Word Embeddings Using Occupational Data

Wenhao Jiang

Department of Sociology
New York University

November 30, 2022

Word Embeddings Basics[1]

[1] Materials in these slides are partly based on Lena Viota's tutorial

Basics

- The way machine learning models "see" data is different from how we humans do
- For example, we can easily understand the text "I saw a cat", but our models can not - they need vectors of features
- Such vectors, or word embeddings, are representations of words which can be fed into your model (e.g., text classifications)

Represent as Discrete Symbols: One-hot Vectors

- The easiest way is to represent words as one-hot vectors, where the vector of the i-th word in the voluntary has 1 on the i-th dimension and 0 on the rest

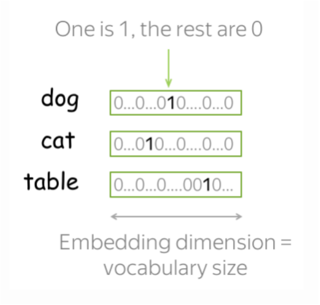


Figure 1: One-hot Vectors

Represent as Discrete Symbols: One-hot Vectors

- Shortcomings are apparent
  - 1. The Matrix can be large and sparse. Vector dimensionality is equal to the vocabulary size
  - 2. These vectors do not know the meaning of the words they represent. The word *cat* is as close to *dog* as it to *table*

- We humans understand the meaning of words from the context they share

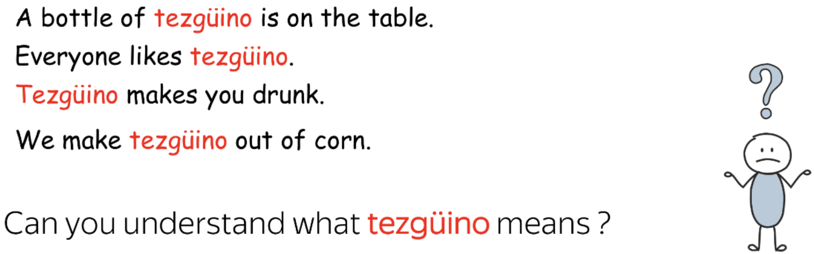


Figure 2: Distributional Hypothesis

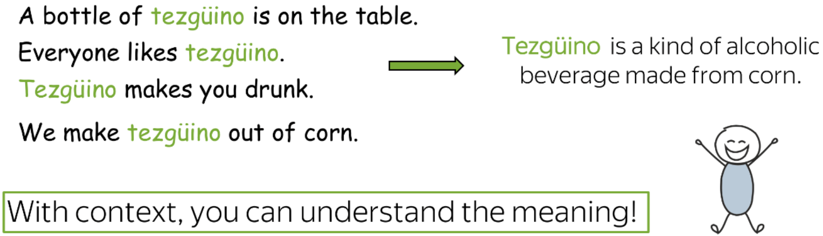


Figure 3: Distributional Hypothesis

(1) A bottle of ______ is on the table.

(2) Everyone likes ______ .

(3) ______ makes you drunk.

(4) We make ______ out of corn.

What other words fit into these contexts ?

	(1)	(2)	(3)	(4)	...
tezgüino	1	1	1	1	
loud	0	0	0	0	
motor oil	1	0	0	1	
tortillas	0	1	0	1	
wine	1	1	1	0	

← contexts

← rows show contextual properties: 1 if a word can appear in the context, 0 if not

Figure 4: Distributional Hypothesis

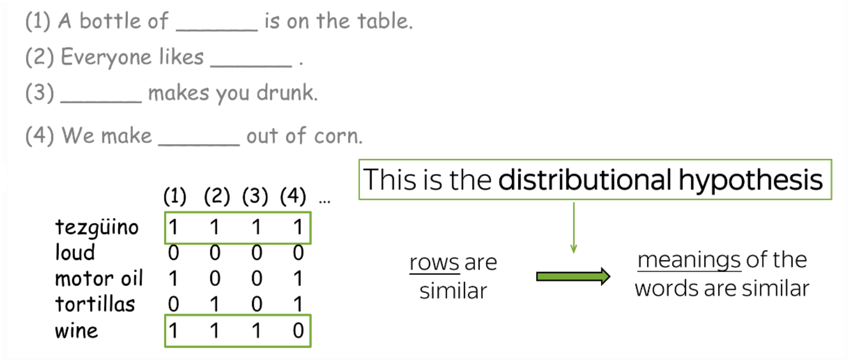


Figure 5: Distributional Hypothesis

Distributional Semantics

- The distributional hypothesis:
- The words which frequently appear in similar contexts have similar meanings
- The main idea for word embeddings to understand the meaning of the words is to put information about word contexts into word representation
- There are two main methods to take word contexts into account into word representation
  - Count-Based Method
  - Prediction-Based Method

Count-Based Method

Count-Based Method

- Main idea: Put the information about contexts into word vectors
- How: Put the information manually based on global corpus statistics

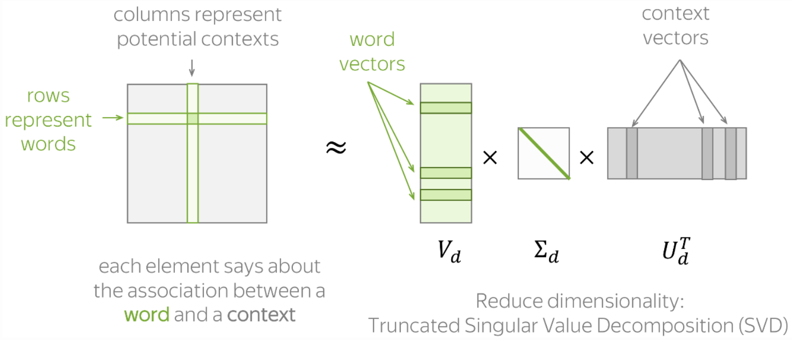


Figure 6: Singular Value Decomposition

Count-Based Method

- Any Matrix A $(m \times n)$ can be unconditionally decomposed into the product of three component Matrices $A = S\Sigma U^T$
- where S $(m \times m)$ is the matrix of the eigenvectors of AA^T, U $(n \times n)$ is the matrix of the eigenvectors of $A^T A$, and Σ $(m \times n)$ is the diagonal matrix where the diagonals are the square roots of the eigenvalues of AA^T with a descending order
- We can approximate A by $A = S\Sigma U^T \approx S_k \Sigma_k U_k^T$, where $k \ll m, n$
- The words are represented by the row vectors of the $m \times k$ matrix $S_k \Sigma_k$
- The contexts/documents are represented by the column vectors of the $k \times n$ matrix $\Sigma_k U_k^T$

Count-Based Method

- How to construct the Word-Context(Document) Matrix A?
- Co-Occurrence Counts
  - where each element $N(w, c)$ corresponds to the number of times word w appears in context c
  - with context c being the surrounding words of w in a $M-$sized window

Count-Based Method

- How to construct the Word-Context(Document) Matrix A?
- Positive Pointwise Mutual Information (PPMI)
  - $PPMI(w, c) = max(0, PMI(w, c))$, where $PMI(w, c) = \log \frac{P(w,c)}{P(w)P(c)}$
  - with context c being the surrounding words of w in a $M-$sized window

Count-Based Method

- How to construct the Word-Context(Document) Matrix A?
- Latent Semantic Analysis (LSA)
  - where A is a matrix of *Term frequency - Inverse Document Frequency* (TF-IDF)
  - $tf - idf(w, d, D) = tf(t, d) \times idf(t, D) = \frac{f_{t,d}}{\sum_{t' \in d f_{t',d}}} \times \log \frac{N}{|\{d \in D : t \in d\}|}$

Prediction-Based Method

Word Embeddings as a Prediction-Based Method

- Main idea: Put information about contexts into word vectors
- How: Learn word vectors by "teaching" them to predict contexts
- The distributional hypothesis: if vectors "know" about contexts, they "know" word meaning
- Known as *word2vec*

Word Embeddings as a Prediction-Based Method

- For each central word as represented by vector w_t in each position t,we can compute the probabilities of context words by the central word

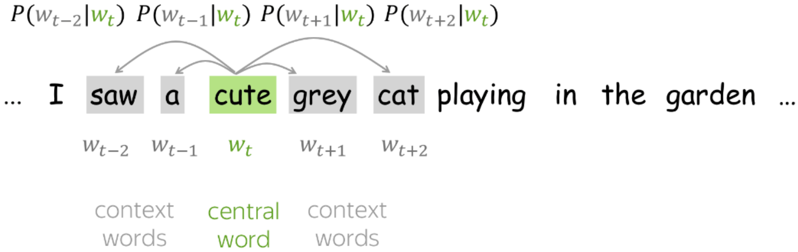


Figure 7: Illustration of Probabilities of Context Words Given Central Words

Word Embeddings as a Prediction-Based Method

- We iterate this process for each word as the central word throughout the corpus

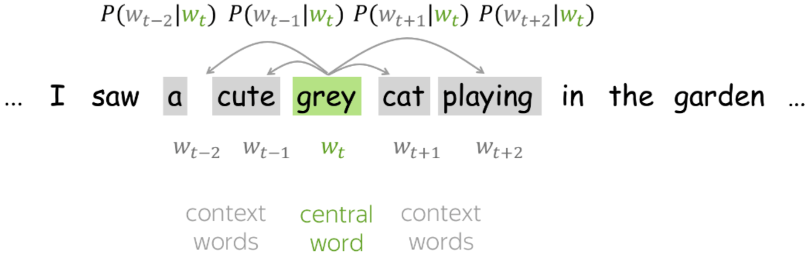


Figure 8: Illustration of Probabilities of Context Words Given Central Words

Word Embeddings as a Prediction-Based Method

- For each position $t = 1, ..., T$ in a text corpus, *word2vec* predicts context words within a m-sized window given the central word w_t:

$$Likelihood = L(\theta) = \prod_{t=1}^{T} \prod_{-m \leq j \leq m} P(w_{t+j} | w_t, \theta)$$

- where θ are all variables to be optimized
- The variables here are the vector representations of each central and context word
- This is the idea of *Skip-Gram*

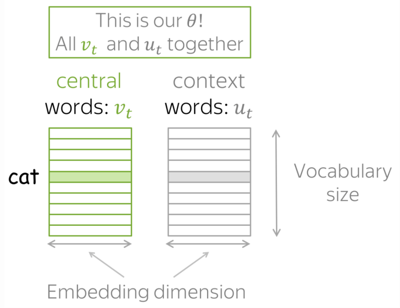
Word Embeddings as a Prediction-Based Method

- The objective loss function $J(\theta)$ is the average negative log-likelihood

$$Loss = J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m}\log P(w_{t+j}|w_t,\theta)$$

Word Embeddings as a Prediction-Based Method

- For each word w, there are two vectors
  - v_w when it is a central word
  - u_w when it is a context word
- Once the vectors are trained, Usually only central word vectors are used



Word Embeddings as a Prediction-Based Method

- How to calculate $P(w_{t+j} | w_t, \theta)$?
- For a central word c and a context word o

$$P(o|c) = \frac{\exp(\boxed{u_o^T v_c})}{\sum_{w \in V} \exp(u_w^T v_c)}$$

Dot product: measures similarity of o and c
Larger dot product = larger probability

Normalize over entire vocabulary
to get probability distribution

- This Softmax function essentially models a multi-classification task (multinomial logistic), where the number of classes are the total number of unique words

Word Embeddings as a Prediction-Based Method

- How to estimate the optimal θ (i.e. vector v_w and u_w) to minimize the loss function (i.e. to maximize the likelihood function)?
- By gradient descending with some learning rate α, a single pair of a central word and one of its context words at a time

$$\theta^{new} = \theta^{old} - \alpha \nabla_\theta J(\theta)$$

Word Embeddings as a Prediction-Based Method

- Each update is for a single pair of a center word and one of its context words

$$Loss = J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m}\log P(w_{t+j}|w_t,\theta) = \frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m}J_{t,j}(\theta)$$

- where for the central word w_t, the loss contains a distinct term $J_{t,j}(\theta) = -\log P(w_{t+j}|w_t,\theta)$ for each of its context word w_{t+j}

Word Embeddings as a Prediction-Based Method

... I saw a cute grey cat playing in the garden ...

- For the central word *cat* and the context word *cute*

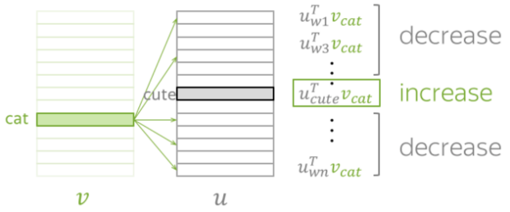
$$J_{t,j}(\theta) = -\log P(cute|cat) = -\log \frac{exp(u_{cute}^T v_{cat})}{\sum_{w \in V} exp(u_w^T v_{cat})} =$$
$$-u_{cute}^T v_{cat} + \log \sum_{w \in V} exp(u_w^T v_{cat})$$

- We therefore only update
  - v_{cat} for the central word
  - but u_w for all context words in the corpus for each update to minimize $J_{t,j}(\theta)$

Word Embeddings as a Prediction-Based Method

$$v_{cat} := v_{cat} - \alpha \frac{\partial J_{t,j}(\theta)}{\partial v_{cat}}$$
$$u_w := u_w - \alpha \frac{\partial J_{t,j}(\theta)}{\partial u_w}, \forall w \in V$$

- By making an update to minimize $J_{t,j}(\theta)$, we force the parameters to increase similarity (dot product) of v_{cat} and u_{cute} and, at the same time, to decrease similarity between v_{cat} and u_w for all other words w in the corpus

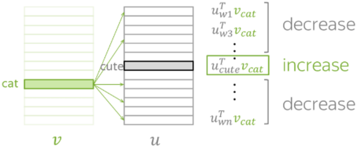


Word Embeddings as a Prediction-Based Method - Negative Sampling

- For each pair of a central word and its context word, we had to update all vectors for context words
- This is highly inefficient: for each step, the time needed to make an update is proportional to the vocabulary size
- Alternatively, we may consider context vectors not for all words, but only for the current target (*cute*) and several randomly chosen words
- This is the idea of *Negative Sampling*

Word Embeddings as a Prediction-Based Method - Negative Sampling

Dot product of v_{cat}:
- with u_{cute} - increase,
- with <u>all other</u> u - decrease

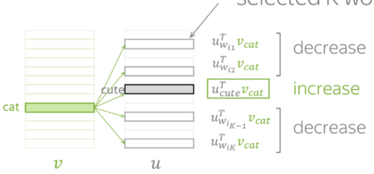


Parameters to be updated:
- v_{cat}
- u_w for all w in the vocabulary

|V| + 1 vectors

Dot product of v_{cat}:
- with u_{cute} - increase,
- with <u>a subset of other</u> u - decrease

<u>Negative samples</u>: randomly selected K words



Parameters to be updated:
- v_{cat}
- u_{cute} and u_w for w in K negative examples

K + 2 vectors

Word Embeddings as a Prediction-Based Method - Negative Sampling

- To further simplify the computation, negative sampling converts the multi-classification task into a binary-classification task
- The new objective is to predict, for any given word-context pair e.g. v_{cat}-u_{cute}, whether the word *cute* is in the context window of the the center word *cat* or not
- $P(D=1|v_{cat}, u_{cute}; \theta) = \sigma(u_{cute}^T v_{cat})$, where $\sigma(x) = \frac{1}{1+e^{-x}}$

$$\theta = \arg\max_{\theta} P(D=1|v_{cat}, u_{cute}; \theta) \times \prod_{w \in \{w_{i1}, \ldots, w_{iK}\}} (1 - P(D=1|v_{cat}, u_w; \theta))$$

- The new objective loss function becomes

$$J_{t,j}(\theta) = -\log \sigma(u_{cute}^T v_{cat}) - \sum_{w \in \{w_{i1}, \ldots, w_{iK}\}} \log(1 - \sigma(u_w^T v_{cat}))$$

Word Embeddings as a Prediction-Based Method

- Combined with the *Skip-Gram*, the above procedures represent the most common algorithm in word embeddings *Skip-Gram with Negative Sampling* (SGNS)
- There are other algorithms, such as *Global Vectors for Word Representation* (GloVe) that combines prediction with count-based method

Word Embeddings as a Prediction-Based Method

- The final outputs of SGNG are vector representations of words, typically with dimension $K = 300$
- When the corpus is sufficiently large, e.g., Google Ngram, SGNG can have some preferable features
- $\overrightarrow{king} + \underbrace{\overrightarrow{woman} - \overrightarrow{man}}_{\text{gender dimension}} \approx \overrightarrow{queen}$
- $\overrightarrow{hockey} + \underbrace{\overrightarrow{affluence} - \overrightarrow{poverty}}_{\text{affluence dimension}} \approx \overrightarrow{lacrosse}$
- The closeness and difference between any two words can be computed as the cosine similarity between the two
- Or equivalently, the euclidean distance when vectors are normalized to have length 1

Word Embeddings as a Prediction-Based Method

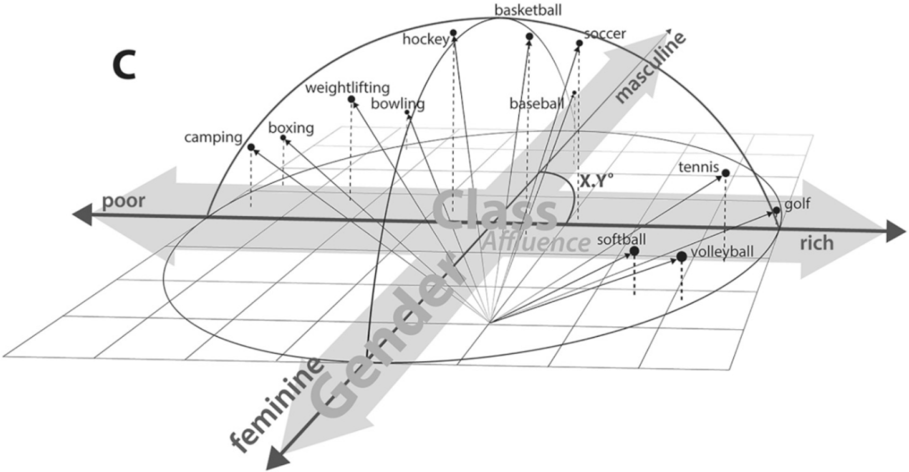


Figure 9: Kozlowski et al. (2019)

Word Embeddings as a Prediction-Based Method

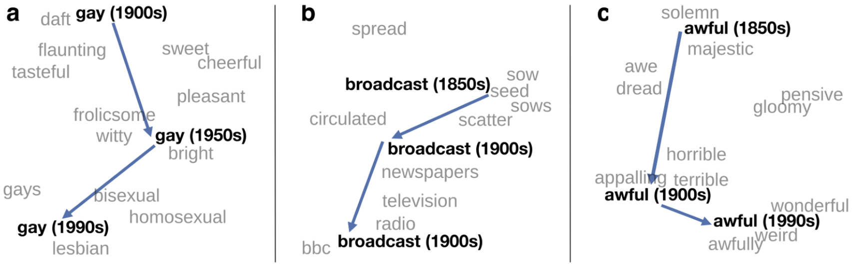


Figure 10: Hamilton et al. (2016)

Application

Application to the Studies of Occupations

- Word embeddings originate from text analysis, which seems far away from occupation and stratification studies
- But we may borrow the idea
  - A. Using vector representation of words from word embeddings as input features (Embedding Layer) in Neural Network algorithms to classify occupations
  - B. Applying Singular Value Decomposition to the Occupation-Labor Market Matrix to understand the spatial "context" of occupations and the occupational composition of labor market space
  - C. Understanding the shared understanding of occupations, such as occupation prestige and gender/race "stereotypes", from historical and contemporary texts

Word Embeddings and Text Classification

- In many cases, we want to classify texts into categories
- BLS, for example, have a NLP team who are trying to automate the process of classifying raw occupation descriptions into standardized occupation categories
- There are several possible ML models to achieve it
- The most straightforward way is to use One-hot Vectors as features and apply a multinomial regression model
- While features in One-hot Vectors are orthognal to each other, with word embeddings, we are able to take semantic similarities between words into account
    - If word *programming* is assigned a “higher weight” in occupation classification into *software engineer* (in e.g. backpropagation in Neural Network), job descriptions with words such as *computer* or *data* that share a similar semantic space with *programming* would also be more likely to be classified as *software engineer* or similar occupations than other less relevant words

Spatial Co-occurence of Occupations

- Scholars are increasingly talking about the new geography of jobs (e.g. Moretti 2013)
  - US labor markets are increasingly polarized across space by the type of occupations
  - With the same level of human capital, service workers have significantly higher earnings in MSAs of innovation hubs
- Scholars have also long noticed the decline of the middle class in the US
- Do professional and managerial occupations increasingly co-appear in the same labor market (e.g. MSA), consequently squeezing out the middle class?

Spatial Co-occurence of Occupations

- Do professional and managerial occupations increasingly co-appear in the same labor market (e.g. MSA), consequently squeezing out the middle class?
- Cross-sectional association seems not working
- We turn to SVD, borrowing the idea of count-based word embeddings model

Spatial Co-occurence of Occupations

- We construct an Occupation-Labor Market Matrix A

	$LOC1$	$LOC2$	$LOC3$	$LOC4$	$LOC5$	$LOC6$
$OCC1$	0.2	0.3	0.7	0.4	0.1	0.2
$OCC2$	0.1	0.2	0.1	0.1	0.5	0.1
$OCC3$	0.6	0.4	0.1	0.2	0.2	0.2
$OCC4$	0.1	0.1	0.1	0.3	0.2	0.5

Spatial Co-occurence of Occupations

- Apply SVD to the matrix A, and approximate A by $A = S \Sigma U^T \approx S_k \Sigma_k U_k^T$
- The occupations are represented by the row vectors of the $m \times k$ matrix $S_k \Sigma_k$
- The labor markets are represented by the column vectors of the $k \times n$ matrix $\Sigma_k U_k^T$
- The occupation vector captures the labor market "context"; occupations that share similar labor market representations/co-appear would share a closer vector space

Spatial Co-occurence of Occupations

- We use v_i to denote the vector for occupation i
- $c_a = \bar{v}_i | i \in$ group a
- $c_b = \bar{v}_i | i \in$ group b
- $S_{a,b} = Cos(\theta) = \frac{<c_a, c_b>}{\|c_a\| \cdot \|c_b\|}$

Spatial Co-occurence of Occupations

- The SVD results suggest that professional and managerial occupations and (wealth) services occupations are increasingly likely to co-occur in the same labor market over time, squeezing out middle class occupations
- Potential polarization of jobs by geography
- There are possible heterogeneities by space, too
- Vector representations of labor markets (e.g. a crystallized index of occupational composition of labor markets) may provide further insight

Occupations as shared understandings

- Google N-gram, the product of a massive project in text digitization across thousands of the world's libraries, distills text from 6 percent of all books ever published (Lin et al. 2012; Michel et al. 2011)
- A representative of the shared understanding of social facts (Kozlowski et al. 2019)
- Historical American English, 5-Grams, 1910-2000 (can be even earlier to 1850)
  - *word2vec*'s skipgram framework trained by each decade
  - transform each occupation (e.g., 1950 census scheme) to a 300-dimension vector
  - $status_i = \frac{1}{n} \sum_{v_o \in O} \|v_o - v_1\| - \|v_o - v_2\|$
  - $v_1 - v_2$ pair including honorable-dishonorable, esteemed-lowly, reputable-disreputable and others (Kozlowski et al. 2019)

Occupations as shared understandings

- Google N-gram, the product of a massive project in text digitization across thousands of the world's libraries, distills text from 6 percent of all books ever published (Lin et al. 2012; Michel et al. 2011)
- A representative of the shared understanding of social facts (Kozlowski et al. 2019)
- Historical American English, 5-Grams, 1910-2000 (can be even earlier to 1850)
  - *word2vec*'s skipgram framework trained by each decade
  - transform each occupation (e.g., 1950 census scheme) to a 300-dimension vector
  - $status_i = \frac{1}{n}\sum_{v_o \in O} \|v_o - v_1\| - \|v_o - v_2\|$
  - $v_1 - v_2$ pair including honorable-dishonorable, esteemed-lowly, reputable-disreputable and others (Kozlowski et al. 2019)
  - Gender and ethnic stereotypes are measured in a similar approach with pairs such as she-he, mother-father, women-men for gender stereotypes, and typical Asian, Hispanic, and White names for ethnic stereotypes (Garg et al. 2018)

Occupations as shared understandings

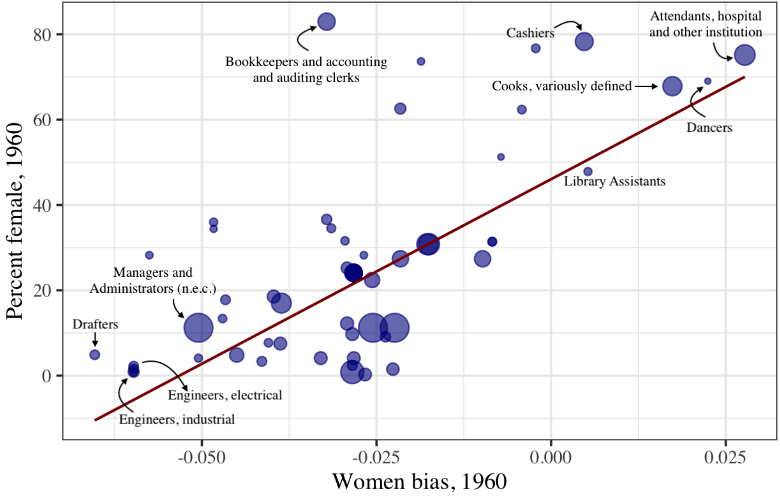


Figure 11: Female Stereotypes and Actual Female Proportion of Occupations, 1950-1960

Occupations as shared understandings

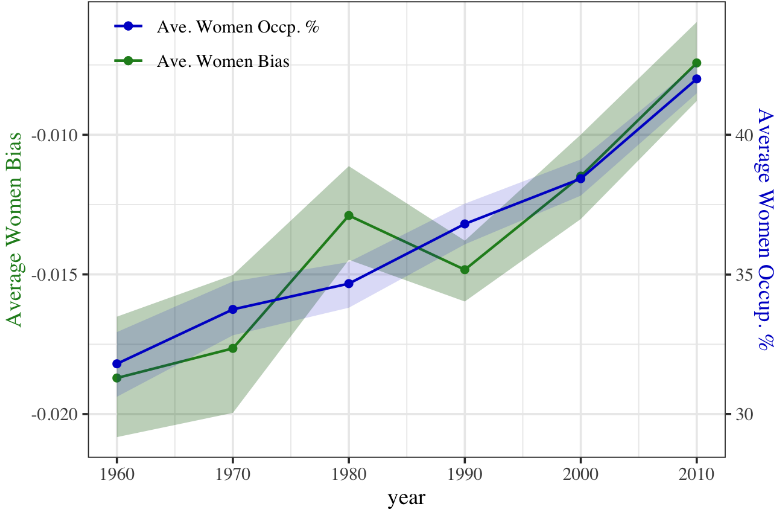
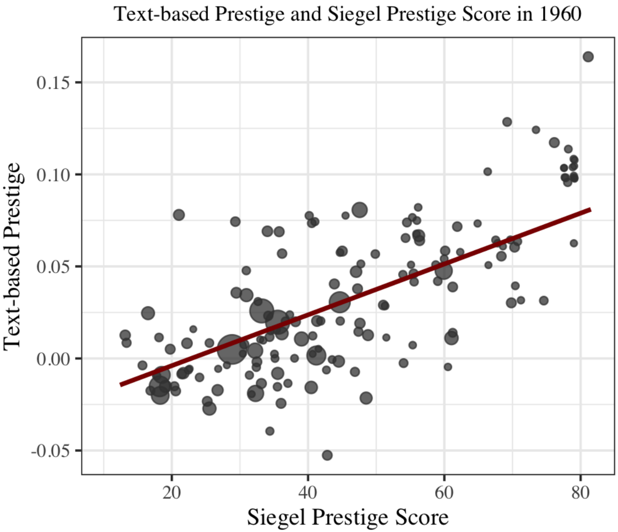


Figure 12: Female Stereotypes and Actual Female Proportion of Occupations Over Time

Occupations as shared understandings



Occupations as shared understandings

- Female stereotypes and status of occupations
- Asian stereotypes and status of occupations

End

- Thanks for listening!
- If you have any questions or would like to chat with me, please feel free to email me at `wj2068@nyu.edu`