A Learner's Experience of Text-as-Data in Labor Economics

Xuanli Zhu
Keio University

June 9, 2025

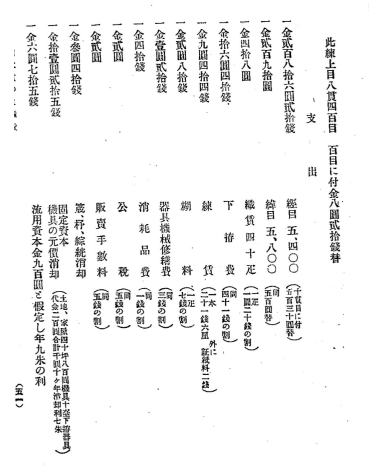
AYEW Roundtable Discussion on Causal ML & Text-as-Data (Extended Version!)

Outline

Self-introduction

- Research interest: skill, task, inequality, technological change, job search/match
- Currently working on empirical labor studies using job ads data and historical data

A Chinese job ads in 2018



A Japanese accounting in 1910

How I Started

- Began to do empirical research during my PhD
  - New question? New theory? New data?
- In China and Japan, the labor data is ... not that good
- I bumped into a seminar talk in which job ads data is used
  - "Oh I know python I can also do web scraping"
- I went through some materials to learn the most basic ML and NLP tools
  - Including Econ review papers, CS lecture notes (see my github repo: Guide2EconRA), blogs, Stack Exchange, `sklearn` documents, ...
  - Somehow often akin to econ models: optimization (cost function), tradeoff (regularization), discrete choice (softmax), MLE (cross-entropy), ...
- I found something interesting in the data and wrote two working papers (2022-2023)

How about Today?

- It's less difficult to catch up (e.g. RNN, transformer, ...) than I thought!
  - The basic framework is often no more difficult than graduate economics models ("lego")
  - The advance/difficulty of the field is very "engineering," but that's largely not our job
- Learning is not much different
  - Except now you can ask a transformer decoder "what is a transformer decoder"
- The shocking facts to me after catching up: "That's It?"
  - GPT-3 is "almost" no different from BERT
  - BERT is just a bunch of simple matrix calculations w/o any magics at all
- Do we really need to know these things?
  - Not that much as recent LLMs often provide the go-to embeddings or classifiers
  - Yes, if you want to justify your choices well or build story based on the mechanisms
  - Often, the simple approach just performs very well and more interpretable

Better Representation ("Embedding") of Texts

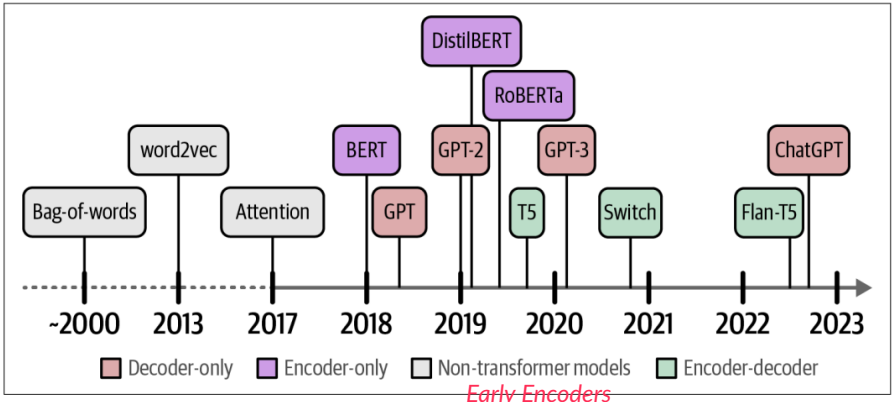


Figure 1-1. A peek into the history of Language AI.

~~Early Encoders~~

(Source: Hands-On Large Language Models)

- Consistent line: better&efficiently represent the meaning of input texts ("encoder") and, via which, better&efficiently simulate text generation ("decoder")

It's All About "Dimension Reduction" 1

One-hot vectors
30k-1m dimensions
discrete

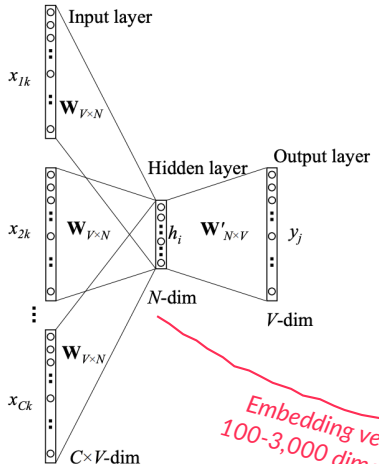
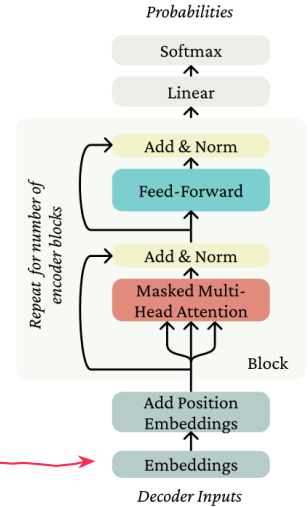


Figure 2: Continuous bag-of-word model

Embedding vectors
100-3,000 dimensions
continuous

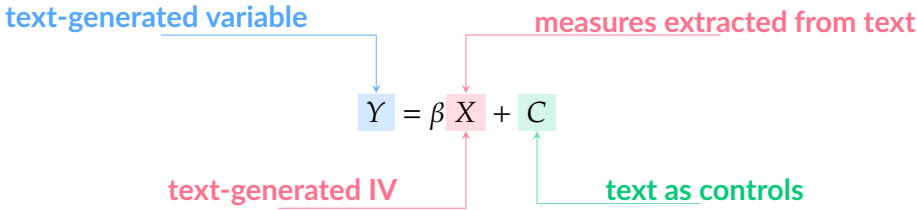


Transformer Decoder

It's All About "Dimension Reduction" 2

- What's the "dimension" of an economics study?
  - What do you remember about the last paper/presentation read/listened?
- Economics: low dimensional (quantitative) "story telling"
  - Care about clean causality but not prediction precisions
  - E.g. a deep learning algorithm that can perfectly predict wage (though very unlikely to exist) does not lead to an economics paper
- Embeddings are still too-high-dimensional; Need to reduce it to tell the specific economics story
  - Use domain knowledge and NLP tools
  - That's why often even simple approaches can work

Use Text-as-Data to Tell Low-dimensional Economics Story



- Y and X can be certain indicator, similarity, or other measures of known interests
- Unexpected stylized facts and X can be explored by examining the embeddings
- C can be any flexible functions with a large set of textual covariates
- Sometimes it can be simply a descriptive story of Y or X

Outline

Natural Language Processing (NLP)

- Natural language is a mapping: [high-dimensional, continuous spatio-temporal reality ⟷ a lower-dimensional, discrete symbolic system]
  - A word: a signifier that maps to a signified (idea, concept, entity, instance)
  - A sequence of words: a set of signifiers that maps to a context
  - Enable complex communication and human intelligence!

- NLP is to design algorithms to allow computers to "understand" natural language in order to perform tasks
  - I.e. how to represent word/words in a form that computer can efficiently process
  - The problem is that word meaning is endlessly complex
  - A key idea: distributional hypothesis: meaning of a word can be derived from the distribution of its contexts

One-hot Vector

- A corpus $C = \{D_1, D_2, \ldots, D_M\}$ is a collection of M documents (articles, sentences, ...)
- A document $D_m = \{w_1, w_2, \ldots, w_n\}$ is a sequence of tokens (words, subwords, ...)
- All unique tokens in the corpus form a vocabulary set V with size $|V|$
- Each word $w \in V$ can then be represented as an $\mathbb{R}^{|V| \times 1}$ one-hot vector:

$$w^{aardvark} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, w^{a} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, w^{aer} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \cdots w^{zzz} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

- Problems:
  - very high-dimensional and sparse vectors ($|V|$ ranges from 0.1 to 10 million)
  - completely independent and no semantic correlations ($[w^{aer}]^T w^{qje} = 0$)

Bag-of-words and Occurrence Matrix

- A document D_m can thus be written as a matrix $\mathbf{X}_m = \{w_1, \ldots, w_{n_m}\} \in \mathbb{R}^{|V| \times n_m}$
  - Hereafter, we abuse the notation to denote w_i as both a token in a sequence and its one-hot vector (in practice often use x_i for inputs)

- Bag-of-words: represent D_m as a vector $\mathbf{x}_m \in \mathbb{R}^{|V| \times 1}$ by summing up $\mathbf{X}_m$ across rows
  - each entry is the count of a word in V in the document
  - again high-dimensional and sparse

- Stack all documents to form a word-document matrix: $\mathbf{X} \in \mathbb{R}^{|V| \times M}$
  - contains co-occurrence info of words (similar words occur in similar contexts)
  - one can normalize and weight the entries using tf–idf
  - $\mathbf{X}^T$ can be used for document-level classification or regression (each word as a feature)

Co-occurrence Matrix and LSA

- Alternatively, store co-occurrences of words in an affinity matrix: $\mathbf{X} \in \mathbb{R}^{|V| \times |V|}$
  - for each word, count the number of times another word appears within a window of certain size across all documents
  - loop across all word pairs to form the affinity matrix
  - window size affects what info to encode (syntactic, semantic, topic)

- Latent semantic analysis (LSA): perform SVD on either $\mathbf{X}$ ($\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}$), and take the submatrix of $\mathbf{U}_{1:|V|,1:k}$ to be the word embedding matrix
  - the cut-off index k is based on the desired percentage variance captured: $\frac{\sum_{i=1}^{k} \sigma_i}{\sum_{i=1}^{|V|} \sigma_i}$
  - simply a PCA without demeaning (to keep sparse)

- Problems:
  - computational cost of SVD is $O\left(mn^2\right)$ for a $m \times n$ matrix
  - optimizing for global reconstruction of the co-occurrence matrix

Word2vec

- Unlike SVD methods, word2vec is iteration-based method and learns co-occurrence of words via training
  - eventually encoding the probability of a word given its context
  - same underlying assumption (linguistics, distributional similarity) as LSA but different factorization and optimization

- Idea: design a statistical model whose parameters are latent word embedding vectors; then learn the parameters by matching its predictions to the data
  - a bit like an econometric model!
  - but not explicitly modeling DGP so not a generative probabilistic model (like LDA)

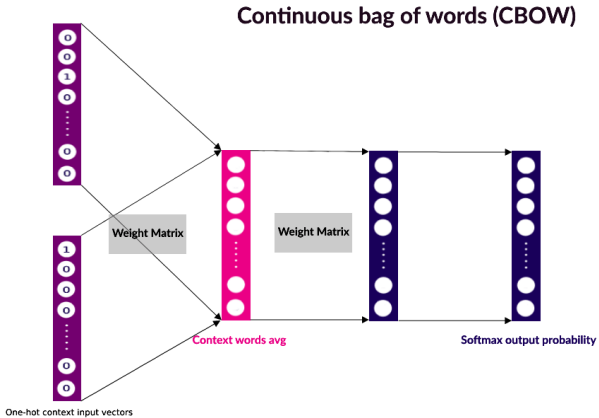
- The architecture is a shallow, one-hidden-layer neural network

- Two algorithms (ways of modeling):
  - continuous bag-of-words (CBOW): predict a center word from context words
  - skip-gram: predicts context words from a center word

Word2vec: CBOW

- For each sentence in the corpus, we can generate a context for each token: $\mathbf{c}_i = \{w_{i-m}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+m}\}$, where w_i is center word and m is window size
- The aim is to find two mappings $\mathbf{U} \in \mathbb{R}^{H \times |V|}$ ("encoder") and $\mathbf{W} \in \mathbb{R}^{|V| \times H}$ ("~~decoder~~")
- $\mathbf{U}$ maps the one-hot vectors of a context into an embedding space of H dimensions: $\{u_{i-m} = \mathbf{U}w_{i-m}, \ldots, u_{i-1} = \mathbf{U}w_{i-1}, u_{i+1} = \mathbf{U}w_{i+1}, \ldots, u_{i+m} = \mathbf{U}w_{i+m} \in \mathbb{R}^H\}$
- $\mathbf{W}$ then maps a context vector into a score vector in dimension $|V|$: $z = \mathbf{W}u_i^o \in \mathbb{R}^{|V|}$, where $u_i^o = \left(\frac{u_{i-m}+\ldots+u_{i-1}+u_{i+1}+\ldots u_{i+m}}{2m}\right) \in \mathbb{R}^H$
- Next, pass through z into a softmax operator to obtain the predicted conditional probability: $\hat{y} = \mathrm{softmax}(z) \in \mathbb{R}^{|V|}$, where $\hat{y}_j \equiv \hat{P}(w_j|\mathbf{c}_i) = \frac{\exp(z_j)}{\sum_{j'=1}^{|V|} \exp(z_{j'})}$
- Lastly, minimize a cross-entropy loss function: $-\sum_{j=1}^{|V|} y_j \log\left(\hat{y}_j\right) = -\log\left(\hat{y}_i\right)$

CBOW



(source: Understanding Bag of Words Models)

CBOW and Skip-gram (some good animations here)

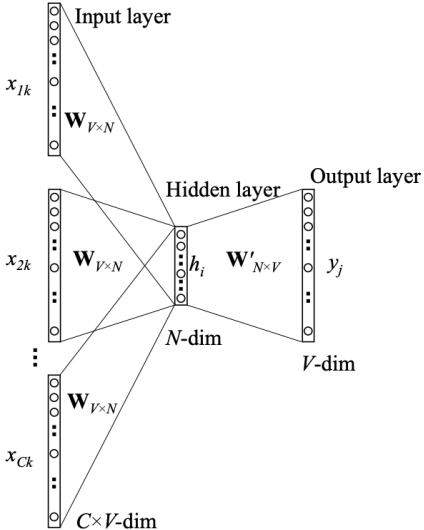


Figure 2: Continuous bag-of-word model

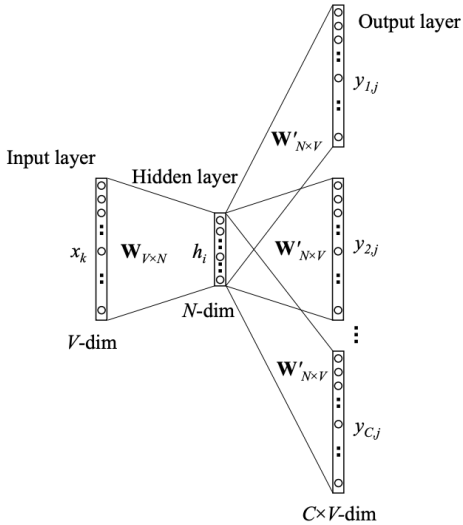


Figure 3: The skip-gram model.

Embedding Vectors

- Embedding vectors are vector representations of tokens with fine-grained semantics and word analogies
  - The local context prediction task forces the embeddings to encode info useful for predicting local linguistic environments
  - Note that word analogy (e.g. $a : b :: c :?$) is different from word similarity

- Linear structures emerge(!) such that the geometric structure of the embedding space allows for vector arithmetic: $u^{king} - u^{man} + u^{woman} = u^{queen}$

- One way to think:
  - $u^{king} - u^{man}$ encapsulates the "royalty added to maleness" features (as expressed by differential context probabilities)
  - $u^{man} - u^{woman}$ encapsulates the "gender difference" features
  - But unfortunately, no ensure to have any single dimension interpretable
  - see more explanations and critiques

Embedding Vectors in Reduced Dimension: Word Analogy

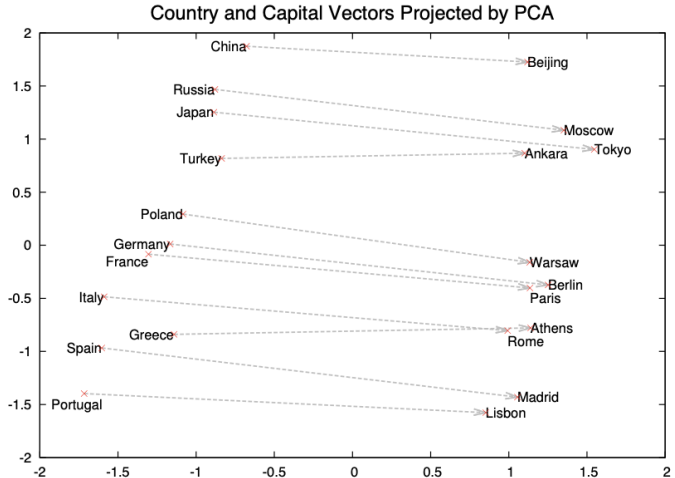


Figure 2: Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities. The figure illustrates ability of the model to automatically organize concepts and learn implicitly the relationships between them, as during the training we did not provide any supervised information about what a capital city means.

Embedding Vectors in Reduced Dimension: Document Similarity

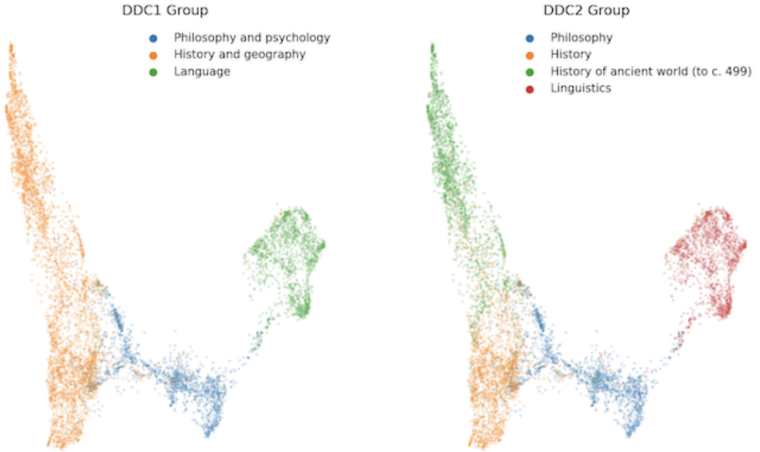


Figure 3. UMAP embedding of selected EThOS data coloured by assigned DDC

(Source: Clustering and Visualising Documents using Word Embeddings)

Softmax Classification

- Most NLP (extrinsic) tasks can be formulated as classification tasks
  - classify topics or sentiment; named-entity recognition (NER); ...
- Softmax classification: probability of a word with embedding vector x being in class j:

  $$p\left(y_j = 1 \mid x\right) = \frac{\exp\left(W'_j \cdot x\right)}{\sum_{c=1}^{C} \exp(W'_c \cdot x)}$$

  - $W' \in \mathbb{R}^{C \times H}$ is the weight matrix for the classification task
  - Use training data to train W'; Retraining $\mathbf{U}$ is risky for small training data
- Use cross-entropy loss function:

  $$-\sum_{j=1}^{C} y_j \log\left(p\left(y_j = 1 \mid x\right)\right) = -\sum_{j=1}^{C} y_j \log\left(\frac{\exp\left(W'_j \cdot x\right)}{\sum_{c=1}^{C} \exp\left(W'_c \cdot x\right)}\right) = -\log\left(\frac{\exp\left(W'_k \cdot x\right)}{\cdots}\right)$$

  - k is the index of the correct class in training data
- Global loss with regularization: $-\sum_{i=1}^{|V|} \log\left(\frac{\exp\left(W'_{k(i)} \cdot x^{(i)}\right)}{\sum_{c=1}^{C} \exp\left(W'_c \cdot x^{(i)}\right)}\right) + \lambda \sum_{k=1}^{C \cdot H + |V| \cdot H} \theta_k^2$

Neural Networks

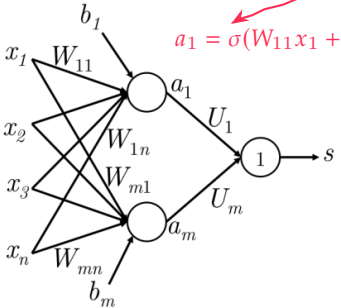


Figure 4: This image captures how a simple feed-forward network might compute its output.

$a_1 = \sigma(W_{11}x_1 + b_1)$

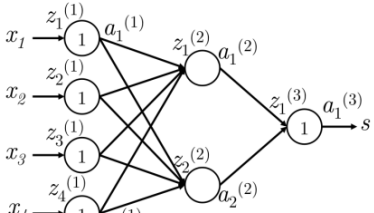


Figure 5: This is a 4-2-1 neural network where neuron j on layer k receives input $z_j^{(k)}$ and produces activation output $a_j^{(k)}$.

activation function

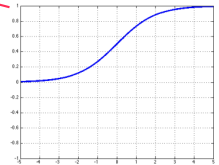


Figure 9: The response of a sigmoid nonlinearity

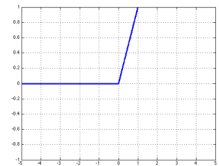


Figure 13: The response of a ReLU nonlinearity

A neuron is a generic computational unit;
$z = Wx + b; a = \sigma(z); s = U^T a$

(A class of non-linear models that have performed particularly well in deep learning applications like NLP)

How Neural Networks Work

Let's see some examples!

Why Neural Networks Work

Jesus Fernandez-Villaverde (University of Pennsylvania) Machine Learning for Macrofinance

MFS Macro Finance Society
チャンネル登録者数 1880人

チャンネル登録

93 共有 保存

Language Models

- How machines do translation or autocomplete?
- A language model is a model that assigns a probability to a sequence of tokens; $P(w_1, w_2, \cdots, w_m)$
  - uni-gram model: $P(w_1, w_2, \cdots, w_m) = \prod_{i=1}^{m} P(w_i)$ (independent word occurrences)
  - bi-gram model: $P(w_1, w_2, \cdots, w_m) = \prod_{i=2}^{m} P(w_i \mid w_{i-1})$ (one-step Markov chain)
  - n-gram model: $P(w_1, w_2, \cdots, w_m) = \prod_{i=n}^{i=m} P(w_i \mid w_{i-n}, \ldots, w_{i-1})$
- Simplest way to compute the probabilities: frequency
  - $p(w_2 \mid w_1) = \frac{\text{count}(w_1, w_2)}{\text{count}(w_1)}$ (bi-gram); $p(w_3 \mid w_1, w_2) = \frac{\text{count}(w_1, w_2, w_3)}{\text{count}(w_1, w_2)}$ (tri-gram)
  - two main issues: sparsity & storage
- We can also incorporate NNs into a word2vec-like window-based architecture

Window-based Neural Language Model

$$\hat{y} = \text{softmax}\left(W^{(2)} \tanh\left(W^{(1)}x + b^{(1)}\right) + W^{(3)}x + b^{(3)}\right)$$

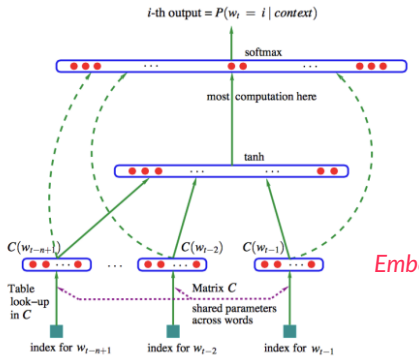


Figure 1: The first deep neural network architecture model for NLP presented by Bengio et al.

Embeddings

$$\hat{y} = \text{softmax}\left(\boldsymbol{U} f\left(\boldsymbol{W}\boldsymbol{e} + \boldsymbol{b}_1\right) + \boldsymbol{b}_2\right)$$

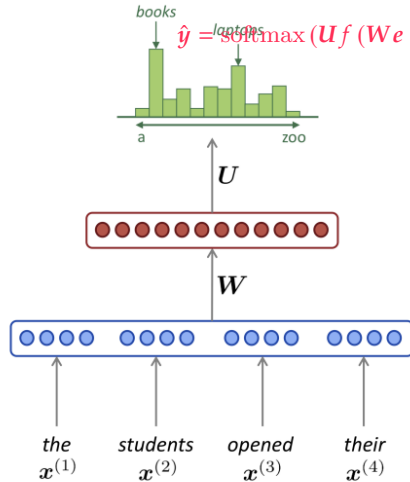


Figure 2: A simplified representation of Figure 1.

(But how wide would the window (context) be enough?)

Recurrent Neural Networks (RNN)

$$h_t = \sigma\left(W^{(hh)}h_{t-1} + W^{(hx)}x_{[t]}\right); \hat{y}_t = \text{softmax}\left(W^{(S)}h_t\right)$$

(Note that same $W^{(hh)}$ and $W^{(hx)}$ used across all layers)

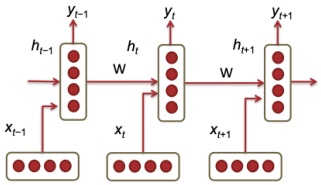


Figure 3: A Recurrent Neural Network (RNN). Three time-steps are shown.

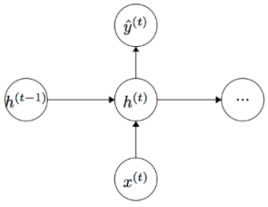


Figure 4: The inputs and outputs to a neuron of a RNN

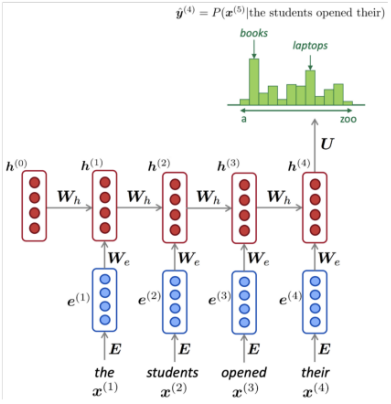


Figure 5: An RNN Language Model

(Inputs can be any length without worrying about the curse of dimensionality!)

Drawbacks of RNN

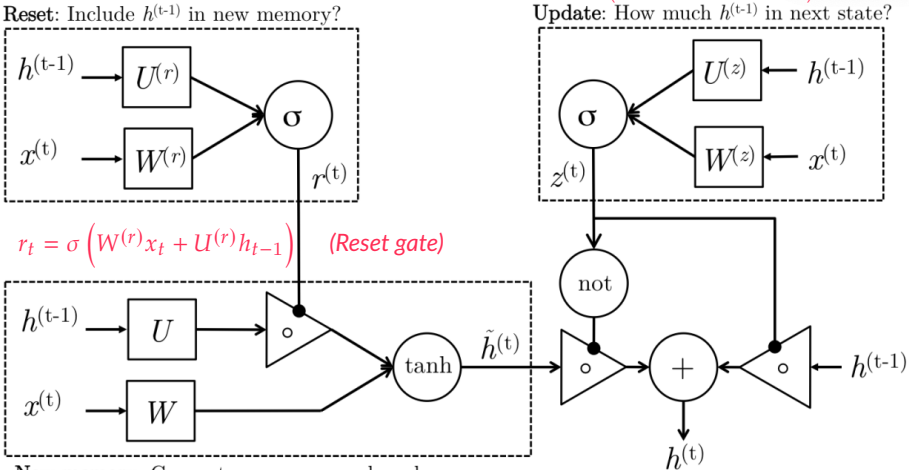
1. Computation is slow: because it is sequential, it cannot be parallelized
   - Modern GPUs are excellent at simple operations in parallel (e.g. $\mathbf{AB}$)
   - We cannot calculate $\mathbf{h}_2 = \sigma\left(W\mathbf{h}_1 + U\mathbf{x}_2\right)$ before calculating $\mathbf{h}_1$

2. In practice, it is difficult to "recall" information from many steps back due to problems such as vanishing gradients
   - RNNs propagate weight matrices from one timestep to the next
   - During the back-propagation phase, the contribution of gradient values gradually vanishes as they propagate to earlier timesteps
   - Use ReLU instead of the sigmoid function can help

both issues had to do with the the depenence on the sequence index ("time")!

Gated Recurrent Units

$$z_t = \sigma\left(W^{(z)}x_t + U^{(z)}h_{t-1}\right) \quad \text{(Update gate)}$$



$$r_t = \sigma\left(W^{(r)}x_t + U^{(r)}h_{t-1}\right) \quad \text{(Reset gate)}$$

$$h_t = (1 - z_t) \circ \tilde{h}_t + z_t \circ h_{t-1} \quad \text{(Hidden state)}$$

$$\tilde{h}_t = \tanh\left(r_t \circ Uh_{t-1} + Wx_t\right) \quad \text{(New memory)}$$

Figure 12: The detailed internals of a GRU

Transformer

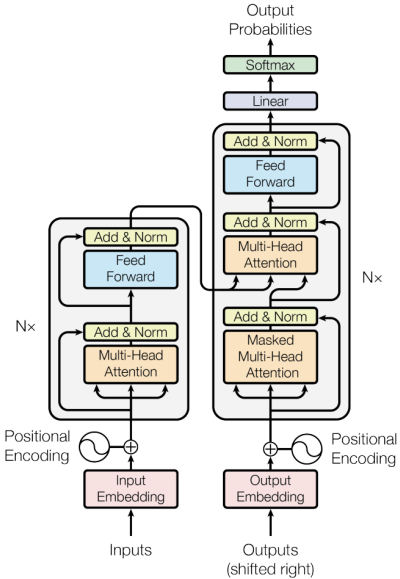
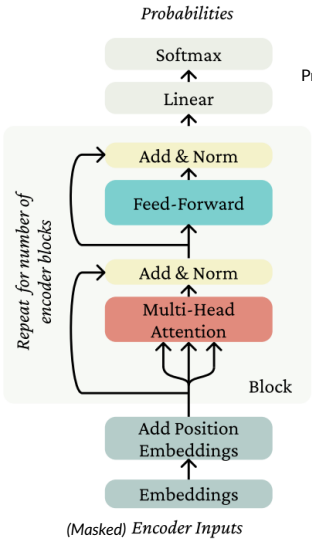


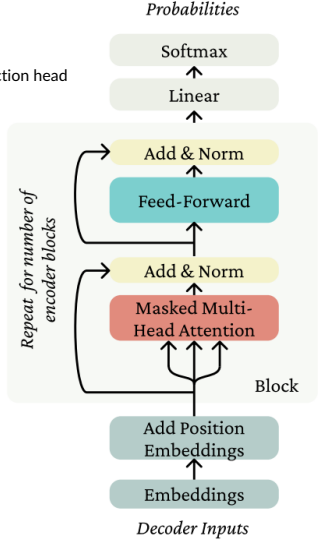
Figure 1: The Transformer - model architecture.

[T]ransformer: ↓B*E*R[T] ↓GP[T]

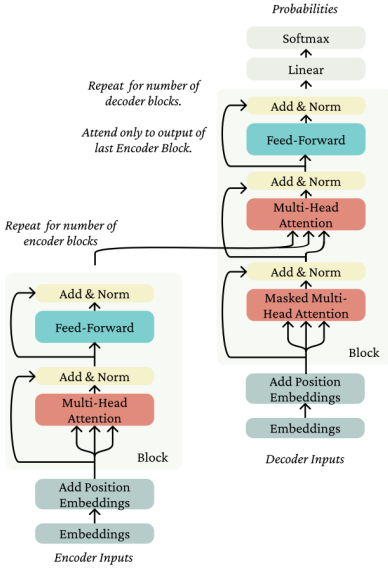


Transformer Encoder

Prediction head

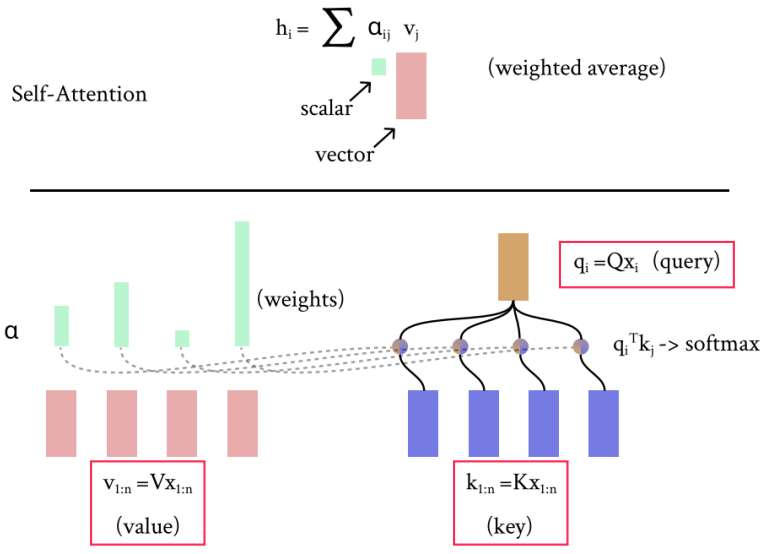


Transformer Decoder



Transformer Encoder-Decoder

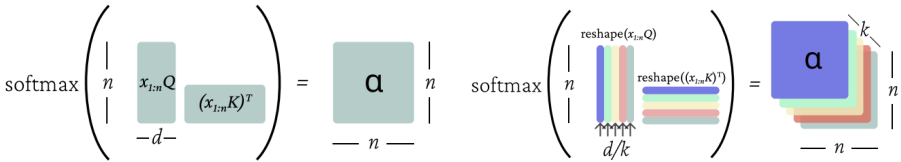
Self-Attention Mechanism: Key-Query-Value



"Attention Is All You Need!"

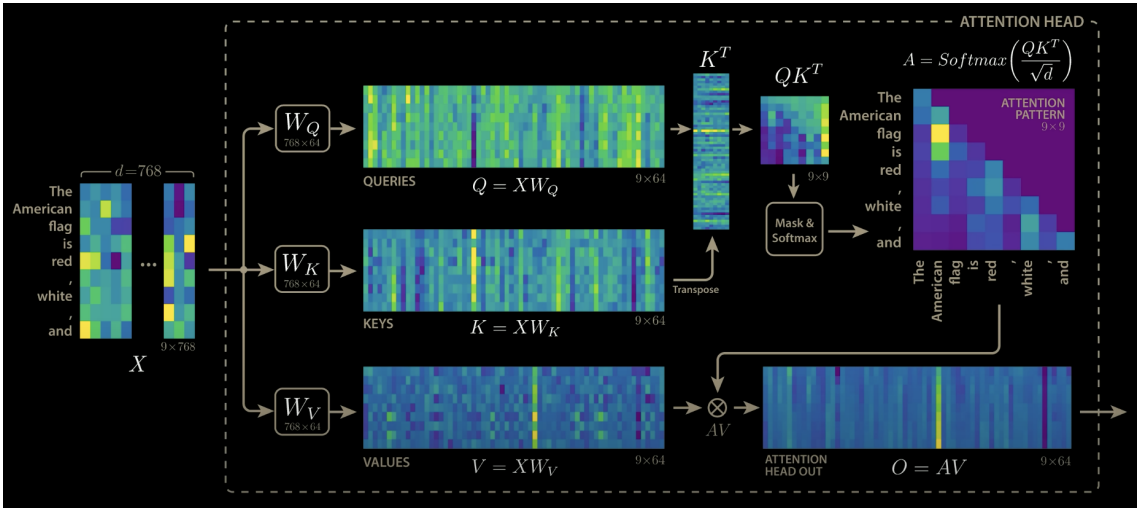
Multi-head Self-Attention

With a single head of self-attention, one similarity measure $\mathbf{q}_i^\top \mathbf{k}_j$ needs to balance different syntactic, semantic, and context dimensions



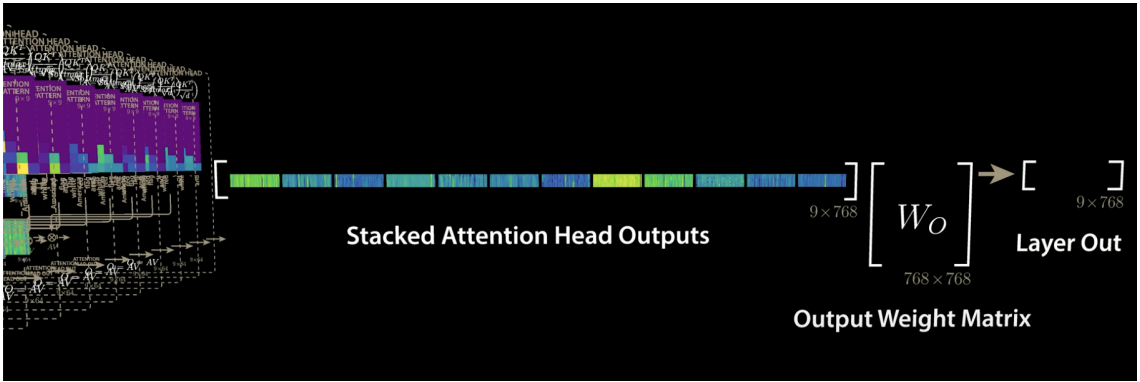
Split a single self-attention head into multiple heads, each with different key, query, and value matrices, and then combines the outputs

Another Great Visualization



(Source: MLA/DeepSeek Attention)

Another Great Visualization



(Source: MLA/DeepSeek Attention)

Other "Lego" Parts

- Position representations:
  - In self-attention operation, no built-in notion of order (unlike RNN w/ native input order)
  - Simple solution: $\tilde{x}_i = P_i + \mathbf{x}_i$, where $P \in \mathbb{R}^{N \times d}$ is a position embedding matrix
- Feed-forward NNs:
  - If we simply stack self-attention layers, there is no elementwise nonlinearities
  - After a layer of self-attention, apply FFN independently to each word representation: $h_{\text{FF}} = W_2 \operatorname{ReLU}(W_1 h_{\text{self-attention}} + b_1) + b_2$ (often, $W_1 \in \mathbb{R}^{5d \times d}$, $W_2 \in \mathbb{R}^{d \times 5d}$)
- Layer norm:
  - motivation: reduce uninformative variation in the activations at a layer to pass over
  - computes statistics: $\hat{\mu}_i = \frac{1}{d} \sum_{j=1}^{d} \mathbf{h}_{ij}$, $\hat{\sigma}_i = \sqrt{\frac{1}{d} \sum_{j=1}^{d} \left(\mathbf{h}_{ij} - \mu_i\right)^2}$ for $i \in \{1, \ldots, n\}$
  - compute the layer norm: $\operatorname{LN}(\mathbf{h}_i) = \frac{\mathbf{h}_i - \hat{\mu}_i}{\hat{\sigma}_i}$
- Add (residual connections):
  - $f_{\text{residual}}(\mathbf{h}_{1:n}) = f(\mathbf{h}_{1:n}) + \mathbf{h}_{1:n}$ (easy to learn from the identity function)
  - In practice: $\mathbf{h}_{\text{pre-norm}} = f(\operatorname{LN}(\mathbf{h})) + \mathbf{h}$ or $\mathbf{h}_{\text{pre-norm}} = f(\operatorname{LN}(\mathbf{h})) + \mathbf{h}$

Encoder vs. Decoder

- Decoder: predict a word given all words so far: $\mathbf{w}_t \sim \text{softmax}\left(f\left(\mathbf{w}_{1:t-1}\right)\right)$
  - No sense to look at the future when predicting it (built-in feature of RNNs)
  - Future masking for representing token i: $\alpha_{ij,\text{ masked}} = \begin{cases} \alpha_{ij} & j \leq i \\ 0 & \text{otherwise} \end{cases}$
- Encoder: learn the embedding of a single sequence $\mathbf{w}_{1:n}$
  - No future masking so it is bi-directional
  - Instead randomly mask 15% tokens (replace 80% of their occurrence with `[Mask]`) and do self-supervised training ("masked language modeling")
  - Also insert a `[CLS]` token before each input sequence (a summary!) and a `[SEP]` token between each two segments
- Both are window-based methods, hence have maximum input sequence length
  - If fewer than the maximum, padding with `[PAD]` tokens

Visualization of BERT Input and Bidirectional Attention

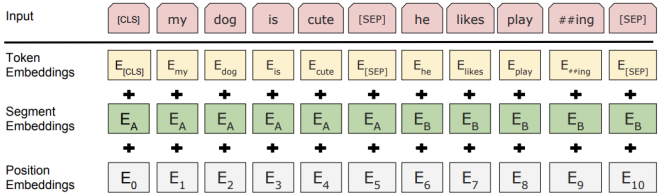


Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

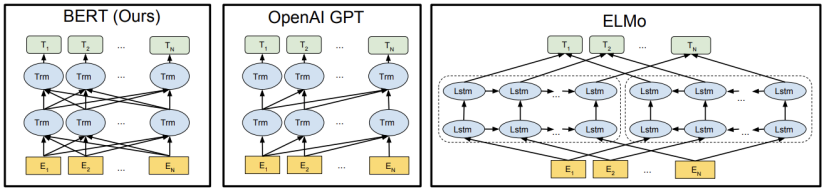


Figure 3: Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTMs to generate features for downstream tasks. Among the three, only BERT representations are jointly conditioned on both left and right context in all layers. In addition to the architecture differences, BERT and OpenAI GPT are fine-tuning approaches, while ELMo is a feature-based approach.

In Practice: Encoder

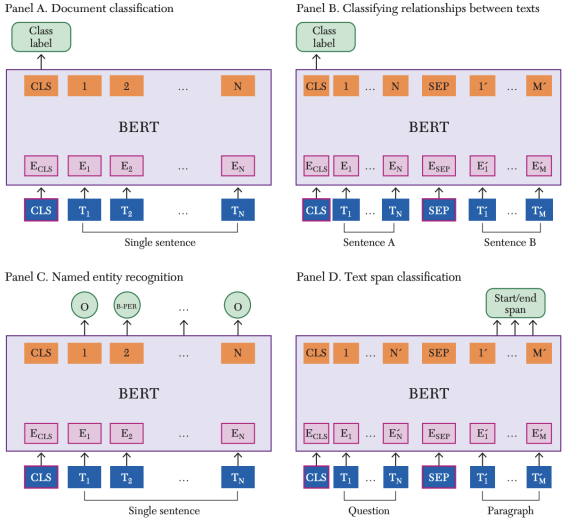


Figure 3. Tasks Performed with a Transformer Encoder Language Model

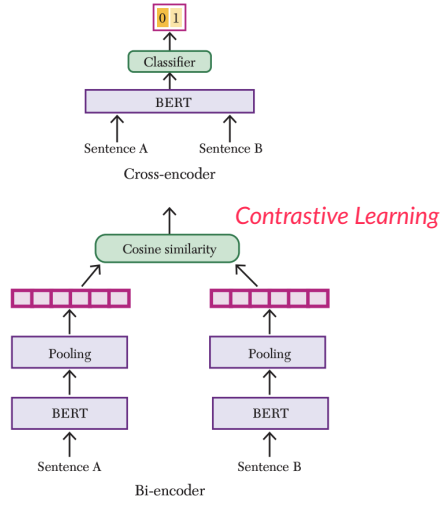


Figure 6. Architectures to Compare Texts

(Source: Dell, M. (2025). Deep Learning for economists. JEL)

In Practice: Classification

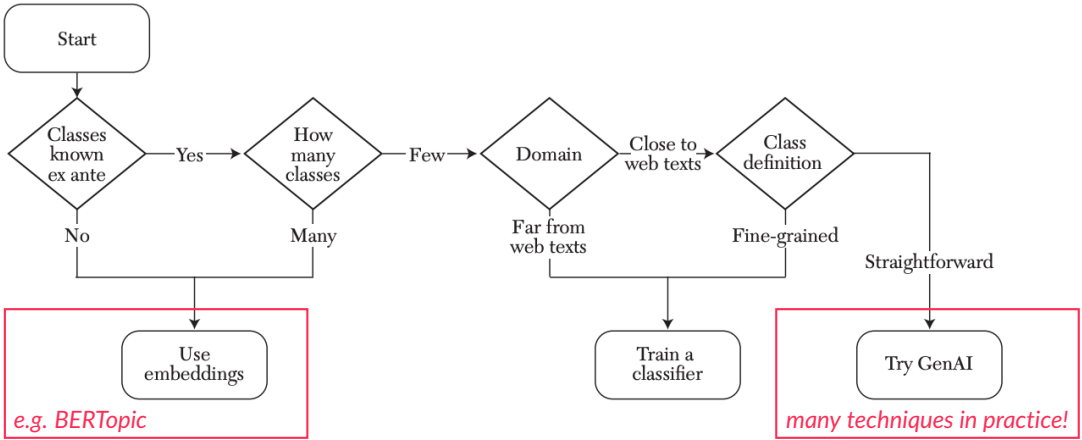
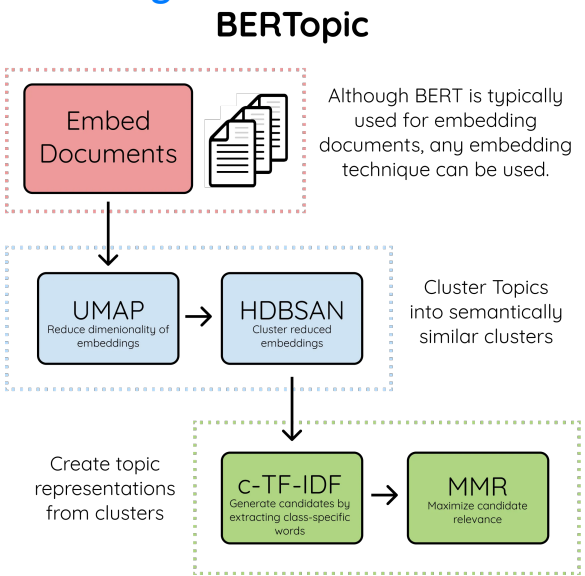


Figure 1. Flowchart for Approaching Classification

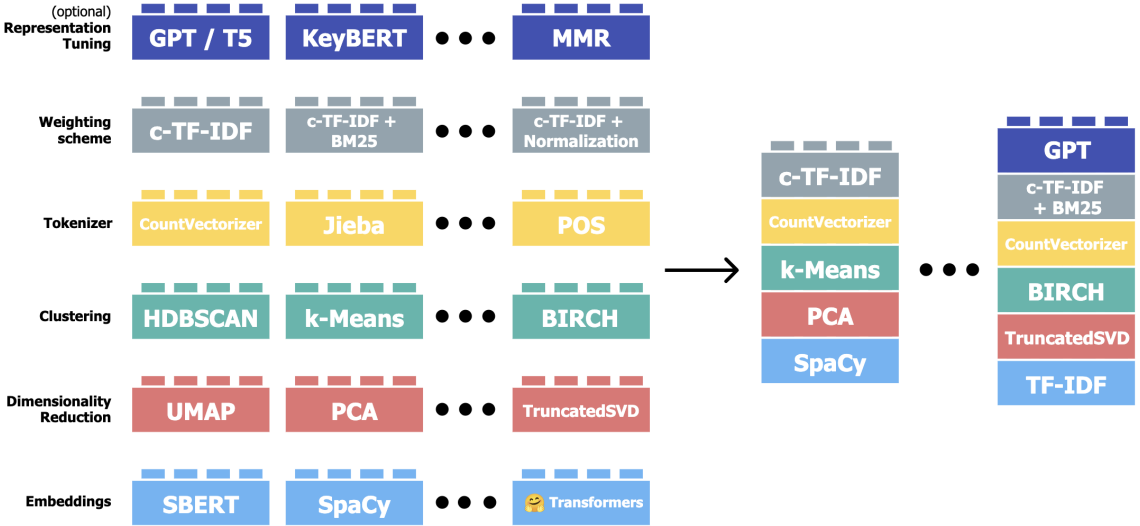
(Source: Dell, M. (2025). Deep Learning for economists. JEL)

In Practice: Topic Modeling



(Source: NLP Tutorial: Topic Modeling in Python with BerTopic)

In Practice: Topic Modeling



(Source: BERTopic: The Algorithm)

Outline

Deming and Kahn (2018): Extract Skill Demand From Job Ads

- Empirical question: do variations of firms' skill demand (conditional on occupation) affect pay?
- Data: US job ads (2010–2015) provided by Burning Glass Technologies
- Method: keyword dictionary based on domain knowledge; job-level indicator

(definitely can do better today but the definition itself can be sometimes tricky)

Table 1
Description of Job Skills

Job Skills	Keywords and Phrases
Cognitive	Problem solving, research, analytical, critical thinking, math, statistics
Social	Communication, teamwork, collaboration, negotiation, presentation
Character	Organized, detail oriented, multitasking, time management, meeting deadlines, energetic
Writing	Writing
Customer service	Customer, sales, client, patient
Project management	Project management
People management	Supervisory, leadership, management (not project), mentoring, staff
Financial	Budgeting, accounting, finance, cost
Computer (general)	Computer, spreadsheets, common software (e.g., Microsoft Excel, PowerPoint)
Software (specific)	Programming language or specialized software (e.g., Java, SQL, Python)

because they are important predictors of productivity and wages … their prominence in the literature

Wage Regression on Average Skill Indicators

- Wage data from labor survey as only 13% job abs has wage posted
- OLS: $\log(\text{Wage})_{om} = \alpha + \overline{\text{Skill}}_{om}\beta' + \text{Controls} + \epsilon_{om}$ (MSA-occupation level)
- Results:

Table 3
Average Wages and Skill Requirements

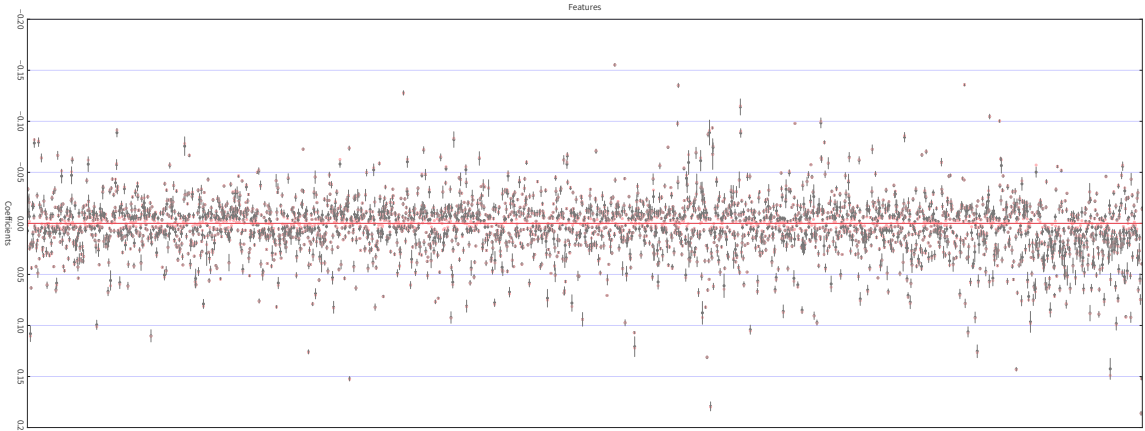
	Dependent Variable: Log(Mean Wages) in MSA-Occupation Cells					
	(1)	(2)	(3)	(4)	(5)	(6)
Cognitive	.113***	−.413***	.245***	.181***	.0792***	.0465***
	(.00908)	(.0166)	(.00784)	(.0139)	(.00873)	(.0122)
Social	.429***	−.0919***	.301***	.236***	.0517***	.0202
	(.0155)	(.0206)	(.0121)	(.0167)	(.00966)	(.0127)
Both required		1.319***		.157***		.0760***
		(.0349)		(.0278)		(.0198)
Years of education	.131***	.129***	.0764***	.0765***	.00865***	.00873***
	(.000770)	(.000763)	(.000844)	(.000844)	(.000995)	(.000995)
Years of experience	.160***	.161***	.0848***	.0849***	.0318***	.0318***
	(.00120)	(.00118)	(.00120)	(.00120)	(.00102)	(.00102)
Base controls			X	X		
Detailed controls					X	X
F-statistic (cognitive and social)	553.1	855.0	1,004	680.4	69.66	51.35
F-statistic (all 10 skills)	1,874	2,054	612.6	560.1	59.93	55.83
MSA-occupation cells	56,611	56,611	56,611	56,611	56,611	56,611
R^2	.702	.710	.846	.846	.940	.941

Zhu (2023): Explore at the Entire Skill/Task Forest

- Empirical question: What are the most important skills/tasks for wages?
- Data: China job ads (2014-2020; self-scrapped) with posted wages
- Method:
  1. Feature selection by Lasso with BIC *(110,000+ $\longrightarrow$ 3100+)*
  2. Feature clustering by K-Means on word2vec embeddings *(3100+ $\longrightarrow$ 8 groups)*
  3. Dimension reduction by PLS *(3100+ $\longrightarrow 8 \times 3 = 24$)*
  4. Wage regression and variance decomposition w/ PLS variables or artificial occupations

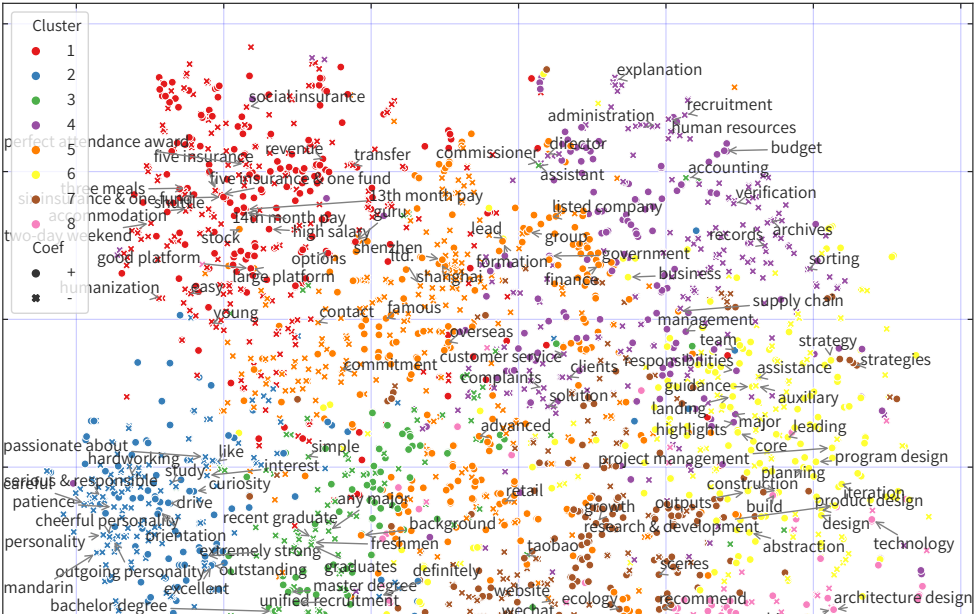
  (1 & 2 is exploratory; No priors at all on what to look!)
- Finding: clusters based on occupational specificity; specific skills/tasks matter most

Lasso Wage Regression

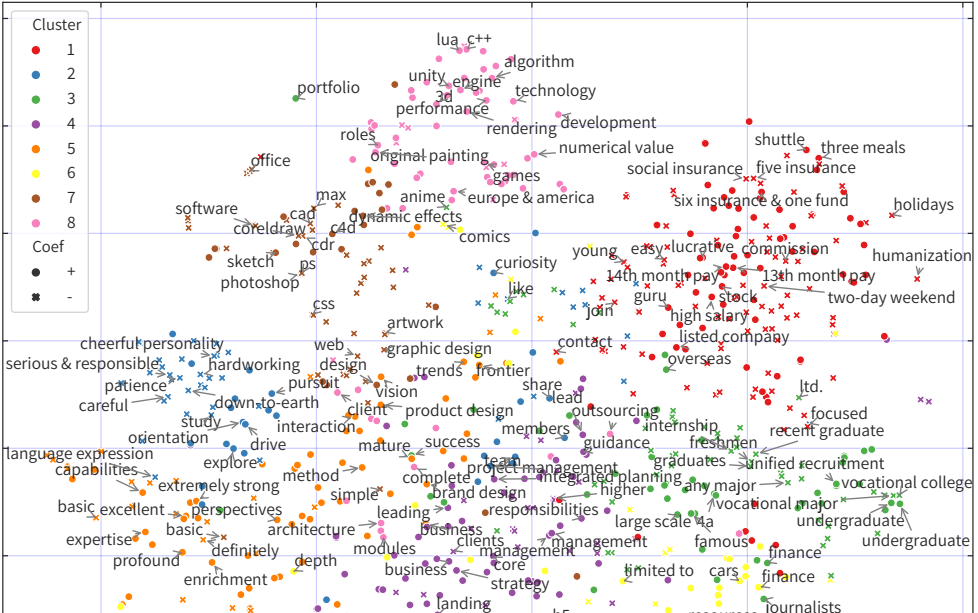


(3100+ non-zero coefficients under BIC criterion; Confidence interval through subsampling)

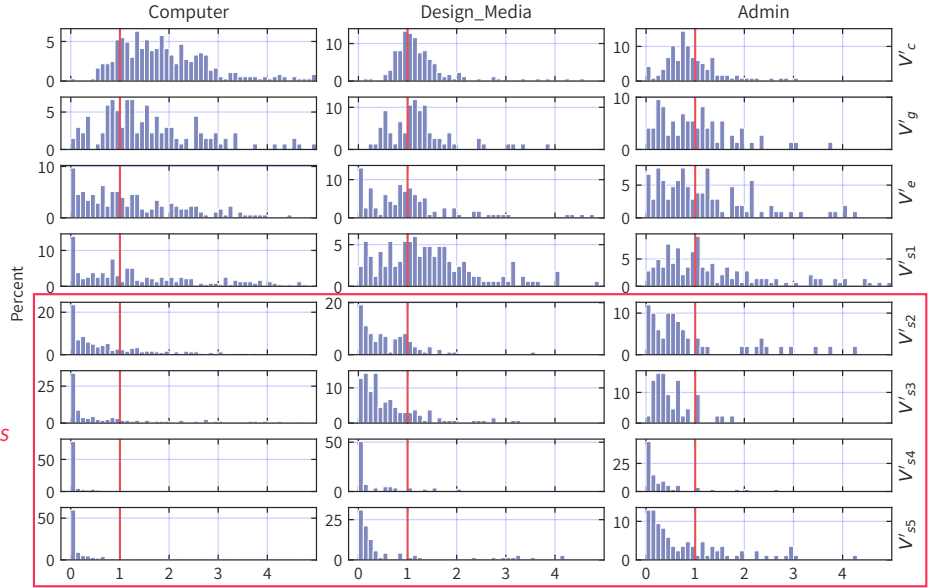
Embedding Clustering Visualization (T-SNE): Pooled Occupations



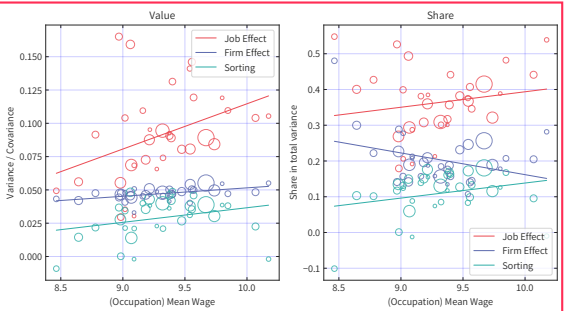
Embedding Clustering Visualization (T-SNE): Computer Occ Only



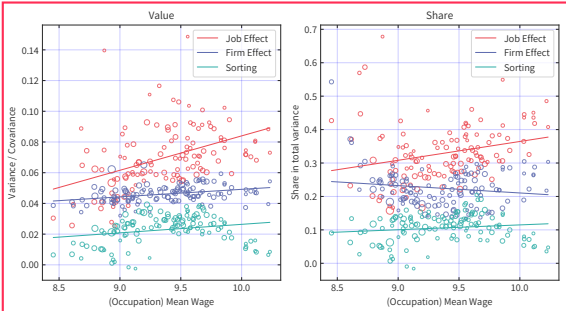
Label Clusters by Characterizing Occupation-specificity ($TF^{o'}/TF^{o}$)



Real vs. "Artificial" Occupations



37 SOC 5-/6-digit Occupations



320 Artificial Occupations (Job-level Embedding Clusters)

Dube et al. (2020): Job Text as Controls

- Empirical question: how does variation in rewards (wage) affect duration of task vacancies (labor supply) in Amazon MTurk?
- Identification problem: omitted variables that affect both (e.g. task difficulty)
- Double Machine Learning estimator (a generalization of FWL, see the textbook of Chernozhukov et al.):

$$\ln(duration) = -\eta \ln(reward) + g_0(Z) + \epsilon, \quad E\left[\epsilon \mid Z, \ln(reward)\right] = 0, \tag{3}$$

$$\ln(reward) = m_0(Z) + \mu, \quad E[\mu \mid Z] = 0. \tag{4}$$

  - $\Rightarrow \ln(\text{duration}) - E[\ln(\text{dur}\hat{\text{a}}\text{tion}) \mid Z] = -\eta \left(\ln(\text{reward}) - E[\ln(\text{rew}\hat{\text{a}}\text{rds}) \mid Z]\right) + \epsilon$
  - Z includes textual covariates (n-grams, topic distributions, Doc2Vec embeddings, ...)
  - Use sample splitting and training/validation sets to select useful features & best algorithms

Autor et al. (2024): Text to Find New Occupations and Link to Tech

- Empirical questions: What new occupations emerged? What technologies generated them?
- To find new occupations: compare some granular census indexes of occupations (CAI) in 1940 vs 1930, 1950 vs 1940, ...
- To attach occupation with the exposures with different technologies:
  - Automation tech: patent data similarity with DOT data
  - Augmenting tech: patent data similarity with CAI data

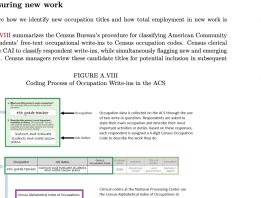
Find New Occupations with Rules, Fuzzy M, and Manual Inspection

D Measuring new work

We detail here how we identify new occupation titles and how total employment in new work is constructed.

Figure A.VIII summarizes the Census Bureau's procedure for classifying American Community Survey respondents' free-text occupational write-ins to Census occupation codes. Census clerical coders use the CAI to classify respondent write-ins, while simultaneously flagging new and emerging write-in titles. Census managers review these candidate titles for potential inclusion in subsequent CAI editions.

FIGURE A.VIII
Coding Process of Occupation Write-Ins in the ACS



1 Procedure for identifying new occupation titles

To extract new work added to the Census Alphabetical Index of Occupations (CAIO) between Census or ACS years $t-1$ and t we use the following steps:

1. Clean titles in both $t-1$ and t by removing capitalization, punctuation, as well as certain common synonyms and decade-specific format changes we identify from inspection of CAIO volumes. This avoids unnecessarily flagging titles as potentially new ("candidate-new") if they are old titles that have been reformatted or reworded in minor or predictable ways.
   (a) Examples of format and wording changes that we discard are titles like "Accounting Work, Accountant" and "Ad Writer" being added in t when "Accountant" and "Advertising Writer" already exist in $t-1$.
   (b) We also unify variations of titles which contain the same terms either in full or abbreviated form, such as "ib" for database, "pl" for physical therapy, "pv" for photovoltaics, and "qc" for quality control.
   (c) Prior to matching, we reduce -man, -person, -work, -er, -or, -ing, -ist etc. titles to the same word base, e.g. "Salesperson", "Salesman" and "Sales work" are changed to "sales"; "Advisor", "Advisor" and "Advising" are degenerated to "advis", and "Motorist" is degenerated to "motor".
   (d) We clean plural forms, including those ending in "-s" or "-es", and other specific plural forms such as "-ies" when it is a plural of "-y".
   (e) We also discard any gender-specific or gender-neutral versions of existing titles, e.g. we treat the titles "Actor" and "Actress" as one and the same; as we do "Waiter", "Waitress", and "Waitstaff"; and we discard "Chipper Operator" as new because it replaced "Chipper man".
   (f) We discard word order duplicates that are classified to the same Census occupation (e.g. out of "Television Station Manager", and "Manager, Television Station", we retain only one): these occur because at the time of its conception the alphabetical index was used in printed form– multiple word orders were included to save coders time in looking up entries. We retain any title duplicates classified to different industries or occupations, as this may reflect (increasing) emergence of a type of job (an example is the prevalence of IT-related titles across many industries).
   (g) Examples of words we automatically denote as synonyms are "auto" and "automobile"; "equipment operator" and "operator"; "sales", "selling", and "sales representative"; "garbage" and "rubbish"; "aide" and "assistant"; "gage" and "gauge".
2. Exact-match and fuzzy-match of cleaned occupation titles between CAIO$_t$ to CAIO$_{t-1}$. We drop all exact title duplicates between $t-1$ and t, disregarding any spacing differences in titles. For the remainder, we retain the three most similar $t-1$ title matches for each t title. Specifically:
   (a) For the exact match, we simply match the cleaned titles in t to $t-1$, discard exact matches, and retain the set of unmatched CAIO$_t$ titles as "candidate-new" titles.
   (b) Next, we fuzzy match the CAIO$_t$ candidate-new titles to all CAIO$_{t-1}$ cleaned titles. We use a fuzzy-matching Jaro-Winkler algorithm which matches based on letter-swaps, implemented in R as the package *stringdist* (Loo, 2014). This assigns high similarity (i.e. low distance) scores to titles where a low number of single-character transpositions are required to change one word into the other.[29] It also gives higher similarity to strings matching from the beginning up to some specified length: we set the constant scaling factor determining this at the standard value of 0.1. For example, titles which are identical except for a hand-keying error ("Mechanotherapist" and "Mechanotherapist") receive a high similarity score.
3. Adjudicate remaining unmatched $t+1$ titles ("candidate-new" titles) by classifying them as new or not new, using a combination of automated assignment and careful manual revision. The large majority of candidate-new titles are manually revised, with only around 1,034 automatically assigned. We observe 273,960 total titles over 1940–2018, of which we identify 28,315 as new over the whole period.

In adjudicating candidate-new titles, our overarching goal is to identify titles that either capture a type of job that was previously nonexistent or reflect further differentiation or specialization of existing work. These latter cases are much more common than are entirely new categories, and can arise from new or specialized work domains, specialization in educational or professional requirements, or the use of specialized work methods (e.g. by hand or using a machine). On the other hand, candidate-new titles are discarded (i.e. marked as not new) when they reflect a renaming, reformatting, or generalization of previously existing work. This (time-consuming) manual revision requires looking beyond fuzzy-match results to search the entire $t-1$ index for comparable work.

We implement these principles with the following specific rules for classifying a candidate-new title as new or not new. While not exhaustive, these rules capture commonly occurring cases. A t candidate-new title is:

1. *New* when it is a differentiation of a t title, e.g. "Clinical Psychologist" is new in 1950 as a differentiation of "Psychologist", and "Assembler, Electrical Controls" is new in 1990 as a differentiation of "Assembler, n.s". This is by far the most commonly occurring type of new title.
2. *New* when it adds specialized work tools to a $t-1$ title, most commonly 'hand' or 'machine', or specializes operators and set-up operators. E.g. "Bookkeeping Clerk" whose title is new in 1970 because before only "Bookkeeping Clerk" was listed; and "Drill-Press Set-Up Operator" is new when it is added to "Drill-Press Operator".
3. *New* when it adds some additional educational or professional differentiation to a $t-1$ title. E.g. "Licensed Addiction Counselor" is new in 2018 as an addition to "Addiction Counselor"; and "Health Therapist, Less Than Associate Degree" is new in 1990 as an addition to "Health Therapist". This is a type of new title that occurs relatively infrequently.
4. *Now when* it adds "not specified" or "not elsewhere classified" to a $t-1$ title. This reflects more types of this title are emerging which (for the time being) are listed as n.s. / n.e.c. For example, "Mechanic, Instrument, n. s" is added in 1980.
5. *Now when* it bifurcates a $t-1$ title into two separate types, usually marked with "incl", "exc", or "any other". E.g. in 1980 the title "Sitter, exc. Child Care" was new since before only "Sitter" had existed.
6. *Not now* when it simply reorganizes information across various columns of the index for the same title. E.g. "Apprentice Dentist" was discarded as new in 1940 because it already existed in 1930. This is a common reason for discarding candidate-new titles.
7. *Not now* when it is generalization from previously-specified title, e.g. "Ad Taker" is not new in 1980 because it simply subsumes the 1970 titles "Classified-Ad Taker" and "Telephone-Ad Taker"; and "Inspector Agricultural commodities" is not new in 1980 because it subsumes "Inspector Fruit", "Inspector Food", and "Inspector Livestock".
8. *Not new* when it is the same as a $t-1$ title except for filler words; or an (un)abbreviated version. E.g. "Software Applications Developer" is not new in 2018 because the title "Software Developer" already existed before; and "RRT" is not new in 2018 because "Registered respiratory therapist" already existed before.
9. *Not new* when a title is a combination of two existing titles. E.g. "Inker and opaquer" is not new in 1980 because both "Inker" and "Opaquer" already existed in 1970.

Overall, we discard 15.56% of all candidate-new titles we manually review ($N = 30,332$). This manual revision is necessary since many newly added titles reflect synonyms or rewordings of existing titles, which would not reliably be captured by an algorithm. Further examples of candidate-new titles we discard are "Nanny" (as the synonyms "Governess", and "Baby sitter" already existed); "Server, food" (as the synonym "Waiter" already existed), "Glass worker" (as various more specific glass workers already existed, including "Glass bender", "Glass blower", "Glass cleaner", "Glass cutter"); "Flight instructor" (as the synonym "Teacher, flying" already existed); "Ad setter" (as the synonym "Advertising-layout man" already existed); "Pilot, aircraft" (as the synonym "Aviator" already existed); "Supervisor of police" (as the synonym "Police superintendent" already existed); "Key attendant" (as "Key boy" and "Key girl" already existed and the Census only deprecated the title); "Mental health aide" (as "Psychiatric aide" already existed); "Dog groomer" (as the more specific "Dog barber" and "Dog beautician" already existed); "Private eye" (as the synonym "Private detective" already existed); "General-office worker" (as the synonym "Office Employee" already existed); "Bingo usher" (as the synonym "Bingo attendant" already existed); "PT" (as the unabbreviated "Physical therapist" already existed); "O.T." (as the unabbreviated "Occupational therapist" already existed); and "NP" (as the unabbreviated "Nurse practitioner" already existed).

2 Examples of new work

Here, we provide several illustrative examples of the emergence of new work at the occupation level. Measuring the emergence of new work requires a consistent set of occupation over the full 1940–2018 interval, which we have constructed at the cost of some information loss ($N = 132$ occupations), as described in Appendix C. [Note that our main analyses do not use this 80-year consistent series because of the loss of occupational granularity.]

As a first example, consider the occupation *Office machine operators*, where 178 new titles have been added over 1940–2018. In the first decades of our sample, examples of such titles are Recordak operator, Letter opener operator, Check writing machine operator (all added in 1940); Telautograph operator, Microfilm cameraman, Multilith operator, I.B.M. operator (all added in 1950); and Xerox-machine operator, Computer operator, Univac operator, Check-sorter operator (all added in 1960). This new work reflects various innovations in office machinery, including salient examples related to the Recordak system developed by the Eastman Kodak Company in 1928, which photographed documents on 16- or 35-millimeter film[30]; the development of I.B.M. and Xerox office machines; and Universal Automatic Computer (Univac) machines used by, among others, the U.S. Census itself.[31] In later decades, new titles in this occupation become increasingly related to digital technology, including Digital computer operator, Machine bookkeeper (both added in 1970), Cryptographic machine operator, Photocopying machine operator, Wires transfer clerk, (all added in 1980), Computer operator, terminal; Camera operator, microfilm or microfiche; Computer typesetter; Facsimile machine operator (all added in 1990). In the same time period there are also new titles related to data entry, such as Data coder operator, Data processing, data entry or keying, and Data processing, any other specified or not specified (all added in 1980). New work emergence has dwindled in the most recent decades, with only a small number of examples such as

"... classifying them as new or not new, using a combination of automated assignment and careful manual revision"

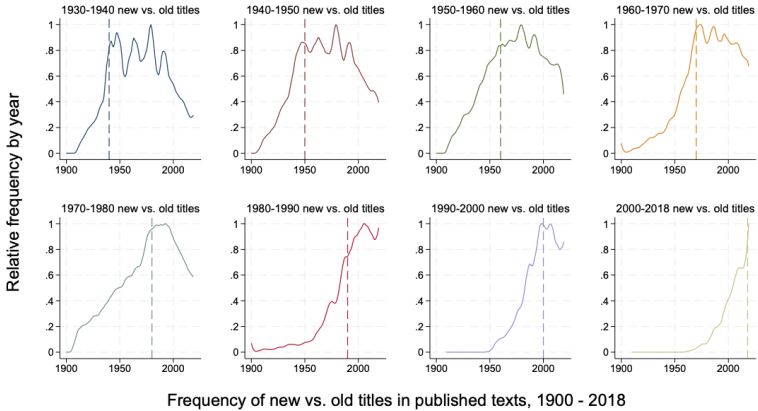
Is it better to use a Generative AI or embeddings?

(And not sure if this is the reason why they had 12 RAs)

Google Ngram Viewer

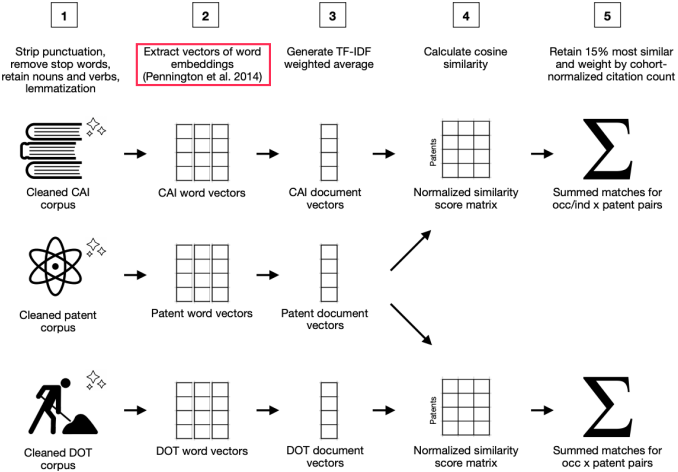
FIGURE A.I

Median Relative Usage Frequency in Published English Language Books of New Occupational Titles added to the *Census Alphabetical Index of Occupations* by Decade, 1940 – 2018



This figure is calculated using Google Ngram Viewer (Michel et al., 2011). It reports the median frequency of each decade's cohort of new titles relative to the median frequency of the decade's cohort of existing titles in digitized English language books published in the United States in every year between 1900 and 2018.

Generate Occupation-level Tech Exposure



Webb (2019)
Kogan et al. (2021)

No explicit explanations on why it works; My guess: DOT is about tasks while CAI titles is about the user

Is it better to use a Generative AI or embeddings or any more explicit ways?

Estimation

$$\ln E\left[\text{Newtitles}_{j,t}\right] = \beta_1 \,\text{AugX}_{j,t} + \beta_2 \,\text{AutX}_{j,t} + \beta_3 \frac{E_{j,t-10}}{\sum_j E_{j,t-10}} + \delta_t \left(+\delta_{J,t}\right)$$

TABLE II
OCCUPATIONAL NEW TITLE EMERGENCE AND AUGMENTATION VERSUS AUTOMATION EXPOSURE, 1940–2018

Dependent Variable: Occupational New Title Count

	(1)	(2)	(3)	(4)	(5)
			A. 1940–2018		
Augmentation Exposure	17.81***	21.46***		16.85***	21.02***
	(3.52)	(3.74)		(3.96)	(3.54)
Automation Exposure			12.75**	1.89	2.35
			(3.93)	(4.52)	(4.07)
			B. 1940–1980		
Augmentation Exposure	23.46***	27.23***		19.48***	26.11***
	(5.09)	(4.80)		(5.79)	(4.45)
Automation Exposure			19.56***	8.15+	9.07+
			(4.21)	(4.85)	(4.87)
			C. 1980–2018		
Augmentation Exposure	8.14*	12.35***		14.87***	13.86***
	(4.08)	(2.31)		(3.72)	(2.08)
Automation Exposure			−1.21	−12.99*	−5.43
			(5.07)	(5.37)	(6.19)
Occ Emp Shares	X	X	X	X	X
Time FE	X		X	X	
Broad Occ × Time FE		X			X

A Detour: Kelly et al. (2021)

- Empirical question: How to measure the novelty and impact of technological innovations in the history using patent text only?
- We can calculate the cosine similarity between any two patents: $\rho_{i,j} = V_{i,t} \cdot V_{j,t}$
- A measure of "backward similarity": $BS_j^\tau = \sum_{i \in \mathcal{B}_{j,\tau}} \rho_{j,i}$
  - $\mathcal{B}_{j,\tau}$ denotes the set of "prior" patents filed in the τ years prior to j's filing
- A measure of "forward similarity": $FS_j^\tau = \sum_{i \in \mathcal{F}_{j,\tau}} \rho_{j,i}$
  - $\mathcal{F}_{j,\tau}$ denotes the set of "post" patents filed over the next τ years following j's filing
- A measure of patent importance: $q_j^\tau = \frac{FS_j^\tau}{BS_j}$
- Define a "breakthrough" patent if top 10% of the estimated q_j^τ

Breakthrough Innovations Varied in Timing Across Industry

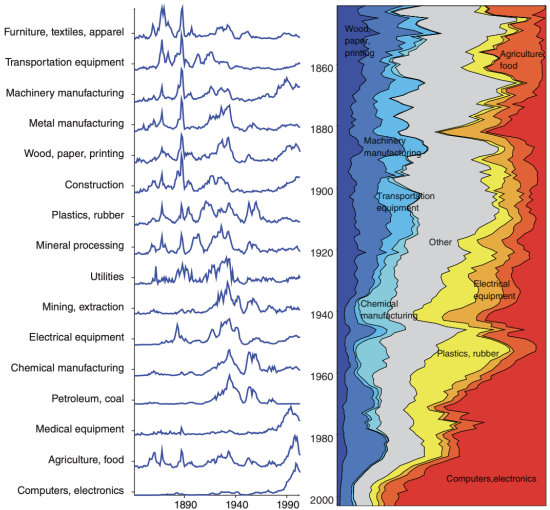
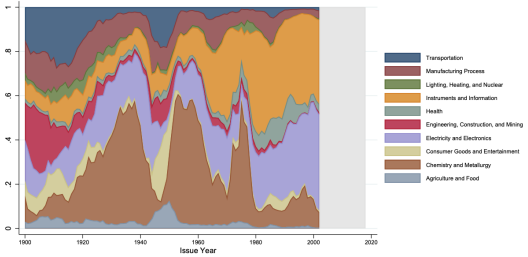
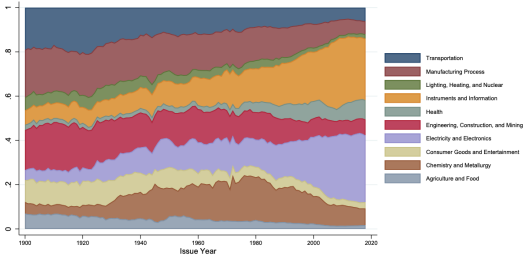


FIGURE 5. BREAKTHROUGH INNOVATION ACROSS INDUSTRIES

Panel A. The flow of top 10% breakthrough patents, 1900–2002



Panel B. The flow of all patents, 1900-2018



IV Estimation

Use exposures to breakthroughs 20-years prior as IVs

TABLE III
First Stage Estimates for New Titles Regressions, 1940–2018

Dependent Variable: Log Patent Count

	(1) Aug	(2) Aut	(3) Aug	Aut
		A. 1940–2018		
Augmentation IV	2.26***		2.20***	0.23
	(0.41)		(0.43)	(0.32)
Automation IV		2.40***	0.67	2.37***
		(0.32)	(0.65)	(0.36)
F-stat	30.88	56.59	27.50	28.96
Sanderson-Windmeijer F-stat	30.88	56.59	49.47	35.39
		B. 1940–1980		
Augmentation IV	1.17+		1.70**	−0.01
	(0.62)		(0.54)	(0.33)
Automation IV		3.45***	2.19+	3.58***
		(0.66)	(1.16)	(0.75)
F-stat	3.63	27.21	8.20	11.46
Sanderson-Windmeijer F-stat	3.63	27.21	8.94	9.29
		C. 1980–2018		
Augmentation IV	2.82***		2.94***	0.78**
	(0.35)		(0.48)	(0.30)
Automation IV		1.66***	−0.82	1.35***
		(0.20)	(0.73)	(0.23)
F-stat	65.73	68.15	40.91	29.68
Sanderson-Windmeijer F-stat	65.73	68.15	62.03	30.37
Occ Emp Shares	X	X	X	X
Time FE	X	X	X	X

$N = 1,276$ in Panel A; $N = 580$ in Panel B; $N = 697$ in Panel C. First stage estimates for columns 1, 2, and 4 in Table

TABLE IV
2SLS Estimates for New Titles Regressions, 1940–2018

Dependent Variable: Log Occupational New Title Count

	(1)	(2)	(3)	(4)	(5)
			A. 1940–2018		
Augmentation Exposure	24.42**	21.30***		29.54***	19.52**
	(7.46)	(5.81)		(8.58)	(6.66)
Automation Exposure			−3.01	−14.56	4.21
			(11.82)	(11.29)	(9.71)
F-stat (Aug)	30.88	60.16		27.50	37.16
F-stat (Aut)			56.59	28.96	52.05
			B. 1940–1980		
Augmentation Exposure	64.64*	30.17***		56.79**	24.82*
	(29.15)	(8.79)		(19.88)	(11.78)
Automation Exposure			9.80	−17.29	15.75
			(15.16)	(17.23)	(17.74)
F-stat (Aug)	3.63	11.04		8.20	8.72
F-stat (Aut)			27.21	11.46	13.79
			C. 1980–2018		
Augmentation Exposure	7.80+	10.50+		21.08**	15.36*
	(4.67)	(6.31)		(7.11)	(5.93)
Automation Exposure			−22.70	−26.73+	−13.43
			(14.99)	(13.60)	(9.80)
F-stat (Aug)	65.73	52.32		40.91	37.44
F-stat (Aut)			68.15	29.68	35.90
Occ Emp Shares	X	X	X	X	X
Time FE	X		X	X	
Broad Occ × Time FE		X			X

$N = 1,276$ in Panel A; $N = 580$ in Panel B; $N = 697$ in Panel C. Coefficients multiplied by 100. Specifications

Horton (2023): LLMs as Economics Agent for Survey/Experiment

- Empirical question: how would employer pick workers when the MW increased?
- The prompt sent to GPT3: (last part to avoid AI's lexical preferences with work experience first)

> You are hiring for the role of "Dishwasher." The typical hourly rate is $12/hour. You have 2 candidates.
>
> Person 1: Has 1 year(s) of experience in this role. Requests $17/hour. Person 2: Has 0 year(s) of experience in this role. Requests $13/hour.
>
> Who would you hire? You have to pick one.
>
> *Person 2. Although they have no experience in this role, their request for $13/hour is closer to the typical rate of $12/hour.*

- Vary scenarios in Person 1's ask wage ($13-19) and if new MW ($15) to generate samples

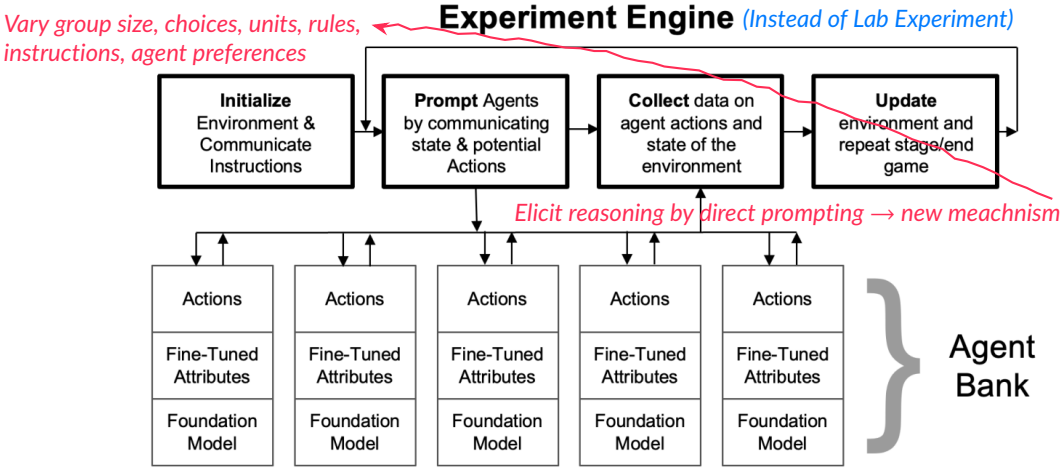
Table 1: Effects of minimum wage on observed wage and hired worker attributes

	Dependent variable:	
	w Hired worker wage	exper Hired worker experience
	(1)	(2)
$15/hour Minimum wage imposed	1.833***	0.167***
	(0.076)	(0.045)
Constant	13.333***	0.667***
	(0.054)	(0.032)
Observations	360	360
R^2	0.621	0.037

Notes: This reports the results of imposing a minimum wage on the (1) hired worker wage and (b) hired worker experience.

Tranchero et al. (2024): LLM Experiment for Testing Theory (use edsl pkg)

Figure 1: Stylized GABE Engine



Note: This figure presents a stylized depiction of the central deterministic engine that powers an interactive experiment using GABE. The engine spawns the AI agents (each consisting of an action space, fine-tuned personality, and

Outline

Why to Use Text-as-Data

- We use text-as-data to study research question of interest otherwise we can't
  - i.e. w/o good structured data to answer
- Tradeoff: novel insights vs. native drawbacks or skepticism
  - Topics already with good data and a strong empirical tradition will lean towards latter *(I feel most pushback on my own work due to this)*
  - The novel insights need to be low-dimensional and interpretable
- E.g. think job ads data has been used in labor economics:
  - capture skills/tasks/technologies/amenities/discrimination not in census/survey data
  - often need to justify sample bias, measure errors, strategic behaviors, ...
  - often as one measure and combined with other census/administrative data
- I guess that's why we see more flourish in politics, media, ... ?

Research with Text-as-Data

- In most cases, we are selling our story but not the techniques
  - A more accurate algorithm to an old question itself does not ensure an economics paper
- Significantly more time to accumulate domain knowledge than to learn NLP tools
  - Often the key to tell a good story
  - AI has made the cost related to learning coding minimal
  - So, if you find some fancy data but don't have the domain knowledge, better not do it or find someone knows it well
- Free of entry in data×method combinations
  - Do it fast and submit it fast
  - Borrow the ideas from applications in other fields (e.g. finance, management, ...)
  - Utilizing most advanced techniques is still rare, not sure frictions or equilibrium

Ways of Using NLP Tools

- Simply use it
  - try both the classic and recent ones
  - see which works
  - a technical tip: often can find highly accelerated pkg for a well-known algorithm
- Select the most suitable methods based on understanding of algorithm and context
  - even though, degree of freedoms in choosing approaches can be high
  - it's often an empirical question
  - convince people it works with simple facts is better than using sophisticated methods
- Incorporate the mechanism into economics model
  - e.g. Gentzkow et al. (2019) style
  - LLM is a new type of probabilistic, generative model w/o explicit rules on DGP

Appendix

Human-rule-based Representation

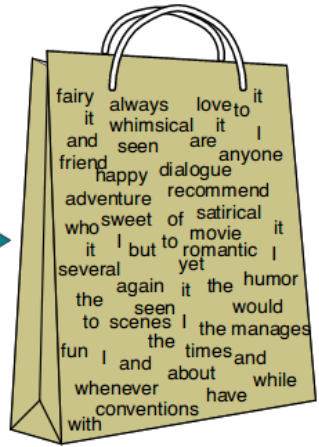
- Idea: represent word as a collection of features and relationships to linguistic categories (grammatical, derivational, semantic) and other words
- E.g.

$$v_{\text{tea}} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \begin{matrix} \text{(plural noun)} \\ \text{(3rd singular verb)} \\ \text{(hyponym-of-beverage)} \\ \vdots \\ \text{(synonym-of-chai)} \end{matrix} \tag{3}$$

- Failures compared to data-driven approaches:
  - updating is costly and they are always incomplete
  - extremely high dimension (much larger than $|V|$) and sparse
  - human ideas of what the right representations tend to underperform

Bag-of-words

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!



it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
…	…

(Source: Force of LSTM and GRU)

TF-IDF

- Term frequency (TF): $TF_{dw} \equiv \frac{\#\text{ token } w \text{ in document } d}{\#\text{ tokens in document } d}$ (frequency)
- Inverse document frequency (IDF): $IDF_w \equiv \log\left(\frac{\#\text{ documents in corpus}}{\#\text{ documents that include term } w}\right)$
- $TF-IDF_{dw} \equiv TF_{dw} \times IDF_w$
- "Backward-IDF" in Kelly et al. (2021) (p for patent instead of d):
  $BIDF_{wp} = \log\left(\frac{\#\text{ patents prior to } p}{1+\#\text{ documents prior to } p \text{ that include term } w}\right)$
  - Idea: e.g. Nikola Tesla's famous 1888 patent introduce the phrase "alternating current," which was used in all following work; Standard IDF would sharply deemphasize this term
- $TFBIDF_{w,i,t} = TF_{w,i} \times BIDF_{w,t}, \quad t \equiv \min(i,j)$ and likewise for patent j
  - Idea: e.g. for a 1990 GM patent of an "alternating current ignition system" (i), and to compare with the Tesla 1888 patent (j), $BIDF_{w,t=1990}$ will deemphasize this term for i
- Calculate the cosine similarity: $\rho_{i,j} = V_{i,t} \cdot V_{j,t}$, where $V_{k,t} = \frac{TFBIDF_{k,t}}{\|TFBIDF_{k,t}\|}$

Stochastic Gradient Descent

- From the CBOW model, we have the global loss as
  $\text{minimize}\, J = \sum_{D_1,\ldots,D_M} \sum_{i=1}^{n} -\log P\left(w_i \mid w_{i-m},\ldots,w_{i-1},w_{i+1},\ldots,w_{i+m}\right)$
- Compute the gradients with respect to the unknown parameters and at each iteration update them via gradient descent: $\mathbf{U}^{(t+1)} = \mathbf{U}^{(t)} - \alpha \nabla_{\mathbf{U}} J\left(\mathbf{U}^{(t)}, \mathbf{W}^{(t)}\right)$
- Computing $J(\mathbf{U}, \mathbf{W})$ is however expensive as it walks over the entire dataset
- Instead perform stochastic gradient descent: for each step, approximating $J(\mathbf{U}, \mathbf{W})$ using a few sampling documents $d_1, \ldots, d_\ell \sim D$ and computing $\hat{J}(\mathbf{U}, \mathbf{W}) = \sum_{d_1,\ldots,d_\ell} \sum_{i=1}^{n} -\log P_{\mathbf{U},\mathbf{W}}\left(w_i^{(d)} \mid \mathbf{c}_i^{(d)}\right)$ and gradients
- Further simplification of the calculation of each P involves a technique called negative sampling to avoid walking over the entire vocabulary in the denominator

Word2vec: Skip-gram

- For the context: $\mathbf{c}_i = \{w_{i-m}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+m}\}$, where w_i is center word
- The aim is again to find two mappings $\mathbf{U} \in \mathbb{R}^{H \times |V|}$ and $\mathbf{W} \in \mathbb{R}^{|V| \times H}$
- $u_i = \mathbf{U} w_i \in \mathbb{R}^h$; $z = \mathbf{W} u_i \in \mathbb{R}^{|V|}$; $\hat{y} = \mathrm{softmax}(z) \in \mathbb{R}^{|V|}$
- Loss function:

$$
\begin{aligned}
J_i^{(d)} &= -\log P\left(w_{i-m}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+m} \mid w_i\right) \\
&= -\log \prod_{j=0, j \neq m}^{2m} P\left(w_{i-m+j} \mid w_i\right) = -\log \prod_{j=0, j \neq m}^{2m} \frac{\exp\left(v_{i-m+j}^T u_i\right)}{\sum_{k=1}^{|V|} \exp\left(v_k^T u_i\right)} \\
&= -\sum_{j=0, j \neq m}^{2m} v_{i-m+j}^T u_i + 2m \log \sum_{k=1}^{|V|} \exp\left(v_k^T u_i\right)
\end{aligned}
$$

  - v_i is the corresponding row of token i in $\mathbf{W}$
  - 2nd line invokes a Naive Bayes assumption to break out the probabilities

GloVe

- LSA is count-based and relies on matrix factorization of global co-occurrence statistics
  - capture word similarities but do poorly on word analogy
- Word2vex is window-based and makes local predictions in local context windows
  - capture complex linguistic patterns but fail to use global co-occurrence statistics
- GloVe is a mix of these two:
  - Recall in skip-gram the global cross-entropy loss is $\hat{J} = -\sum_{i \in \text{ corpus }} \sum_{j \in \text{ context }(i)} \log \hat{y}_{ij}$, which is same as $\hat{J} = -\sum_{i=1}^{|V|} \sum_{j=1}^{|V|} X_{ij} \log y_{ij}$, where X_{ij} is from co-occurrence matrix
  - Instead of using the softmax for $\hat{y}$, GloVe uses a weighted least square objective $\hat{J} = \sum_{i=1}^{|V|} \sum_{j=1}^{|V|} f\left(X_{ij}\right) \left(v_j^T u_i - \log X_{ij}\right)^2$
  - This way also avoids the expensive summation required for the denominator of softmax
  - It was argued to outperform on word analogy and word similarity tasks but see the debates here and here
- It turns out that all these methods de facto factorize some word-context matrices

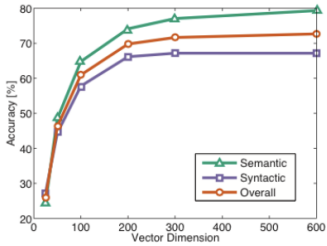
Intrinsic Evaluation for Tuning Hyperparameters

Model	Dimension	Size	Semantics	Syntax	Total
ivLBL	100	1.5B	55.9	50.1	53.2
HPCA	100	1.6B	4.2	16.4	10.8
GloVE	100	1.6B	67.5	54.3	60.3
SG	300	1B	61	61	61
CBOW	300	1.6B	16.1	52.6	36.1
vLBL	300	1.5B	54.2	64.8	60.0
ivLBL	300	1.5B	65.2	63.0	64.0
GloVe	300	1.6B	80.8	61.5	70.3
SVD	300	6B	6.3	8.1	7.3
SVD-S	300	6B	36.7	46.6	42.1
SVD-L	300	6B	56.6	63.0	60.1
CBOW	300	6B	63.6	67.4	65.7
SG	300	6B	73.0	66.0	69.1
GloVe	300	6B	77.4	67.0	71.7
CBOW	1000	6B	57.3	68.9	63.7
SG	1000	6B	66.1	65.1	65.6
SVD-L	300	42B	38.4	58.2	49.2
GloVe	300	42B	81.9	69.3	75.0

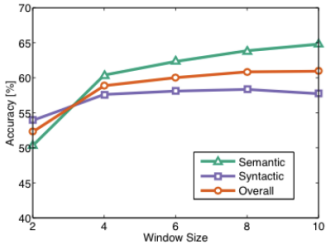
Table 5: Here we compare the performance of different models under the use of different hyperparameters and datasets

(Performance depends on model, corpus size, and vector dimension)

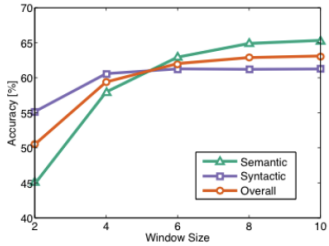
Intrinsic Evaluation for Tuning Hyperparameters



(a) Symmetric context



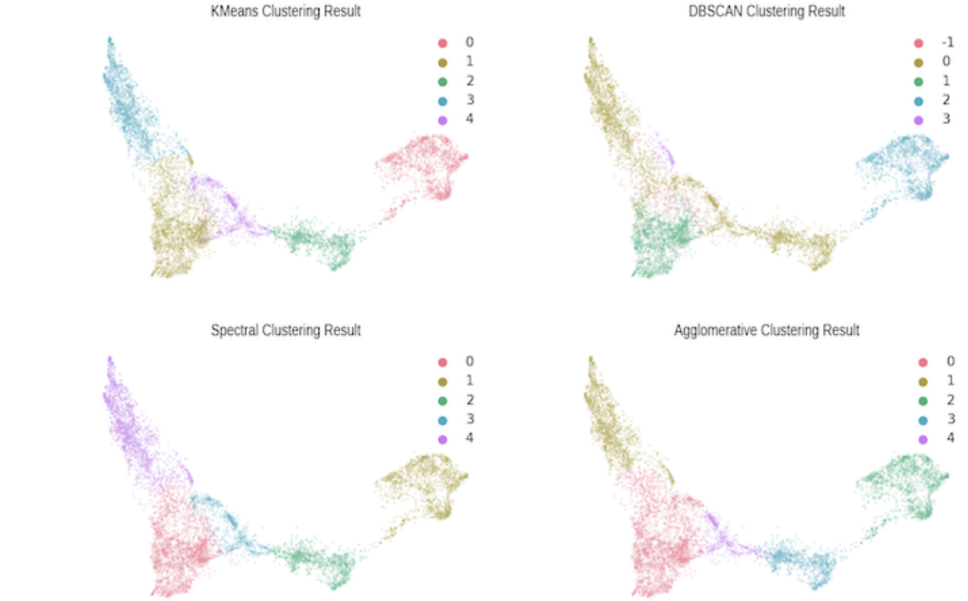
(b) Symmetric context



(c) Asymmetric context

Figure 4: We see how accuracies vary with vector dimension and context window size for GloVe

Embedding Vectors in Reduced Dimension: Document Similarity



Activation Functions

- Sigmoid: $\sigma(z) = \frac{1}{1+\exp(-z)}$ where $\sigma(z) \in (0,1)$ and $\sigma'(z) = \frac{-\exp(-z)}{1+\exp(-z)} = \sigma(z)(1-\sigma(z))$
- Tanh: $\tanh(z) = \frac{\exp(z)-\exp(-z)}{\exp(z)+\exp(-z)} = 2\sigma(2z) - 1$, where $\tanh(z) \in (-1,1)$ and $\tanh'(z) = 1 - \tanh^2(z)$
- ReLu: $\text{rect}(z) = \max(z, 0)$, where $\text{rect}'(z) = \begin{cases} 1 & : z > 0 \\ 0 & : \text{ otherwise} \end{cases}$ and

  $\text{rect}'(z) = \begin{cases} 1 & : z > 0 \\ 0 & : \text{ otherwise} \end{cases}$
- Leaky ReLU: $\text{leaky}(z) = \max(z, k \cdot z)$, where $0 < k < 1$ and

  $\text{leaky}'(z) = \begin{cases} 1 & : z > 0 \\ k & : \text{ otherwise} \end{cases}$

Objective Function for NN

- Maximum margin is a popular optimization objective (error metric) for NNs: $\text{minimize } J = \max(s_c - s, 0)$
  - s is the score for "true" labeled data and s_c is the score for "false" labeled data
  - it only cares the the "true" data point have a higher score than the "false" data point and that the rest does not matter (not like a softmax)

- It is often modified to have a margin of safety: $\text{minimize } J = \max(\Delta + s_c - s, 0)$
  - $\Delta > 0$ to avoid the optimization objective being too risky
  - Δ can be normalized to 1

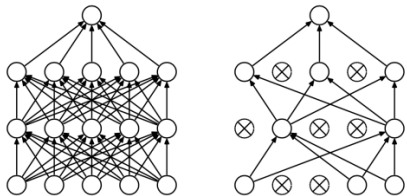
- It is popular because we have: $\frac{\partial J}{\partial s} = -\frac{\partial J}{\partial s_c} = -1$ when $J > 0$
  - convenient for calculating loss gradients

- Again, it is common to add an regularization penalty to address overfitting: $J_R = J + \lambda \sum_{i=1}^{L} \left\| W^{(i)} \right\|_F$
  - $\left\| W^{(i)} \right\|_F$ is the Frobenius norm (L_2) of the i-th weight matrix in the network

Dropout

- Dropout can effectively act as another form of regularization:
  - during training, randomly "drop" with some probability $(1 - p)$ a subset of neurons during each forward/backward pass
  - during testing, use the full network to compute our predictions

- The result is that the network typically learns more meaningful information
  - Intuitive reason: essentially it's training exponentially many smaller networks at once and averaging over their predictions



Dropout applied to an artificial neural network. Image credits to Srivastava et al.

Learning Strategy

- The rate of model parameter updates during training can be controlled using the learning rate: i.e. α in Gradient Descent formulation: $\theta^{\text{new}} = \theta^{\text{old}} - \alpha \nabla_\theta J_t(\theta)$
- If α is too large:
  - overshoot the convex minima
  - diverging loss functions
- If α is too low:
  - not converge in a reasonable amount of time
  - caught in local minima
- Efficient strategies to tune α:
  - scaling by the inverse square root of the fan-in of the neuron
  - annealing: after several iterations, α is reduced in some way
  - Momentum methods: use the "velocity" of updates as a more effective update scheme
  - AdaGrad: parameters with a scarce history of updates are updated faster

Deep Bidirectional RNNs

$$\vec{h}_t = f\left(\vec{W}x_t + \vec{V}\vec{h}_{t-1} + \vec{b}\right)$$

$$\overleftarrow{h}_t = f\left(\overleftarrow{W}x_t + \overleftarrow{V}\overleftarrow{h}_{t+1} + \overleftarrow{b}\right)$$

$$\hat{y}_t = g\left(Uh_t + c\right) = g\left(U\left[\vec{h}_t; \overleftarrow{h}_t\right] + c\right)$$

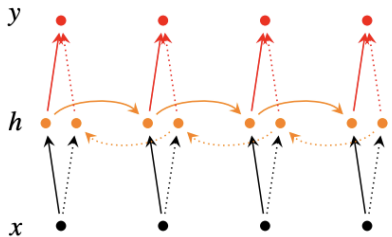


Figure 8: A bi-directional RNN model

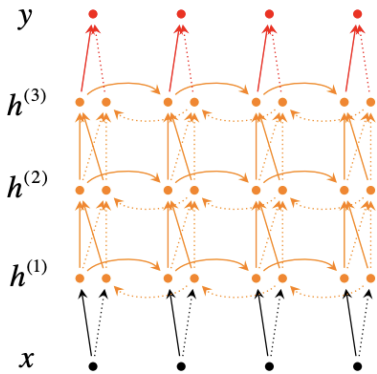
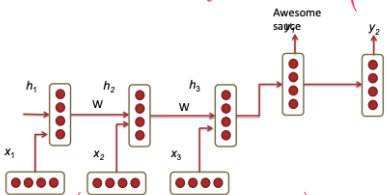


Figure 9: A deep bi-directional RNN with three RNN layers.

RNN Translation Model

$$h_t = \phi(h_{t-1}) = f\left(W^{(hh)} h_{t-1}\right)$$

$$y_t = \text{softmax}\left(W^{(S)} h_t\right)$$



$$h_t = \phi(h_{t-1}, x_t) = f\left(W^{(hh)} h_{t-1} + W^{(hx)} x_t\right)$$

Figure 10: A RNN-based translation model. The first three RNN hidden layers belong to the source language model encoder, and the last two belong to the destination language model decoder.

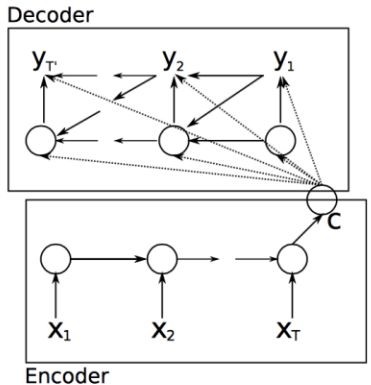
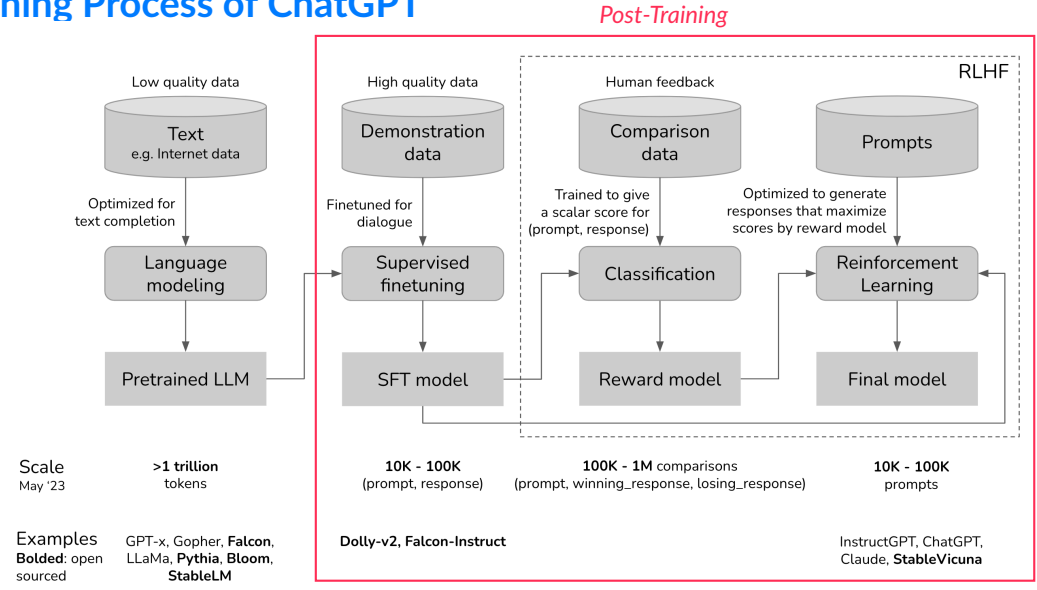


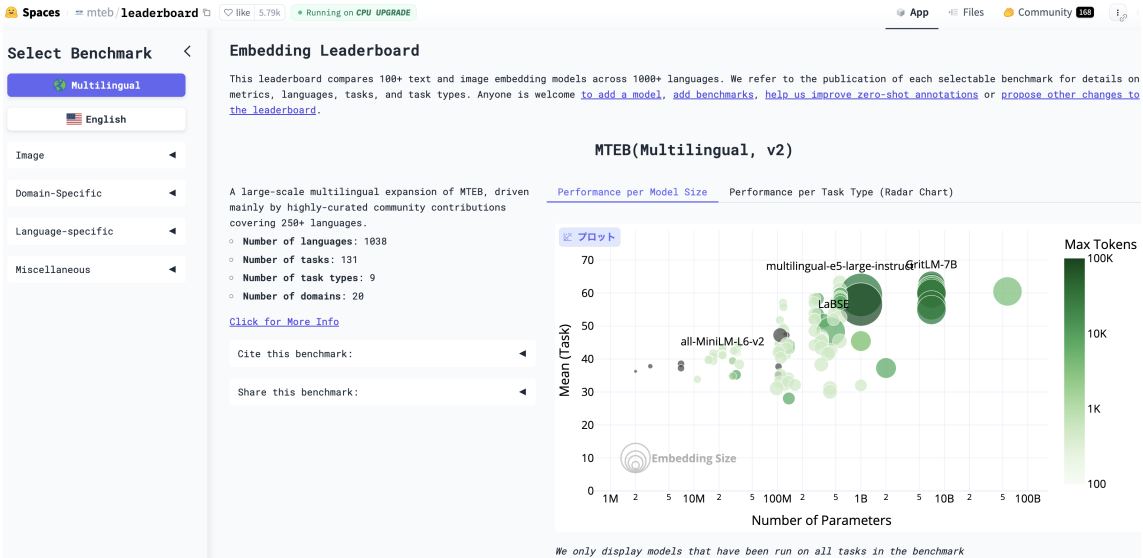
Figure 11: Language model with three inputs to each decoder neuron: (h_{t-1}, c, y_{t-1})

Training Process of ChatGPT



(source: RLHF: Reinforcement Learning from Human Feedback)

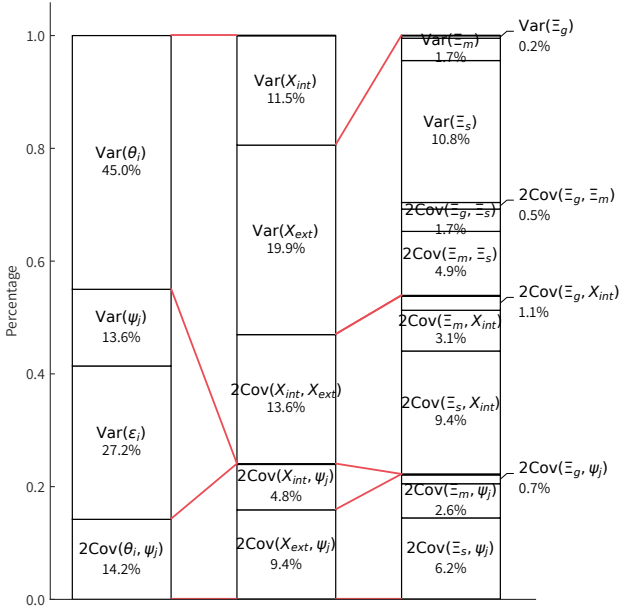
Model Selection



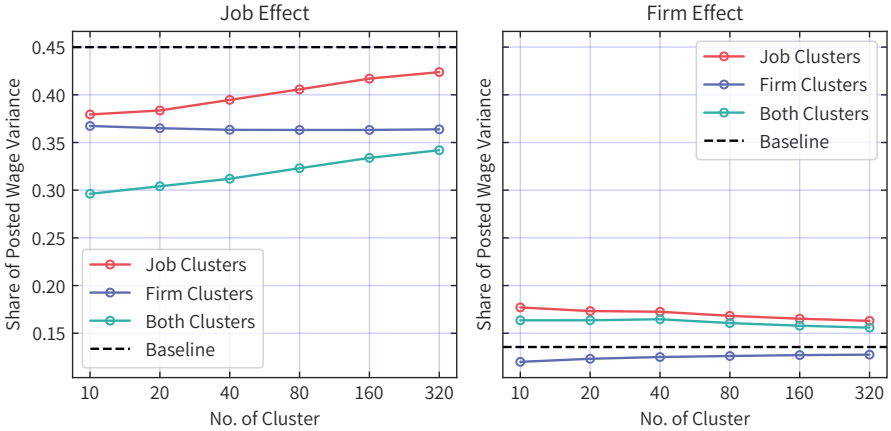
Lise and Postel-Vinay (2020): Interpretable PCA

- Say a set of P different skill measures observed for N occupations
- PCA decomposes the matrix $\mathbf{M} \in \mathbb{R}^{N \times P}$ as $\mathbf{M} = \mathbf{FL}$
  - $\mathbf{F} \in \mathbb{R}^{N \times P}$ is the orthonormal matrix of principal eigenvectors of $\mathbf{MM}^\top$
  - $\mathbf{L} \in \mathbb{R}^{P \times P}$ is a matrix of factor loadings
  - ($\mathbf{F} \equiv \mathbf{U}, \mathbf{L} \equiv \mathbf{\Sigma V}^\top$ in SVD $\mathbf{M} = \mathbf{U\Sigma V}^\top$)
  - If use first 3 principal components only, decompose as $\mathbf{M} = \mathbf{F}_3\mathbf{L}_3 + \mathbf{U}$
  - $\mathbf{F}_3\mathbf{L}_3 \in \mathbb{R}^{N \times P}$ is the reconstructed raw data based on 3 PCs that capture most variances but not interpretable
- The decomposition can rewritten as $\mathbf{M} = (\mathbf{F}_3\mathbf{L}_{3,3}) \left(\mathbf{L}_{3,3}^{-1}\mathbf{L}_3\right) + \mathbf{U}$
  - $\mathbf{F}_3\mathbf{L}_{3,3}$ is the loaded principal components based on first three columns of $\mathbf{M}$
  - E.g. set 1st vector to mathematics score to only reflect cognitive skill; set 2nd vector to social perceptiveness score to only reflect interpersonal skill; ...
  - Exclusion restrictions is achieved as the new loading $\mathbf{L}_3^{\text{new}} \equiv \mathbf{L}_{3,3}^{-1}\mathbf{L}_3$ has its $\mathbf{L}_{3,3}^{\text{new}}$ to be the identity matrix
  - Advantage: no need to decide which measure in P belongs to which board category

Zhu (2023): Variance Decomposition with PLS Variables



Zhu (2023): Variance Decomposition with Artificial Occupations



Zhu (2023): Replicate DK2018 with Full Features

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Cognitive	.045	.054	.027	.047	.013	.032	.011	.033
	(.000)	(.001)	(.000)	(.001)	(.000)	(.001)	(.000)	(.001)
Social	.035	.041	.030	.045	.020	.033	.025	.041
	(.001)	(.001)	(.001)	(.001)	(.000)	(.001)	(.001)	(.001)
Both required		-.012		-.026		-.024		-.029
		(.001)		(.001)		(.001)		(.001)
Ξ_g, Ξ_m			✓	✓			✓	✓
Ξ_s					✓	✓	✓	✓
Education FE	✓	✓	✓	✓	✓	✓	✓	✓
Experience FE	✓	✓	✓	✓	✓	✓	✓	✓
Occupation FE	✓	✓	✓	✓	✓	✓	✓	✓
Year FE	✓	✓	✓	✓	✓	✓	✓	✓
Adj. R^2	.582	.582	.604	.604	.636	.636	.641	.641

Hampole et al. (2025): AI Exposure at Firm-Task Level

Figure 1: Illustration of process for identifying AI applications from resumes and exposed occupation tasks

Example resume job description of a worker employed at JP Morgan:

> Technology delivery lead for risk and fraud forecasting models in auto, card, and home lending businesses. AI/ML model delivery in public cloud, private cloud and on prem. managing credit risk deployment services platform with continuous delivery, development and deployment of quantitative risk models that serve regulatory and credit risk assessments.

Step 1: Identify AI-related terms (if any)

"Technology delivery lead for risk and fraud forecasting models in auto, card, and home lending businesses. **AI/ML** model delivery in public cloud, private cloud and on prem. managing credit risk deployment services platform with continuous delivery, development and deployment of quantitative risk models that serve regulatory and credit risk assessments."

Step 2: Use large language models to extract the phrases likely to contain specific AI applications

"**Technology delivery lead for risk and fraud forecasting models in auto, card, and home lending businesses.** AI/ML model delivery in public cloud, private cloud and on prem. Managing credit risk deployment services platform with continuous delivery, **development and deployment of quantitative risk models that serve regulatory and credit risk assessments.**"

Step 3: Use large language models to clean the extracted AI applications

Extracted phrase: "Technology delivery lead for risk and fraud forecasting models in auto, card, and home lending businesses."

> Cleaned AI application: **"Forecast risk and fraud in various lending businesses, including auto, card, and home lending."**

Extracted phrase: "Development and deployment of quantitative risk models that serve regulatory and credit risk assessments."

> Cleaned AI application: **"Assess credit risk and provide regulatory compliance across different lines of business."**

Step 4: Use GTE sentence embeddings to measure textual similarity to identify highly exposed tasks

AI application: "Forecast risk and fraud in various lending businesses, including auto, card, and home lending."

> Most exposed O*NET occupation task by cosine similarity: **"Prepare reports that include the degree of risk involved in extending credit or lending money."** (Credit Analysts, SOC code = 132041)

AI application: "Development and deployment of quantitative risk models that serve regulatory and credit risk assessments."

> Most exposed O*NET occupation task by cosine similarity: **"Analyze credit data and financial statements to determine the degree of risk involved in extending credit or lending money."** (Credit Analysts, SOC code = 132041)

Note: This figure shows an example of our process for identifying AI applications from online resumes and linking with exposed job tasks. See section 2 in the main text and appendix A.2 for further details.

Hansen et al. (2023): Classify Job Ads for WFH *(website)*

Method

Our **Large Language Model (LLM)** is built using the DistilBERT model.

This LLM is pre-trained on the **entire English-language Wikipedia corpus**, which helps the framework interpret the intended meaning of a given document or passage.

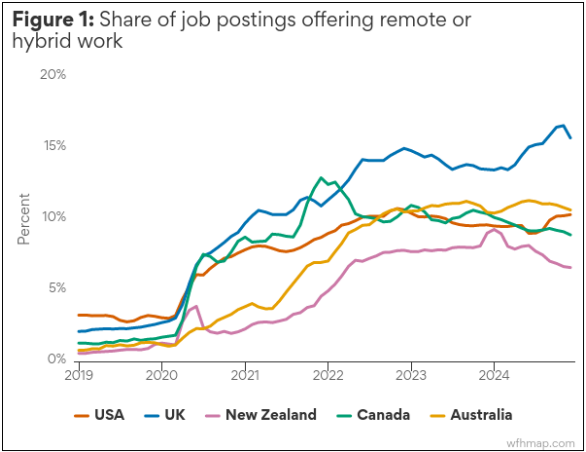
We further pre-train this model on roughly **one million text sequences** drawn from our corpus of online vacancy postings. This ensures the **language model is familiar with the language of job ad text**.

Finally, we use **30,000 human-coded text extracts** from job ads. Our human auditors were asked to flag text which indicates an offer of remote work. We then use this to **train the model** to identify jobs which offer remote work.

We also use the human-coded extracts to evaluate the predictive performance of the model. We find that the final model has **99% accuracy** relative to human beings.

For further information about our method, including a **comparison of its performance** relative to other text-algorithms (including **recent Generative AI** models), see our paper: ***"Remote Work across Jobs, Companies, and Space"*** (2023).

Researchers and other non-commercial users can **contact us to gain access to the** underlying code and information used to construct the WHAM model.



Use ML to Find Novel and Interpretable Hypothesis

- Ludwig and Mullainathan (2024): "Machine Learning as a Tool for Hypothesis Generation"
  - Researchers do exploratory "data" analysis to generate hypotheses
  - ML algorithms help to automatically detect patterns, esp. ones that are never considered
  - A key challenge: generate human-interpretable hypotheses that are novel & testable

- Their application: Find novel features that explain judge's jailing decision
  1. A DL model finds a striking fact: defendant's face has large explanatory power
  2. Control for all known features to ensure the finding is truely novel
  3. Generate synthetic images that vary in new features; Train independent study subjects in an experimental design; Ask the subjects to name the features *("well-groomed", "heavyfaced")*

- Some thoughts: *(many are recognized in the paper)*
  - Including facial images as input data is itself a researcher-driven hypothesis
  - ML/DL works well as the DGP in real world is high dimensional and non-linear
  - In essence, it's about utilizing new unstructured data and (human-)interpreting ML results
  - The interpretation steps are not that different from those in BERTopic
  - New notions are often distilled by experts but not by people w/o domain knowledge
  - If new features can be named well, why would they be novel to practitioners?

Interpret Features of Generative LLMs (but more dark matter)

Feature #34M/31164353 **Golden Gate Bridge** feature example

The feature activates strongly on English descriptions and associated concepts

in the Presidio at the end (that's the⏎ huge park right next to the Golden Gate bridge), perfect. But not all people

repainted, roughly, every dozen years." "while across the country in san fran cisco, the golden gate bridge was

it is a suspension bridge and has similar coloring, it is often⏎> compared to the Golden Gate Bridge in San Francisco, US

They also activate in multiple other languages on the same concepts

ゴールデン・ゲート・ブリッジ、金門橋は、ア メリカ西海岸のサンフランシスコ湾と太平洋が 接続するゴールデンゲート海

골든게이트 교 또는 금문교 는 미국 캘리포니아주 골든게이트 해협에 위치한 현수교이다. 골든게이 트 교는 캘리포니아주 샌프란시

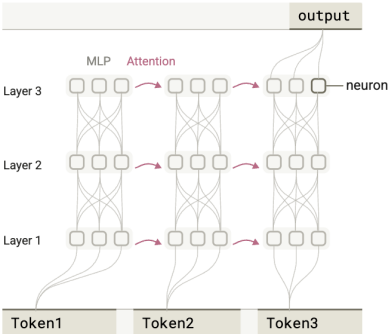
мост золотые ворота — висячий мост через пролив золотые ворота. он со единяет город сан-фран

And on relevant images as well

Open the Blackbox of Generative LLMs

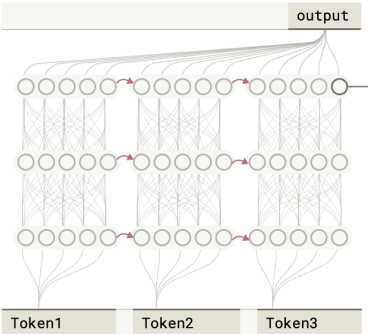
Original Transformer Model

The underlying model that we study is a transformer-based large language model.

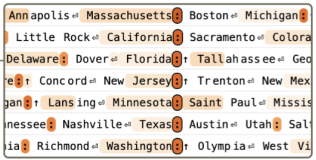


Replacement Model

We replace the neurons of the original model with *features*. There are typically more features than neurons. Features are sparsely active and often represent interpretable concepts.



Feature



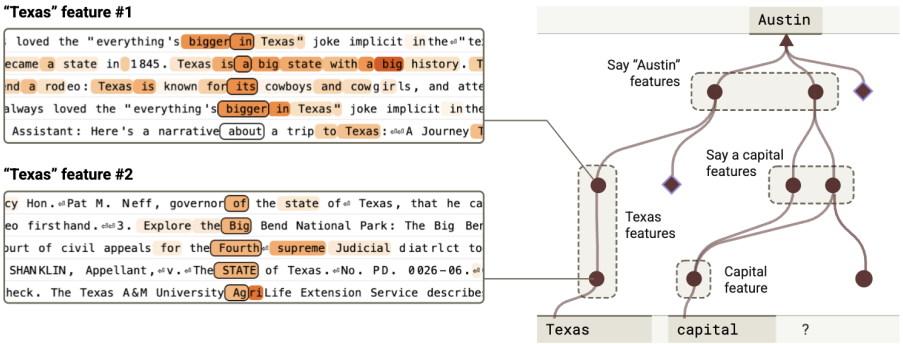
To understand what a feature represents, we use a *feature visualization*, which shows dataset examples for which the feature is most strongly active. In this example, the feature fires strongly when the model is about to say a state capital.

Figure 1: The replacement model is obtained by replacing the original model's neurons with the cross-layer transcoder's sparsely-active features.

Open the Blackbox of Generative LLMs

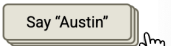
Group Related Nodes Into "Supernodes"

We group together features with related meanings that appear to play similar roles in the graph.



Supernodes

Throughout the paper, we represent supernodes as stacked boxes



Hover over nodes for detailed feature visualizations. Select a feature to view in the top bar after hovering

Figure 3: Grouping related graph nodes into supernodes produces a simpler graph.

Is the "dimension reduction" in Economics necessarily?

- Think about LDA vs. Transformers
- Both are probabilistic, generative models
- LDA models the DGP explicitly; Transformers approximate the data distribution flexibly
- LDA replies on interpretable but simple, arbitrary assumptions on DGP; Transformers don’t
- Classical DGP: A generative story humans invent to explain phenomena
- Neural DGP: A compressed statistical representation of observed data
- "All models are wrong, some are useful"

Reference I

Autor, D., C. Chin, A. Salomons, and B. Seegmiller (2024). New frontiers: The origins and content of new work, 1940–2018. *The Quarterly Journal of Economics 139*(3), 1399–1465.

Deming, D. and L. B. Kahn (2018). Skill requirements across firms and labor markets: Evidence from job postings for professionals. *Journal of Labor Economics 36*(S1), S337–S369.

Dube, A., J. Jacobs, S. Naidu, and S. Suri (2020). Monopsony in online labor markets. *American Economic Review: Insights 2*(1), 33–46.

Gentzkow, M., J. M. Shapiro, and M. Taddy (2019). Measuring group differences in high-dimensional choices: method and application to congressional speech. *Econometrica 87*(4), 1307–1340.

Hampole, M., D. Papanikolaou, L. D. Schmidt, and B. Seegmiller (2025). Artificial intelligence and the labor market. Technical report, National Bureau of Economic Research.

Hansen, S., P. J. Lambert, N. Bloom, S. J. Davis, R. Sadun, and B. Taska (2023). Remote work across jobs, companies, and space. Technical report, National Bureau of Economic Research.

Horton, J. J. (2023). Large language models as simulated economic agents: What can we learn from homo silicus? Technical report, National Bureau of Economic Research.

Kelly, B., D. Papanikolaou, A. Seru, and M. Taddy (2021). Measuring technological innovation over the long run. *American Economic Review: Insights 3*(3), 303–320.

Kogan, L., D. Papanikolaou, L. D. Schmidt, and B. Seegmiller (2021). *Technology-skill complementarity and labor displacement: Evidence from linking two centuries of patents with occupations*. National Bureau of Economic Research.

Reference II

Lise, J. and F. Postel-Vinay (2020). Multidimensional skills, sorting, and human capital accumulation. *American Economic Review 110*(8), 2328–2376.

Ludwig, J. and S. Mullainathan (2024). Machine learning as a tool for hypothesis generation. *The Quarterly Journal of Economics 139*(2), 751–827.

Tranchero, M., C.-F. Brenninkmeijer, A. Murugan, and A. Nagaraj (2024). Theorizing with large language models. Technical report, National Bureau of Economic Research.

Webb, M. (2019). The impact of artificial intelligence on the labor market. *Available at SSRN 3482150*.

Zhu, X. (2023). Posted wage inequality.