What is a labor market? classifying workers and jobs using network theory.* PhD thesis, University of Michigan.

Goldberg, Y. (2017). *Neural network methods for natural language processing.* Number 1. Synthesis Lectures on Human Language Technologies 10.

Hotz, V. J. and Miller, R. A. (1993). Conditional choice probabilities and the estimation of dynamic models. *The Review of Economic Studies*, 60(3):497–529.

Keane, M. P. and Wolpin., K. I. (1997). The career decisions of young men. *Journal of political Economy*, 105(3):473–522.

McFadden, D. (1974). The measurement of urban travel demand. *Journal of public economics*, 3(4):303–328.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781.*

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, pages 3111–3119.

Rust, J. (1987). Optimal replacement of gmc bus engines: An empirical model of harold zurcher:journal of the econometric society. *Econometrica*, pages 999–1033.

Saggio, R. (2012). Discrete unobserved heterogeneity in discrete choice panel data models.

Sahlgren, M. (2008). The distributional hypothesis. *Italian Journal of Linguistics*, 20.

Sorkin, I. (2017). Ranking firms using revealed preference. *The quarterly journal of economics*, 133(3):1331–1393.

Su, Liangjun, Z. S. and Phillips, P. C. (2016). Identifying latent structures in panel data. *Econometrica*, 84(6):2215–2264.

Vafa, K., Athey, S., and Blei, D. M. (2023). Decomposing changes in the gender wage gap over worker careers. *WP*.

Vafa, K., Palikot, E., Du, T., Kanodia, A., Athey, S., and Blei, D. M. (2022). Career: Transfer learning for economic prediction of labor sequence data. *arXiv preprint arXiv:2202.08370*.