Data Science for Public Policy

Unsupervised ML and Text Data

ETHZ Zurich

26/03/2025

Outline

Unsupervised Learning

- **Unsupervised learning** is a type of machine learning where the goal is to discover patterns in data **without any labeled** examples
- Unlike supervised learning, there are no target variables to predict, and the algorithm must find patterns and structure in the data on its own

Unsupervised Learning

- **Unsupervised learning** is a type of machine learning where the goal is to discover patterns in data **without any labeled** examples
- Unlike supervised learning, there are no target variables to predict, and the algorithm must find patterns and structure in the data on its own
- It can be used for tasks such as clustering, anomaly detection, and dimensionality reduction.

Dimensionality reduction

- **Why do we need dimensionality reduction?**

Dimensionality reduction

- **Why do we need dimensionality reduction?**
  - ML problems often involve thousands of features.
  - Especially in the case of text data.

Dimensionality reduction

- **Why do we need dimensionality reduction?**
  - ML problems often involve thousands of features.
  - Especially in the case of text data.
  - Need for computational tractability and finding a good solution

Dimensionality reduction

- **Why do we need dimensionality reduction?**
  - ML problems often involve thousands of features.
  - Especially in the case of text data.
  - Need for computational tractability and finding a good solution
- Can be used as a descriptive tool.
  - Extract information from the data and visualize it.
  - Discover subgroups among the variables or the observations

Dimensionality reduction

- **Why do we need dimensionality reduction?**
  - ML problems often involve thousands of features.
  - Especially in the case of text data.
  - Need for computational tractability and finding a good solution
- Can be used as a descriptive tool.
  - Extract information from the data and visualize it.
  - Discover subgroups among the variables or the observations
- Examples
  - Dimension reduction for pre-processing
  - Costumer segmentation in marketing

Principal Component Analysis (PCA)

- PCA is a technique for reducing the dimensionality of high-dimensional data

Principal Component Analysis (PCA)

- PCA is a technique for reducing the dimensionality of high-dimensional data
- **Goals:**
  - Summarize a large set of features with a smaller number of representative ones

Principal Component Analysis (PCA)

- PCA is a technique for reducing the dimensionality of high-dimensional data
- **Goals:**
  - Summarize a large set of features with a smaller number of representative ones
  - Find a low-dimensional representation of the data that captures as much of the information possible

Principal Component Analysis (PCA)

- PCA is a technique for reducing the dimensionality of high-dimensional data
- **Goals:**
  - Summarize a large set of features with a smaller number of representative ones
  - Find a low-dimensional representation of the data that captures as much of the information possible
- **How it works:**
  1. Identifies the axis that accounts for the largest amount of variance in the data

Principal Component Analysis (PCA)

- PCA is a technique for reducing the dimensionality of high-dimensional data
- **Goals:**
  - Summarize a large set of features with a smaller number of representative ones
  - Find a low-dimensional representation of the data that captures as much of the information possible
- **How it works:**
  1. Identifies the axis that accounts for the largest amount of variance in the data
  2. Finds a second axis, orthogonal to the first, that accounts for the largest amount of the remaining variance
  3. And so on...

Principal Component Analysis (PCA)

- PCA is a technique for reducing the dimensionality of high-dimensional data
- **Goals:**
  - Summarize a large set of features with a smaller number of representative ones
  - Find a low-dimensional representation of the data that captures as much of the information possible
- **How it works:**
  1. Identifies the axis that accounts for the largest amount of variance in the data
  2. Finds a second axis, orthogonal to the first, that accounts for the largest amount of the remaining variance
  3. And so on...
- The unit vector defining the i^{th} axis is called the i^{th} principal component.

Principal Component Analysis

What are we after?

Principal Component Analysis

What are we after?

- Each of the dimensions found by PCA is a linear combination of the p features
- The first principal component of a set of features $X_1, X_2, ..., X_p$ is the normalized linear combination of the features:

$$Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + ... + \phi_{p1}X_p$$

Principal Component Analysis

What are we after?

- Each of the dimensions found by PCA is a linear combination of the p features
- The first principal component of a set of features $X_1, X_2, ..., X_p$ is the normalized linear combination of the features:

$$Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + ... + \phi_{p1}X_p$$

- ...that has the largest variance $\sim$ the largest amount of info

Principal Component Analysis

What are we after?

- Each of the dimensions found by PCA is a linear combination of the p features
- The first principal component of a set of features $X_1, X_2, ..., X_p$ is the normalized linear combination of the features:

$$Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + ... + \phi_{p1}X_p$$

- ...that has the largest variance $\sim$ the largest amount of info
- $\phi_1 = (\phi_{11}, \phi_{21}, ..., \phi_{p1})^T$ is the **loading vector** of the first principal component, where $\sum_{j=1}^{p} \phi_{j1}^2 = 1$
- **The loading vector** represents the weights of the original variables that make up each principal component.

Principal Component Analysis - Computing the First PC

Maximizing the Sample Variance of Z_1:

- We want to find the values of $\phi_{11}, \phi_{21}, ..., \phi_{p1}$ that maximize the sample variance of Z_1, subject to the constraint that $\sum_{j=1}^{p} \phi_{j1}^2 = 1$
- We can write the optimization problem as:

$$\max_{\phi_{11},\phi_{21},...,\phi_{p1}} \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_{j1} x_{ij} \right)^2$$

$$\text{subject to } \sum_{j=1}^{p} \phi_{j1}^2 = 1$$

Principal Component Analysis - Computing the First PC

- We can rewrite the objective function as:

$$\frac{1}{n}\sum_{i=1}^{n} z_{i1}^2,$$

  where z_{i1} is the ith observation's value for the first principal component, and $z_{i1} = \phi_{11}x_{i1} + \phi_{21}x_{i2} + ... + \phi_{p1}x_{ip}$.
- Since the data has mean zero, we have $\frac{1}{n}\sum_{i=1}^{n} x_{ij} = 0$
- Using eigen decomposition (outside the scope of the class)

Principal Component Analysis - Computing the First PC

- We can rewrite the objective function as:

$$\frac{1}{n}\sum_{i=1}^{n} z_{i1}^2,$$

  where z_{i1} is the ith observation's value for the first principal component, and $z_{i1} = \phi_{11}x_{i1} + \phi_{21}x_{i2} + ... + \phi_{p1}x_{ip}$.
- Since the data has mean zero, we have $\frac{1}{n}\sum_{i=1}^{n} x_{ij} = 0$
- Using eigen decomposition (outside the scope of the class)
- $z_{11}, ..., z_{n1}$ are the **scores** of the first principal component
- The **score** represents the contribution of each observation to each principal component
- Solved using Singular Value Decomposition (SVD) [a standard linear algebra tool]

Principal Component Analysis - Computing the Second PC

Second Principal Component

- We can get the second principal component by finding the linear combination of the features that has the largest variance, subject to the constraint that it is orthogonal to the first principal component

Principal Component Analysis - Computing the Second PC

Second Principal Component

- We can get the second principal component by finding the linear combination of the features that has the largest variance, subject to the constraint that it is orthogonal to the first principal component

- Z_2 is the linear combination of $X_1, X_2, ..., X_p$:

$$Z_2 = \phi_{12}X_1 + \phi_{22}X_2 + ... + \phi_{p2}X_p$$

- Z_2 has maximal variance out of all linear combinations uncorrelated with Z_1

Principal Component Analysis - Computing the Second PC

Second Principal Component

- We can get the second principal component by finding the linear combination of the features that has the largest variance, subject to the constraint that it is orthogonal to the first principal component

- Z_2 is the linear combination of $X_1, X_2, ..., X_p$:

$$Z_2 = \phi_{12}X_1 + \phi_{22}X_2 + ... + \phi_{p2}X_p$$

- Z_2 has maximal variance out of all linear combinations uncorrelated with Z_1

- To ensure that the second principal component is orthogonal to the first principal component, we need to add the constraint that:

$$\Phi_1^T \Phi_2 = 0$$

Principal Component Analysis - Projection on a 2D space

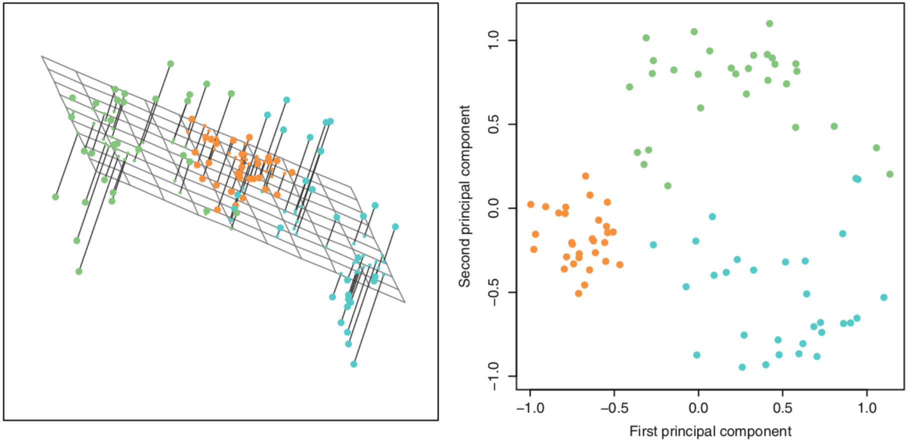


Figure 1: Illustration in 3D, projected on a 2D space.

- **Left:** Simulated data in 3 dimensions.
- **Right:** Projection on the first two principal components (plane represented on the left).

Principal Component Analysis - Pre-processing the variables

- Variables should:
  - be centered, to have mean zero
  - have the same variance 1
- the results obtained depend on whether the variables have been individually scaled

Principal Component Analysis - Pre-processing the variables

- Variables should:
  - be centered, to have mean zero
  - have the same variance 1
- the results obtained depend on whether the variables have been individually scaled

```
from sklearn.decomposition import PCA
pca = PCA(n_components=10)
X_train_pca = pca.fit_transform(X_train)
```

Principal Component Analysis - Proportion of the Variance Explained

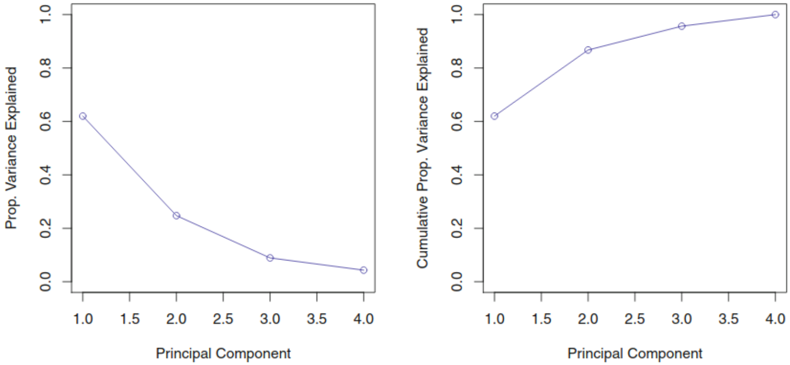
- PVE (Proportion of the Variance Explained) measures how much of the information in a given data set is lost by projecting the observations onto the first few principal components

Principal Component Analysis - Proportion of the Variance Explained

- PVE (Proportion of the Variance Explained) measures how much of the information in a given data set is lost by projecting the observations onto the first few principal components
- The PVE for the m^{th} principal component is defined as:

$$PVE_m = \frac{\text{Variance explained by the } m^{th} \text{ component}}{\text{Total variance}}$$

Principal Component Analysis - Proportion of the Variance Explained



- **Left:** proportion of variance explained by each of the four principal components
- **Right:** the cumulative proportion of variance explained by the four principal components

Principal Component Analysis -Choosing the Number of Dimensions

No criteria for deciding how many principal components (PC) are required, but some rules of thumb:

Principal Component Analysis -Choosing the Number of Dimensions

No criteria for deciding how many principal components (PC) are required, but some rules of thumb:

- Choose the smallest number of PC required to explain a sizable amount of the variation in the data

Principal Component Analysis -Choosing the Number of Dimensions

No criteria for deciding how many principal components (PC) are required, but some rules of thumb:

- Choose the smallest number of PC required to explain a sizable amount of the variation in the data
- For dimensionality reduction:
  - Explaining 95% of the variance is a good objective.

Principal Component Analysis -Choosing the Number of Dimensions

No criteria for deciding how many principal components (PC) are required, but some rules of thumb:

- Choose the smallest number of PC required to explain a sizable amount of the variation in the data
- For dimensionality reduction:
  - Explaining 95% of the variance is a good objective.
- For data visualization:
  - Focus on a small number of axes that you can interpret.
  - Do not interpret the components explaining less than 10%.

Clustering

- Clustering refers to a very broad set of techniques for finding subgroups, or clustering clusters, in a data set.

Clustering

- Clustering refers to a very broad set of techniques for finding subgroups, or clustering clusters, in a data set.
- **Goal:** Group data into subsets so that we find some structure in the data
  - The objects grouped in each subset are similar, close to one another, homogeneous

Clustering

- Clustering refers to a very broad set of techniques for finding subgroups, or clustering clusters, in a data set.
- **Goal:** Group data into subsets so that we find some structure in the data
  - The objects grouped in each subset are similar, close to one another, homogeneous
  - And different from the objects in other groups

K-means Clustering

What is K-means Clustering?

K-means Clustering

What is K-means Clustering?

- K-means clustering is a popular unsupervised machine learning algorithm used for partitioning data into a pre-specified number (k) of clusters

K-means Clustering

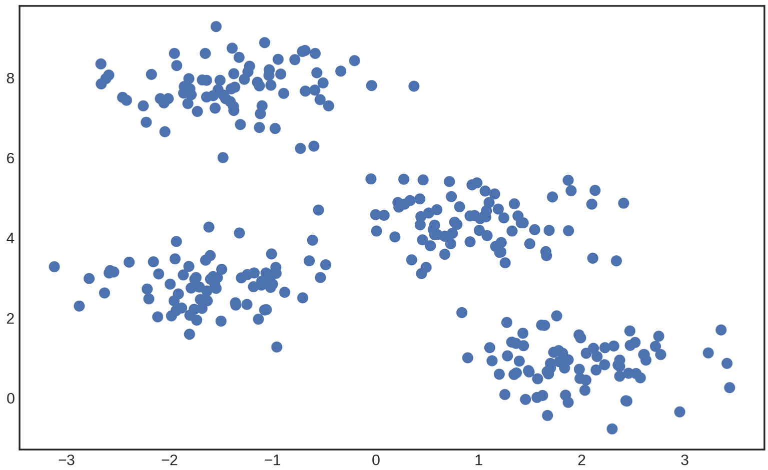
What is K-means Clustering?

- K-means clustering is a popular unsupervised machine learning algorithm used for partitioning data into a pre-specified number (k) of clusters
- The partitioning corresponds to an optimization problem that consists of:
  - Partitioning the data into k clusters of equal variance.
  - Minimizing the within-cluster sum-of-squares (**inertia**):

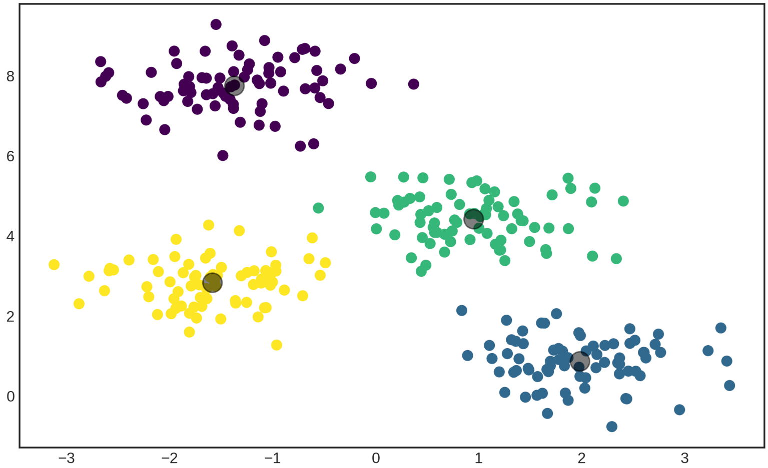
$$\sum_{i=0}^{k} \min_{\mu_j}(\|x_i - \mu_j\|^2)$$

- Each cluster is represented by the central vector or centroid μ_j.

K-means Clustering



K-means Clustering



4 clusters and their centroids

K-means Algorithm

Step 1: Randomly Assign Cluster Numbers

- Assign a number (1 to k) to each of the observations.
- This is the initial cluster assignment

K-means Algorithm

Step 1: Randomly Assign Cluster Numbers

- Assign a number (1 to k) to each of the observations.
- This is the initial cluster assignment

Step 2: Iterate Until Cluster Assignments Stop Changing

1. For each of the k clusters:
   - Compute the cluster centroid.
   - The k^{th} cluster centroid is the vector of the p feature means for the observations in the k^{th} cluster

K-means Algorithm

Step 1: Randomly Assign Cluster Numbers

- Assign a number (1 to k) to each of the observations.
- This is the initial cluster assignment

Step 2: Iterate Until Cluster Assignments Stop Changing

1. For each of the k clusters:
    - Compute the cluster centroid.
    - The k^{th} cluster centroid is the vector of the p feature means for the observations in the k^{th} cluster
2. Assign each observation to the cluster whose centroid is closest, where closest is defined using Euclidean distance.

K-means Algorithm

Step 1: Randomly Assign Cluster Numbers

- Assign a number (1 to k) to each of the observations.
- This is the initial cluster assignment

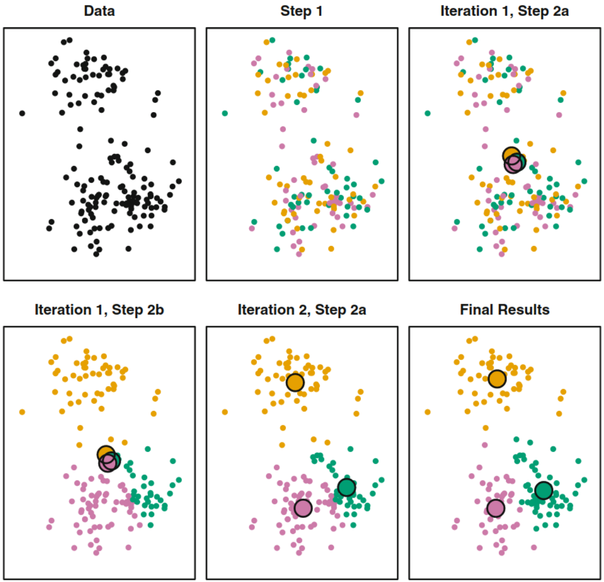
Step 2: Iterate Until Cluster Assignments Stop Changing

1. For each of the k clusters:
   - Compute the cluster centroid.
   - The k^{th} cluster centroid is the vector of the p feature means for the observations in the k^{th} cluster
2. Assign each observation to the cluster whose centroid is closest, where closest is defined using Euclidean distance.

Objective: Minimize Inertia

- The algorithm aims to choose centroids that minimize the inertia (within-cluster sum-of-squares criterion).

K-means Clustering

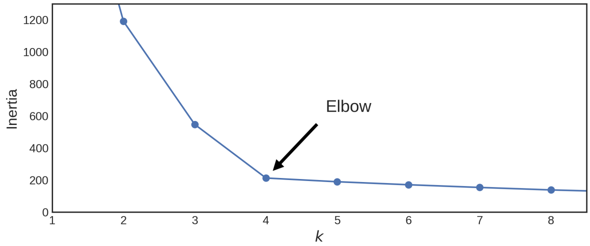


Finding the Optimal Number of Clusters

- Most of the time, the number of clusters does not stand out from looking at the data
- Inertia decreases with the number of clusters (e.g., each observation as a cluster)
- **Rule of Thumb: Choose the number of clusters at the "elbow"**

Finding the Optimal Number of Clusters

- Most of the time, the number of clusters does not stand out from looking at the data
- Inertia decreases with the number of clusters (e.g., each observation as a cluster)
- **Rule of Thumb: Choose the number of clusters at the "elbow"**



- The elbow is the point of inflection in the curve of inertia versus the number of clusters

Finding the Optimal Number of Clusters

- The **silhouette score** measures how well each point fits into its assigned cluster and how far it is from other clusters

- The silhouette score for a data point i is defined as:

$$\frac{b_i - a_i}{\max(a_i, b_i)}$$

  where a_i is the mean distance between i and other points in the same cluster, and b_i is the mean distance between i and other points in the second closest cluster

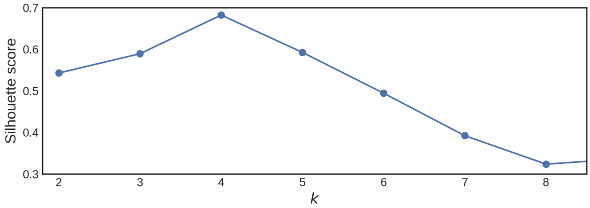
- The optimal number of clusters can be selected based on the highest silhouette score.

Finding the Optimal Number of Clusters

- The **silhouette score** measures how well each point fits into its assigned cluster and how far it is from other clusters
- The silhouette score for a data point i is defined as:

$$\frac{b_i - a_i}{\max(a_i, b_i)}$$

  where a_i is the mean distance between i and other points in the same cluster, and b_i is the mean distance between i and other points in the second closest cluster
- The optimal number of clusters can be selected based on the highest silhouette score.



Outline

Unsupervised Learning

- Dimensionality Reduction
- Clustering

Text as Data

Text is high-dimensional

- Sample of documents, each n_L words long, drawn from vocabulary of n_V words.

Text is high-dimensional

- Sample of documents, each n_L words long, drawn from vocabulary of n_V words.
- The unique representation of each document has dimensions $n_V^{n_L}$
- Example: 30-word Twitter messages using only the 1000 most common English words:

$$\text{Dimensionality} = 1000^{30} = 10^{90}$$

Methods Overview

- This Week
  - Tokenization
  - Dictionary-Based Methods
  - Measuring Document Distance
  - Supervised Learning with Text
  - Topic Models
  - Embeddings
  - Linguistic Parsing

Methods Overview

- This Week
  - Tokenization
  - Dictionary-Based Methods
  - Measuring Document Distance
  - Supervised Learning with Text
  - Topic Models
  - Embeddings
  - Linguistic Parsing
- Next week
  - Large Language Models (LLMs)
  - Generative AI

Tokenization

Without LLMs, how do we go from text to an input that we can use in analyses?

Tokenization

Without LLMs, how do we go from text to an input that we can use in analyses?

- We split the documents into *tokens* through pre-processing steps

Tokenization

Without LLMs, how do we go from text to an input that we can use in analyses?

- We split the documents into *tokens* through pre-processing steps
- Input:
  - A set of documents D

Tokenization

Without LLMs, how do we go from text to an input that we can use in analyses?

- We split the documents into *tokens* through pre-processing steps
- Input:
  - A set of documents D
- Pre-processing:
  - Removing page numbers, capitalization, punctuation, etc.
  - This can be done *manually* (e.g., using `regex`) or automated with `spacy` or `nltk`

Tokenization

Without LLMs, how do we go from text to an input that we can use in analyses?

- We split the documents into *tokens* through pre-processing steps
- Input:
  - A set of documents D
- Pre-processing:
  - Removing page numbers, capitalization, punctuation, etc.
  - This can be done *manually* (e.g., using `regex`) or automated with `spacy` or `nltk`
- Output
  - Tokens: A sequence, w with a list of tokens (words) in document i to use in natural language processing
  - Document-term matrix X: frequencies of words/phrases in each document

Segmenting paragraphs/sentences

- Many NLP tasks require analysis at the sentence level rather than whole documents
  - `spacy` does a good (but not perfect) job of splitting sentences, accounting for periods in abbreviations, etc.

Segmenting paragraphs/sentences

- Many NLP tasks require analysis at the sentence level rather than whole documents
  - `spacy` does a good (but not perfect) job of splitting sentences, accounting for periods in abbreviations, etc.
- There isn't a grammar-based paragraph tokenizer
  - Most corpora have new paragraphs annotated
  - or use line breaks

Pre-processing

- A key part of text analysis is deciding what data to remove
  - Uninformative data introduces noise, reduces precision, and increases computational load.

Pre-processing

- A key part of text analysis is deciding what data to remove
  - Uninformative data introduces noise, reduces precision, and increases computational load.
- For example:
  - Capitalization
  - Punctuation
  - Stopwords
  - Word endings

Tokens

- The most basic units of representation in a text are
  - **Characters:** sequence of letters {`h,e,l,l,o`}
  - **Words:** separated by whitespace {`hello, world`}
  - **N-grams:** phrases treated as single tokens: `Princeton University` $\rightarrow$ `princeton_university`

Bag-of-words representation

Say we want to convert a corpus D to a matrix X

- In the “bag-of-words” representation, each row of X is just a frequency distribution over words in the documents corresponding to that row
- More generally, “bag-of-terms" representation refers to counts over any informative features, e.g., n-grams, syntax features, etc..

Bag-of-words representation

Say we want to convert a corpus D to a matrix X

- In the "bag-of-words" representation, each row of X is just a frequency distribution over words in the documents corresponding to that row
- More generally, "bag-of-terms" representation refers to counts over any informative features, e.g., n-grams, syntax features, etc..

Suppose we have the sentences: "I love animals", "I love pizza", "I love pizza and animals"

Bag-of-words representation

Say we want to convert a corpus D to a matrix X

- In the "bag-of-words" representation, each row of X is just a frequency distribution over words in the documents corresponding to that row
- More generally, "bag-of-terms" representation refers to counts over any informative features, e.g., n-grams, syntax features, etc..

Suppose we have the sentences: "I love animals", "I love pizza", "I love pizza and animals"

	and	**animals**	**I**	**love**	**pizza**
I love animals	0	1	1	1	0
I love pizza	0	0	1	1	1
I love pizza and animals	1	1	1	1	1

Overview of Dictionary-Based Methods

- Dictionary-based methods reduce dimensionality. How?

Overview of Dictionary-Based Methods

- Dictionary-based methods reduce dimensionality. How?
- Go full text or document-term matrix to counts of specific occurrences

Overview of Dictionary-Based Methods

- Dictionary-based methods reduce dimensionality. How?
- Go full text or document-term matrix to counts of specific occurrences
- Use pre-selected word lists to analyze texts.
- Corpus-specific: Counting occurrences of specific words or phrases.
  - Example: Frequency of judge saying "justice" vs. "efficiency".
  - In practice: use regular expression (`regex`) for this task

Overview of Dictionary-Based Methods

- Dictionary-based methods reduce dimensionality. How?
- Go full text or document-term matrix to counts of specific occurrences
- Use pre-selected word lists to analyze texts.
- Corpus-specific: Counting occurrences of specific words or phrases.
  - Example: Frequency of judge saying "justice" vs. "efficiency".
  - In practice: use regular expression (`regex`) for this task
- General dictionaries: WordNet, LIWC, MFD.

Measuring uncertainty in macroeconomy

Baker, Bloom, and Davis (QJE 2016)

For each newspaper on each day since 1985, submit the following query:

1. Article contains “uncertain” OR “uncertainty”, AND
2. Article contains “economic” OR “economy”, AND
3. Article contains “congress” OR “deficit” OR “federal reserve” OR “legislation” OR "regulation” OR “white house”.

Normalize the resulting article counter by total monthly articles.

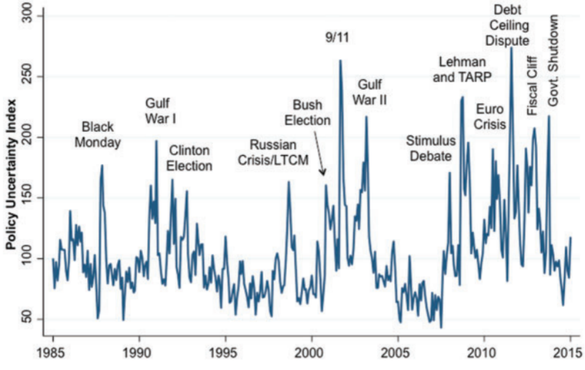
Measuring uncertainty in macroeconomy

Baker, Bloom, and Davis (QJE 2016)

For each newspaper on each day since 1985, submit the following query:

1. Article contains “uncertain” OR “uncertainty”, AND
2. Article contains “economic” OR “economy”, AND
3. Article contains “congress” OR “deficit” OR “federal reserve” OR “legislation” OR "regulation” OR “white house”.

Normalize the resulting article counter by total monthly articles.



Index reflects scaled monthly counts of articles containing 'uncertain' or 'uncertainty', 'economic' or 'economy', and one or more policy relevant terms: 'regulation', 'federal reserve', 'deficit', 'congress', 'legislation', or 'white house'. The series is normalized to mean 100 from 1985-2009 and based on queries run on 2 February, 2015 for the USA Today, Miami Herald, Chicago Tribune, Washington Post, LA Times, Boston Globe, SF Chronicle, Dallas Morning News, NY Times, and the Wall Street Journal.

FIGURE I

EPU Index for the United States

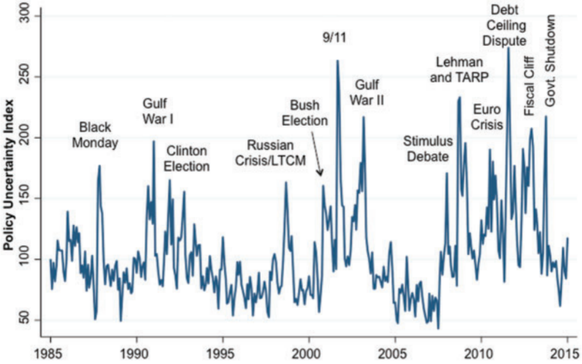
Measuring uncertainty in macroeconomy

Baker, Bloom, and Davis (QJE 2016)

For each newspaper on each day since 1985, submit the following query:

1. Article contains “uncertain” OR “uncertainty”, AND
2. Article contains “economic” OR “economy”, AND
3. Article contains “congress” OR “deficit” OR “federal reserve” OR “legislation” OR "regulation” OR “white house”.

Normalize the resulting article counter by total monthly articles.



Index reflects scaled monthly counts of articles containing ‘uncertain’ or ‘uncertainty’, ‘economic’ or ‘economy’, and one or more policy relevant terms: ‘regulation’, ‘federal reserve’, ‘deficit’, ‘congress’, ‘legislation’, or ‘white house’. The series is normalized to mean 100 from 1985-2009 and based on queries run on 2 February, 2015 for the USA Today, Miami Herald, Chicago Tribune, Washington Post, LA Times, Boston Globe, SF Chronicle, Dallas Morning News, NY Times, and the Wall Street Journal.

FIGURE I

EPU Index for the United States

But see Keith et al. (2020) for critiques: arxiv.org/abs/2010.04706

General Dictionaries

- **Function words** ("stopwords"): e.g., *for, rather, than*.
  - Can be used to get at non-topical dimensions, identify authors.
- **LIWC (Linguistic Inquiry and Word Counts)**:
  - 70+ lists of category-relevant words, e.g., emotion, cognition, work, family, positive, negative
- **Mohammad and Turney (2011)**:
  - 10,000 words on 4 emotional dimensions: joy-sadness, anger-fear, trust-disgust, anticipation-surprise
- **Warriner et al. (2013)**:
  - 14,000 words on three emotional dimensions: valence, arousal, dominance

Document-Term Matrix

- Document-term matrix X:
  - Rows (d): represent documents
  - Columns (w): represent words (or tern more generally, e.g., n-grams)

Document-Term Matrix

- Document-term matrix X:
  - Rows (d): represent documents
  - Columns (w): represent words (or tern more generally, e.g., n-grams)
- Each document/row $\mathrm{X}_{[d,:]}$ is a distribution over terms
  - these vectors have a spatial interpretation $\rightarrow$ geometric distances between document vectors reflect semantic distances between documents in terms of shared terms.

Document-Term Matrix

- Document-term matrix X:
  - Rows (d): represent documents
  - Columns (w): represent words (or tern more generally, e.g., n-grams)
- Each document/row $\mathrm{X}_{[d,:]}$ is a distribution over terms
  - these vectors have a spatial interpretation $\rightarrow$ geometric distances between document vectors reflect semantic distances between documents in terms of shared terms.
- Each word/column $\mathrm{X}_{[:,w]}$ is a distribution over documents
  - these vectors also have a spatial interpretation! Geometric distances between word vectors reflect semantic distances between words in terms of showing up in the same documents.

Cosine Similarity

- Each document is a vector x_d of e.g., term counts of TF-IDF frequencies Documents represented as vectors (x_d).
- Can measure similarity between documents i and j by the cosine of the angle between x_i and x_j:

$$\text{cos\_sim}(x_i, x_j) = \frac{x_i \cdot x_j}{||x_i||\,||x_j||}$$

  - With perfectly collinear documents (that is, $x_i == \alpha x_i, \alpha > 0$) $\cos(0) = 1$
  - For orthogonal documents (no words in common): $\cos(\pi/2) = 0$

Machine Learning with Text Data

- We have a corpus D of $n_D \geq 1$ documents d_i
- Each document i has an associated outcome or label y_i with dimension $n_y \geq 1$

Machine Learning with Text Data

- We have a corpus D of $n_D \geq 1$ documents d_i
- Each document i has an associated outcome or label y_i with dimension $n_y \geq 1$
- **Goal**: Learn a function $\hat{y}(d_i)$ from labeled data to classify/predict the unlabeled data.

First Problem

- Each document is a sequence of symbols d_i, while (standard) ML algorithms work on numbers.

First Problem

- Each document is a sequence of symbols d_i, while (standard) ML algorithms work on numbers.
- Solution: extract informative numerical features from text, such as:
  - Style features
  - Counts over dictionary patterns
  - Tokens
  - N-grams

First Problem

- Each document is a sequence of symbols d_i, while (standard) ML algorithms work on numbers.
- Solution: extract informative numerical features from text, such as:
  - Style features
  - Counts over dictionary patterns
  - Tokens
  - N-grams
- Documents can thus be **featurized** – represented as a matrix of vectors x with $n_x 1$ features

Unsupervised ML with Text: Topic Models

- Core methods for topic models were developed in computer science and statistics
  - Summarize unstructured text
  - Use words within document to infer subject
  - Useful for dimension reduction

Unsupervised ML with Text: Topic Models

- Core methods for topic models were developed in computer science and statistics
  - Summarize unstructured text
  - Use words within document to infer subject
  - Useful for dimension reduction
- Social scientists use topics as a form of measurement
  - How observed covariates drive trends in language
  - Tell a story not just about what but how and why
  - **Topic models are more interpretable** than other dimension reduction methods, such as PCA.

Standard Topic Model (LDA)

- **Latent Dirichlet Allocation (LDA)**:
  - Each topic is a distribution over words.
  - Each document is a distribution over topics.

Standard Topic Model (LDA)

- **Latent Dirichlet Allocation (LDA)**:
  - Each topic is a distribution over words.
  - Each document is a distribution over topics.
- Input: $N \times M$ document-term count matrix X
- Assume there are K topics (tunable hyperparameter)

Standard Topic Model (LDA)

- **Latent Dirichlet Allocation (LDA)**:
  - Each topic is a distribution over words.
  - Each document is a distribution over topics.
- Input: $N \times M$ document-term count matrix X
- Assume there are K topics (tunable hyperparameter)
- Like PCA, LDA works by factorizing X into:
  - an $N \times K$ document-topic matrix
  - an $K \times M$ topic-term matrix

Standard Topic Model (LDA)

- **Latent Dirichlet Allocation (LDA)**:
  - Each topic is a distribution over words.
  - Each document is a distribution over topics.
- Input: $N \times M$ document-term count matrix X
- Assume there are K topics (tunable hyperparameter)
- Like PCA, LDA works by factorizing X into:
  - an $N \times K$ document-topic matrix
  - an $K \times M$ topic-term matrix
- LDA discovers topics based upon co-occurrence of individual words (though labeling is up to the user)

Using an LDA Model

Once trained, can easily get topic proportions for a corpus.

- for any document – doesn't have to be in training corpus
- "Main topic" is the highest probability topic
- Representative documents: highest share in a topic.
- Documents with the highest share in a topic work as representative documents for the topic

Using an LDA Model

Once trained, can easily get topic proportions for a corpus.

- for any document – doesn't have to be in training corpus
- "Main topic" is the highest probability topic
- Representative documents: highest share in a topic.
- Documents with the highest share in a topic work as representative documents for the topic

Can then use the topic proportions as variables in a social science analysis.

Example

CEO Behavior and Firm Performance - O. Bandiera, A. Prat, S. Hansen, R. Sadun 2020

- They record diaries of 1,114 CEOs of manufacturing firms in six countries
- Use LDA to find the combination of features that best differentiate among CEOs
- They identify two CEO types
    - Manager: more time spent with employees
    - Leader: more time spent with C-suite executives
- Leader CEOs are more likely to lead more productive and profitable firms.

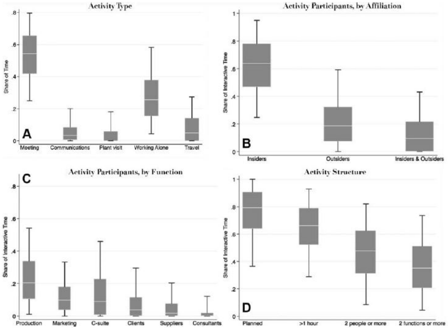


TABLE 2

Most Important Behavioral Distinctions in CEO Time-Use Data

Feature	Times Less/More Likely
Less likely in behavior 1:	
Plant visits	.11
Just outsiders	.58
Production	.46
Suppliers	.32
More likely in behavior 1:	
Communications	1.90
Outsiders and insiders	1.90
C-suite	33.90
Multifunction	1.49

Word Embeddings

- **Previously:** focus on global document counts or predict an outcome

Word Embeddings

- **Previously:** focus on global document counts or predict an outcome
- **Now**: represent the meaning of words by the neighboring words – their **local contexts**.

Word Embeddings

- **Previously:** focus on global document counts or predict an outcome
- **Now**: represent the meaning of words by the neighboring words – their **local contexts**.
  - → rather than predicting some metadata, they predict the co-occurence of neighboring words.

Word Embeddings

- **Previously:** focus on global document counts or predict an outcome
- **Now**: represent the meaning of words by the neighboring words – their **local contexts**.
  $\rightarrow$ rather than predicting some metadata, they predict the co-occurence of neighboring words.
- From high-dimensional sparse representations to low-dimensional dense representations

Word Embeddings

- **Previously:** focus on global document counts or predict an outcome
- **Now**: represent the meaning of words by the neighboring words – their **local contexts**.
  $\rightarrow$ rather than predicting some metadata, they predict the co-occurence of neighboring words.
- From high-dimensional sparse representations to low-dimensional dense representations
- "Word embeddings" often refer to Word2Vec or GloVe – these are particular (popular) models for producing word embeddings.

Words and Contexts

A long line of NLP research aims to capture the distributional properties of words using a **word-context matrix M**:

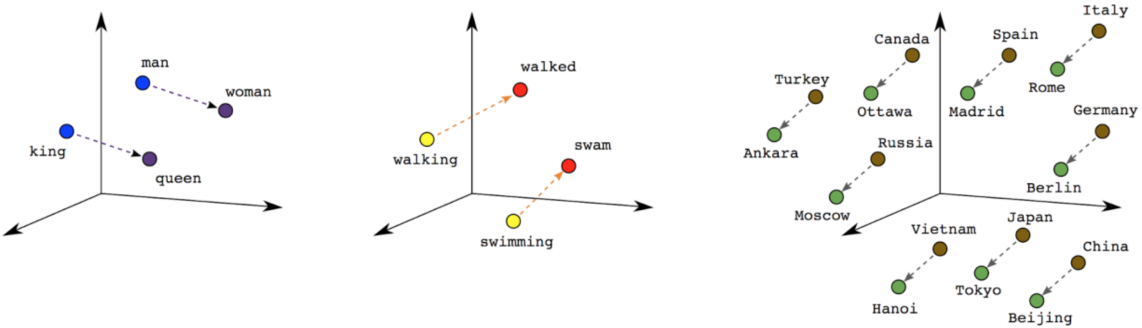
- each row w represents a **word** (e.g. “income”), each column c represents a linguistic **context** in which words can occur (e.g. “... pay corporate income __ to the relevant ...”).
  - A matrix entry $\boldsymbol{M}_{[w,c]}$ quantifies the strength of association between a word and a context in a large corpus.
  - Popular embeddings (word2vec and glove) generally use 5- or 10-word windows as the context.

Words and Contexts

A long line of NLP research aims to capture the distributional properties of words using a **word-context matrix** M:

- each row w represents a **word** (e.g. "income"), each column c represents a linguistic **context** in which words can occur (e.g. "... pay corporate income __ to the relevant ...").
  - A matrix entry $\boldsymbol{M}_{[w,c]}$ quantifies the strength of association between a word and a context in a large corpus.
  - Popular embeddings (word2vec and glove) generally use 5- or 10-word windows as the context.
- each word (row) $\boldsymbol{M}_{[w,:]}$ gives a distribution over contexts.
  - different definitions of contexts and different measures of association $\rightarrow$ different types of **word vectors**.
  - these vectors have a **spatial interpretation** $\rightarrow$ geometric distances between word vectors reflect semantic distances between words.

Semantic Dimensions



Male-Female　　Verb Tense　　Country-Capital

- Once words are represented as vectors $\{v_1, v_2, ...\}$, we can use linear algebra to understand the relationships between words:

$$\text{vec}(\text{king}) - \text{vec}(\text{man}) + \text{vec}(\text{woman}) \approx \text{vec}(\text{queen})$$

- Also applicable to sentences ("sentence embeddings").

Bias in NLP

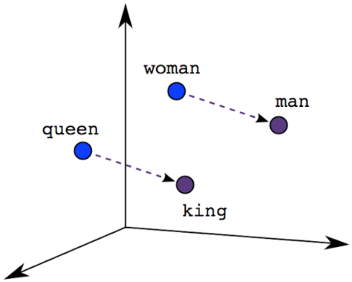
Caliskan, Bryson, and Narayanan (Science 2017)

- "We replicated a spectrum of known biases, as measured by the Implicit Association Test, using a widely used, purely statistical machine-learning model trained on a standard corpus of text from the World Wide Web. . . "

Bias in NLP

Caliskan, Bryson, and Narayanan (Science 2017)

- “We replicated a spectrum of known biases, as measured by the Implicit Association Test, using a widely used, purely statistical machine-learning model trained on a standard corpus of text from the World Wide Web. . . “



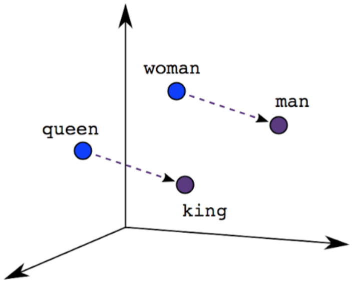
Analogies

- king : queen :: man : woman
- walked : walking :: swam : swimming

Bias in NLP

Caliskan, Bryson, and Narayanan (Science 2017)

- "We replicated a spectrum of known biases, as measured by the Implicit Association Test, using a widely used, purely statistical machine-learning model trained on a standard corpus of text from the World Wide Web. . . "



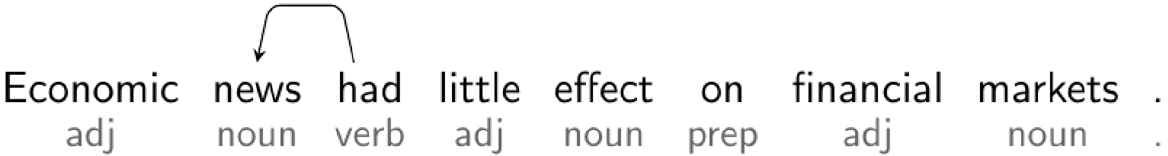
Analogies

- king : queen :: man : woman
- walked : walking :: swam : swimming
- **man : programmer :: woman : homemaker**
- **he : physician :: she : nurse**

Dependency Parsing

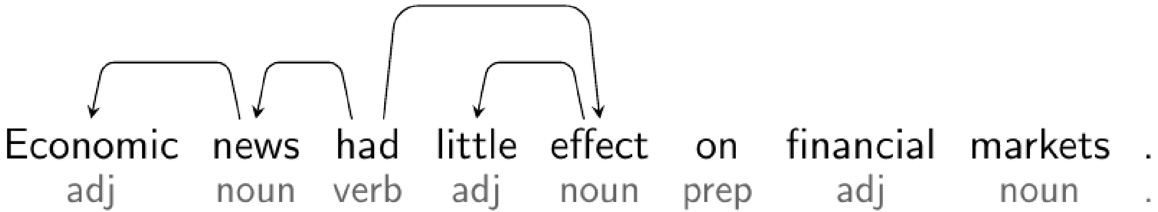
- The models we have seen so far have counted tokens, now we also incorporate grammatical concepts
- The basic idea:
  - **Syntactic structure** consists of **words**, linked by binary directed relations called **dependencies**.
  - Dependencies identify the grammatical relations between words.

Dependencies: Binary Directed Relations Between Words (Head and Dependent)



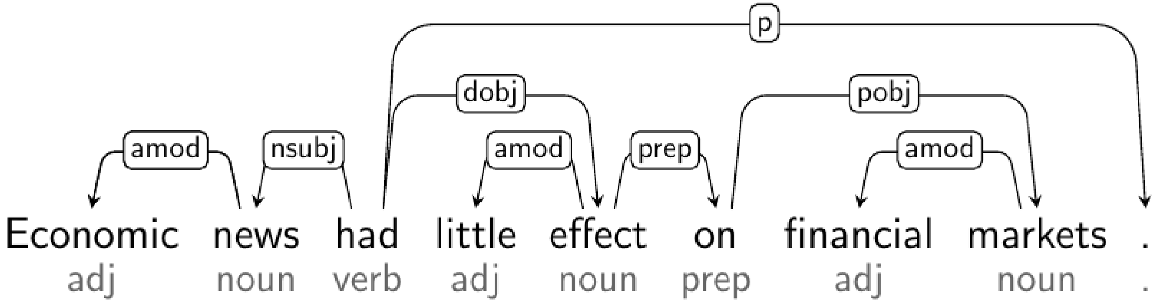
- the “root” of a sentence is the main verb (for compound sentences, the first verb).

Dependencies: Binary Directed Relations Between Words (Head and Dependent)



- directed arcs indicate dependencies: a one-way link from a “head” token to a “dependent” token.
- A word can be “head” multiple times, but “dependent” only one.

Dependencies: Binary Directed Relations Between Words (Head and Dependent)



- arc labels indicate functional relations, e.g.:
  - nsubj: verb $\rightarrow$ subject doing the verb
  - dobj: verb $\rightarrow$ object targeted by the verb
  - amod: noun $\rightarrow$ attribute of the noun
- spaCy dependency visualizer: https://explosion.ai/demos/displacy

Example

Arold et al. (2024): Do Words Matter? The Value of Collective Bargaining Agreements

- New corpus: 30,000 collective bargaining agreements from Canada, 1986-2015
- Classify clauses looking at syntactic structure of sentences

Example

Arold et al. (2024): Do Words Matter? The Value of Collective Bargaining Agreements

- New corpus: 30,000 collective bargaining agreements from Canada, 1986-2015
- Classify clauses looking at syntactic structure of sentences
- Subject categories, assigned with lexicons
    - worker, union, owner, manager

Example

Arold et al. (2024): Do Words Matter? The Value of Collective Bargaining Agreements

- New corpus: 30,000 collective bargaining agreements from Canada, 1986-2015
- Classify clauses looking at syntactic structure of sentences
- Subject categories, assigned with lexicons
  - worker, union, owner, manager
- In contracts, modal verbs impose legal requirements:
  - strict (shall, will, must) modals express necessity.
  - permissive (may, can) modals express possibility.

Example

Arold et al. (2024): Do Words Matter? The Value of Collective Bargaining Agreements

- New corpus: 30,000 collective bargaining agreements from Canada, 1986-2015
- Classify clauses looking at syntactic structure of sentences
- Subject categories, assigned with lexicons
  - worker, union, owner, manager
- In contracts, modal verbs impose legal requirements:
  - strict (shall, will, must) modals express necessity.
  - permissive (may, can) modals express possibility.
- Negation ("shall not")
- Active/passive ("shall provide" vs "shall be provided").
- Special verbs:
  - Obligation Verbs (have to, ought to, be required, be expected, be compelled, be obliged, be obligated)
  - Prohibition Verbs (be prohibited, be forbidden, be banned, be barred, be restricted, be proscribed)
  - Permission Verbs (be allowed, be permitted, be authorized)
  - Entitlement Verbs (have, receive, retain).

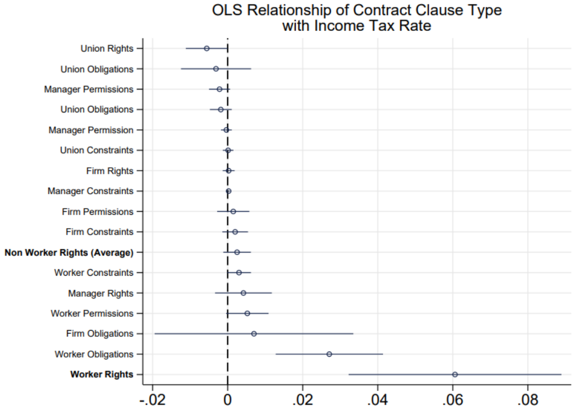
Example

- Clauses are mostly about workers (55.3%), followed by the firm/employer (30.5%)

Example

- Clauses are mostly about workers (55.3%), followed by the firm/employer (30.5%)
- Show that rights and obligations are used to compensate workers

Figure 4: Effect of Labor Income Tax Rates on Contract Terms



Note: Figure presents coefficients and 95% confidence intervals of effect of labor tax rate on contract clause types. Each coefficient is from a separate OLS regression. Outcome: Clause type share (number of clauses of type in question over the number of all clauses). Treatment: Labor tax rate, defined as logarithmized implicit personal income tax rate. Controls: Province-by-sector fixed effects and year-by-sector fixed effects. Inference: Standard errors clustered at the province-by-sector level. Data sources: Employment and Social Development Canada, Center for the Study of Living Standards.