Week 9: Embeddings

POP77032 Quantitative Text Analysis for Social Scientists

Tom Paskhalis

Overview

- Vector space arithmetics
- Embeddings in social sciences
- Choosing embeddings
- Dimension reduction

Vector Space Distances

Distance Metrics

- Once we have word embeddings, we can measure similarity/distance between words.
- Several distance metrics are commonly used:
  - **Cosine similarity**: measures the cosine of the angle between two vectors (ranges from -1 to 1).
  - **Jaccard similarity**: measures the overlap between two sets of features (ranges from 0 to 1).
  - **Euclidean distance**: straight-line distance between two points in vector space (ranges from 0 to infinity).
- For word embeddings, *cosine similarity* is most commonly used.
- As it focuses on the *direction* of vectors rather than their magnitude.

Recap: Cosine Similarity

- Recall ho we calculated between 2 documents A and B using their vector representations $\mathbf{W}_A$ and $\mathbf{W}_B$:

$$\cos(\mathbf{W}_A, \mathbf{W}_B) = \frac{\mathbf{W}_A \cdot \mathbf{W}_B}{\|\mathbf{W}_A\| \times \|\mathbf{W}_B\|}$$

  where $\|\mathbf{W}\| = \sqrt{\sum_{j=1}^{J} W_j^2}$ is the magnitude of the vector.

- The same idea can be applied to words represented as vectors.
- Values close to 0 indicate orthogonal (unrelated) vectors.
- For word embeddings, higher values indicate more similar meanings.

Finding Similar Words

- Let's find the most similar words to *taoiseach* in our Dail 33 corpus.

```r
find_similar <- function(target_word, word_vectors, top_n = 10) {
  # Calculate similarities between target word and all words
  similarities <- text2vec::sim2(
    x = matrix(word_vectors[target_word, ], nrow = 1),
    y = word_vectors,
    method = "cosine",
    norm = "l2"
  )

  # Convert to named vector and sort
  similarities <- as.vector(similarities)
  names(similarities) <- rownames(word_vectors)
  similarities <- sort(similarities, decreasing = TRUE)

  head(similarities, top_n + 1)  # +1 to exclude the word itself
}
```

```r
find_similar("taoiseach", word_vectors, top_n = 10)
```

```
taoiseach   tánaiste   minister  yesterday government      today     asking
1.0000000  0.9549996  0.8972657  0.8315870  0.8162795  0.8012301  0.7942152
    asked     stated    finally       said
0.7935601  0.7869792  0.7861874  0.7716069
```

Word Vector Arithmetic

- One remarkable property of word embeddings is that vector differences encode semantic relationships.
- This allows arithmetic operations on word vectors to discover analogies.
- Classic example from Mikolov et al. (2013):

$$\overrightarrow{king} - \overrightarrow{man} + \overrightarrow{woman} \approx \overrightarrow{queen}$$

- The idea is to subtract the “male” concept, add the “female” concept.

Example: Vector Arithmetic in Dáil

- Let's explore political analogies in the context of the Dail corpus.

$$\overrightarrow{government} - \overrightarrow{taoiseach} + \overrightarrow{tánaiste} \approx ?$$

```r
word_analogy <- function(word_vectors, positive, negative, top_n = 5) {
  # positive: words to add, negative: words to subtract
  result_vec <- matrix(0, nrow = 1, ncol = ncol(word_vectors))

  # Add positive words
  for (word in positive) {
    result_vec <- result_vec + matrix(word_vectors[word, ], nrow = 1)
  }

  # Subtract negative words
  for (word in negative) {
    result_vec <- result_vec - matrix(word_vectors[word, ], nrow = 1)
  }

  # Find most similar words to the result
  similarities <- text2vec::sim2(
    x = result_vec,
    y = word_vectors,
    method = "cosine",
    norm = "l2"
  )

  similarities <- as.vector(similarities)
```

Example: Vector Arithmetic in Dáil

$$\overrightarrow{government} - \overrightarrow{taoiseach} + \overrightarrow{tánaiste} \approx ?$$

```
word_analogy(word_vectors,
             positive = c("government", "tánaiste"),
             negative = c("taoiseach"),
             top_n = 5)
```

```
government's     minister        clear      believe        state
   0.8152813    0.8098405    0.7953141    0.7817719    0.7738497
```

$$\overrightarrow{minister} - \overrightarrow{government} + \overrightarrow{party} \approx ?$$

```
word_analogy(word_vectors,
             positive = c("minister", "party"),
             negative = c("government"),
             top_n = 5)
```

```
colleague    deputy     chair  chairman  deputies
0.8127980 0.7764321 0.7644270 0.7586249 0.7440956
```

Embeddings in Social Sciences

Embeddings in Social Sciences

- NLP researchers are largely focussed on how useful embeddings are for downstream tasks (e.g. classification, part-of-speech tagging, etc.)
- For social scientists, the interpretability of embeddings is often more important than their predictive performance.
- E.g. if the distance between words “immigrants” and “hardworking” is smaller for liberals than for conservatives, this may be an interesting finding in itself.
- However, this requires careful validation of the embedding space to ensure that it captures meaningful semantic relationships.

Example: Social Class

(Boris Ioganson, State Tretyakov Gallery)

Dimensions of Class

- *Social class* - hierarchical distinction between groups of people in social standing.
- Some aspects that social class can be based on:
  - Wealth (e.g. Georg Simmel)
  - Relationship to the means of production (e.g. Karl Marx)
  - Education
  - Gender
  - Race
  - Consumption patterns

Kozlowski, Taddy & Evans (2019)

- Use Google Ngram corpus of 5-grams divided by decade between 1900 and 1999,
- Derive cultural dimensions of word embeddings using *word2vec* skipgram architecture.
- Validate on similar data for 2000-2012 along 3 sociological axes:
  - affluence/class
  - gender
  - race
- Validation relies on MTurkers rating words along the 0-100 scale on these dimensions.
- For historical validation they construct 20 cultural dimensions derived from *An Atlas of Semantic Profiles for 360 Words* by Jenkins, Russell & Suci (1958).

Scale Construction

- Kozlowski, Taddy & Evans (2019)] argue that the reason that:

$$\overrightarrow{king} - \overrightarrow{man} + \overrightarrow{woman} \approx \overrightarrow{queen}$$

- Is because ($\overrightarrow{woman} - \overrightarrow{man}$) closely corresponds to 'gender dimension'.
- E.g. in the case of class they find that:

$$\overrightarrow{hockey} + \overrightarrow{affluence} - \overrightarrow{poverty} \approx \overrightarrow{lacrosse}$$

- Thus, by taking the mean of all antonym pairs across a given dimension:

$$\frac{\sum_{p}^{|P|} \overrightarrow{p_1} - \overrightarrow{p_2}}{|P|}$$

one can construct a scale that would position words along that dimension.

Mapping Words to Class

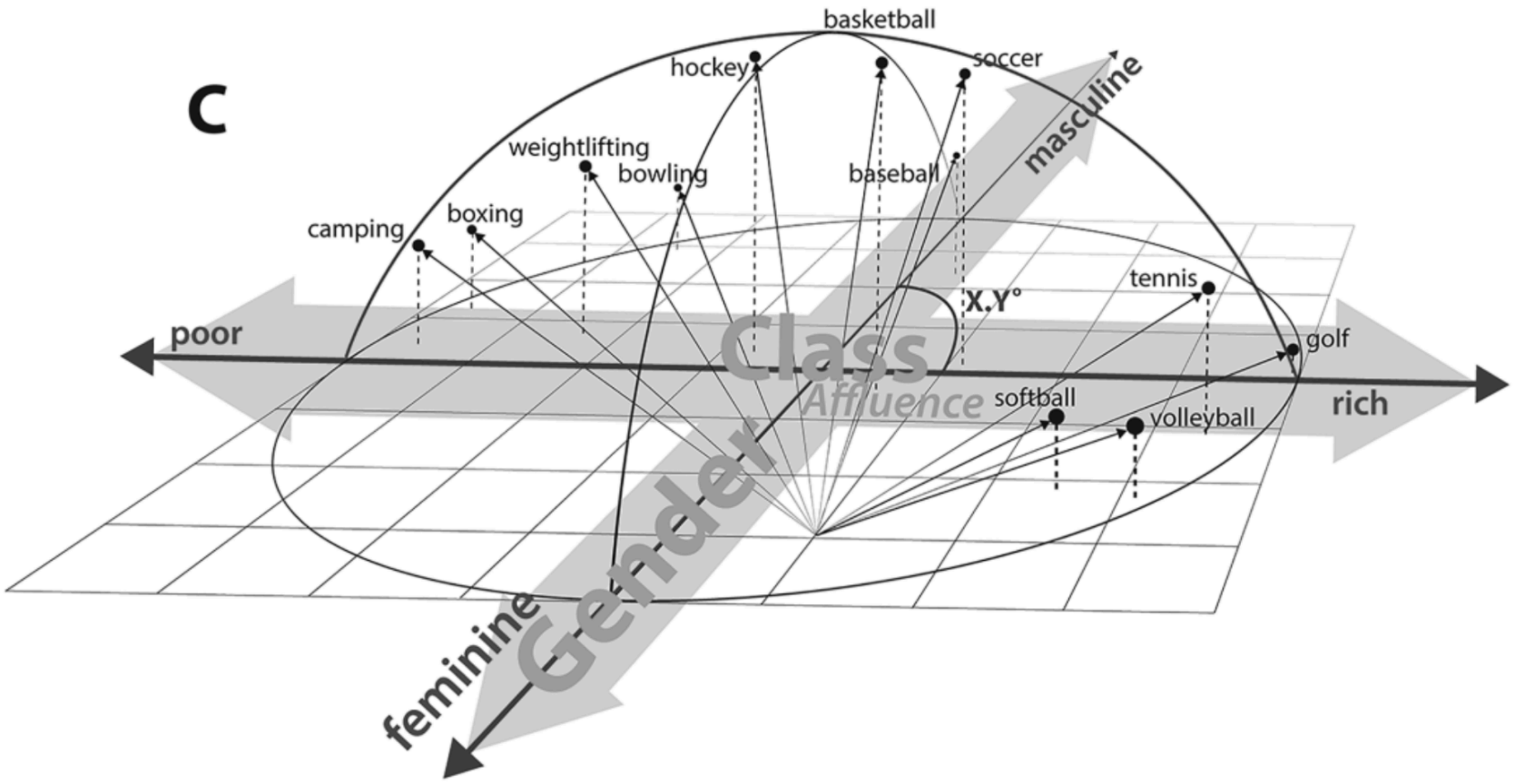


Figure 2. Conceptual Diagram of (A) the Construction of a Cultural Dimension; (B) the Projection of Words onto That Dimension; and (C) the Simultaneous Projection of Words onto Multiple Dimensions

(Kozlowski, Taddy & Evans, 2019)

Social Class over Time

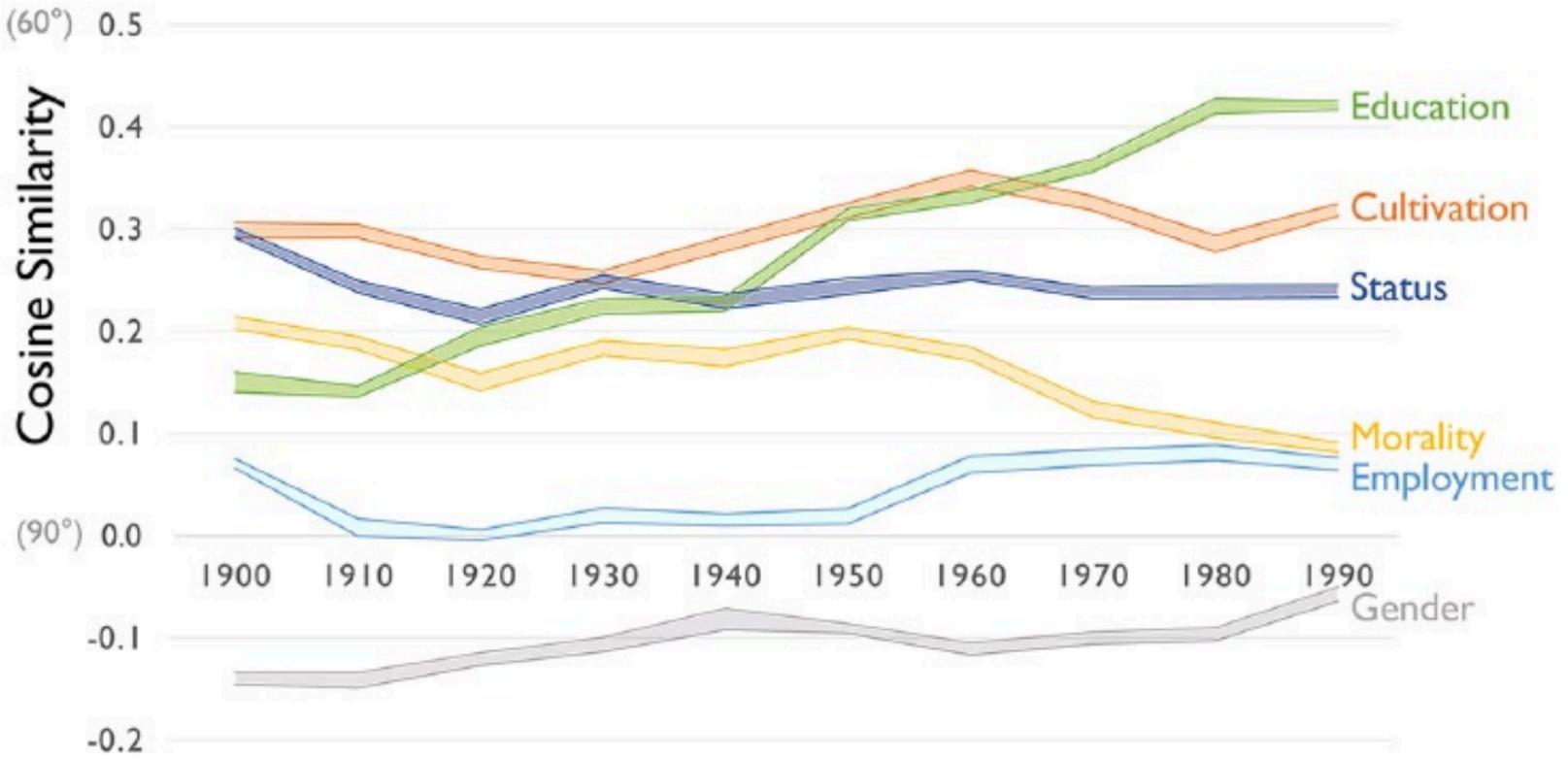


Figure 5. Cosine Similarity between the Affluence Dimension and Six Other Cultural Dimensions of Class by Decade; 1900 to 1999 Google Ngrams Corpus
Note: Bands represent 90 percent bootstrapped confidence intervals produced by subsampling.

(Kozlowski, Taddy & Evans, 2019)

Choosing Embeddings

- Applying word embeddings comes with a number of analytical choices.
- These choices come on top of previously discussed preprocessing decisions.
- One must choose between using ‘off-the-shelf’ embeddings or training their own.
- In each case these key decisions have to be made (or were made for pre-trained):
  - Corpus
  - Context window size
  - Dimensionality of the embedding
  - Embedding algorithm

Validating Embeddings

- Similar to other text analysis tasks, validating embeddings is essential.
- E.g. Rodriguez & Spirling (2022) propose a Turing-test style assessment:
  1. Ask humans (e.g. MTurkers) to produce closest analogies to a given prompt word.
  2. Extract machine-generated nearest neighbors from a chosen embedding for the same prompt word.
  3. Ask a different set of humans to choose a closer match between human- and machine-derived analogies.
  4. Compute probability that machine-generated analogies are picked over human-derived ones.
- A similar approach (step 2–4) could be used to compare across different computer-based embeddings.

Comparing Embeddings

- Applying this approach to 102-111 US Congresses (~1.4M documents):

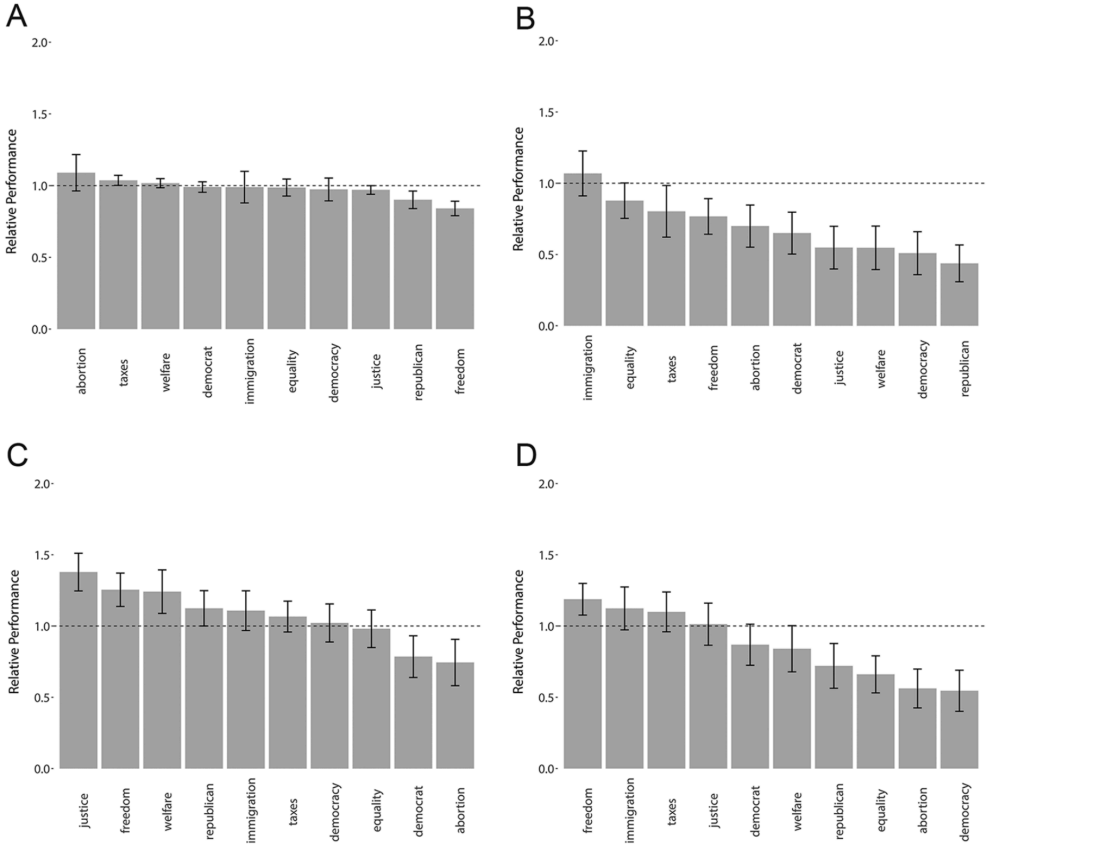


Figure 2. Human preferences: Turing assessment. *A*, Candidate: local 48-300; baseline: local 6-300. *B*, Candidate: local 6-300; baseline: human. *C*, Candidate: GloVe; baseline: local 6-300. *D*, Candidate: GloVe; baseline: human.

(Rodriguez & Spirling, 2022)

Dimension Reduction

Why Dimension Reduction?

- Word embeddings are typically high-dimensional (50-300 dimensions).
- But visualising and interpreting relationships in such high-dimensional spaces is difficult.
- Which is of particular concern for social scientists interested in interpretability.
- **Dimension reduction** transforms high-dimensional data into lower dimensions while preserving important structure.
- Goal: retain as much meaningful information as possible with fewer dimensions.

Principal Component Analysis

- Principal Component Analysis (PCA) identifies the directions (principal components) along which the data varies the most.
- **First principal component (PC1)**: Direction of maximum variance in the data.
- **Second principal component (PC2)**: Direction of maximum remaining variance, orthogonal to PC1.
- And so on for subsequent components.
- Each principal component is a linear combination of original features.

Correlated Data

- Let's apply PCA to a simple synthetic dataset with 2 correlated variables:

```r
set.seed(123)
n <- 100
x1 <- rnorm(n, mean = 5, sd = 2)
x2 <- 0.8 * x1 + rnorm(n, mean = 2, sd = 1)
pca_data <- data.frame(x1 = x1, x2 = x2)
```

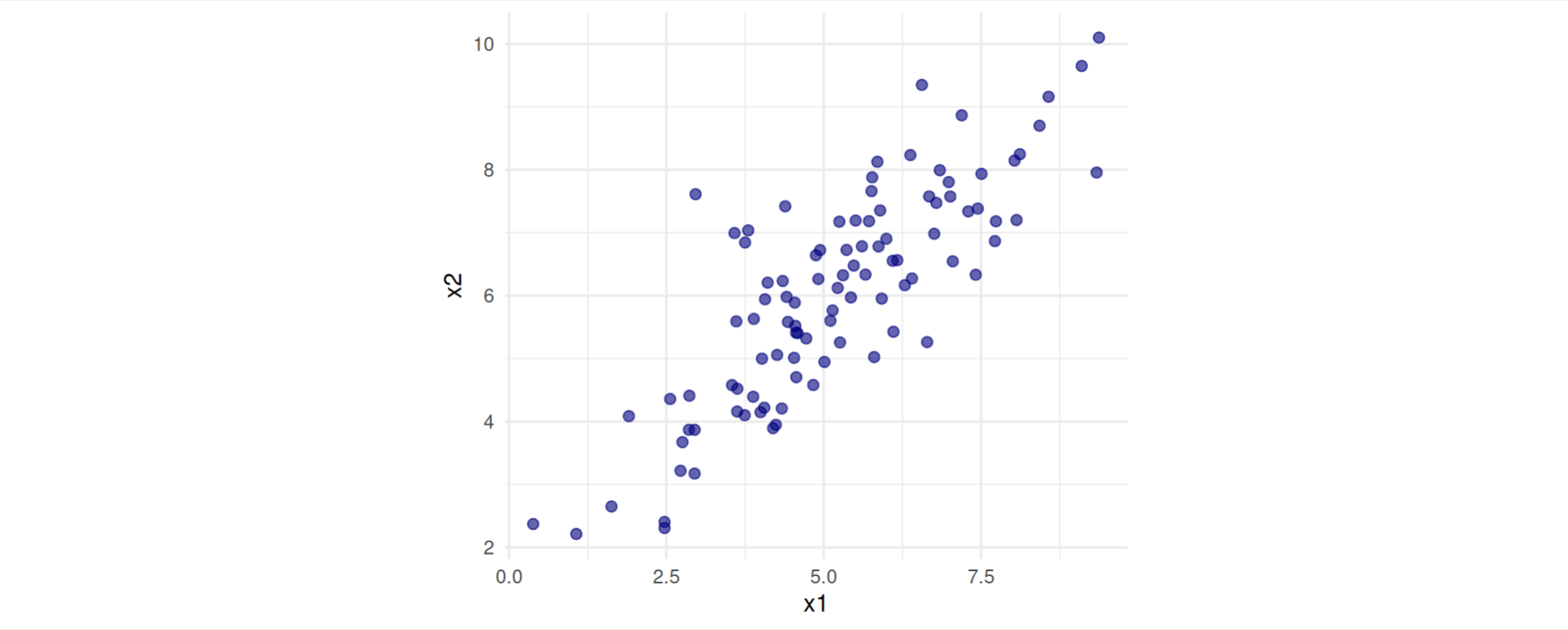
```r
cor.test(pca_data$x1, pca_data$x2, method = "pearson")
```

```
    Pearson's product-moment correlation

data:  pca_data$x1 and pca_data$x2
t = 14.479, df = 98, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.7508279 0.8793420
sample estimates:
      cor
0.8255038
```

Correlated Data

Plot | Code



- Notice the positive correlation: as x_1 increases, so does x_2.

Applying PCA

- To apply PCA in R we can use built-in `prcomp()` function.

```r
pca <- prcomp(pca_data, center = TRUE, scale = TRUE)
summary(pca)
```

```
Importance of components:
                          PC1     PC2
Standard deviation     1.3511 0.41773
Proportion of Variance 0.9127 0.08725
Cumulative Proportion  0.9127 1.00000
```

- We can then extract the loading vectors (eigenvectors), that show how each of the original features contributes to each component:

```r
pca$rotation
```

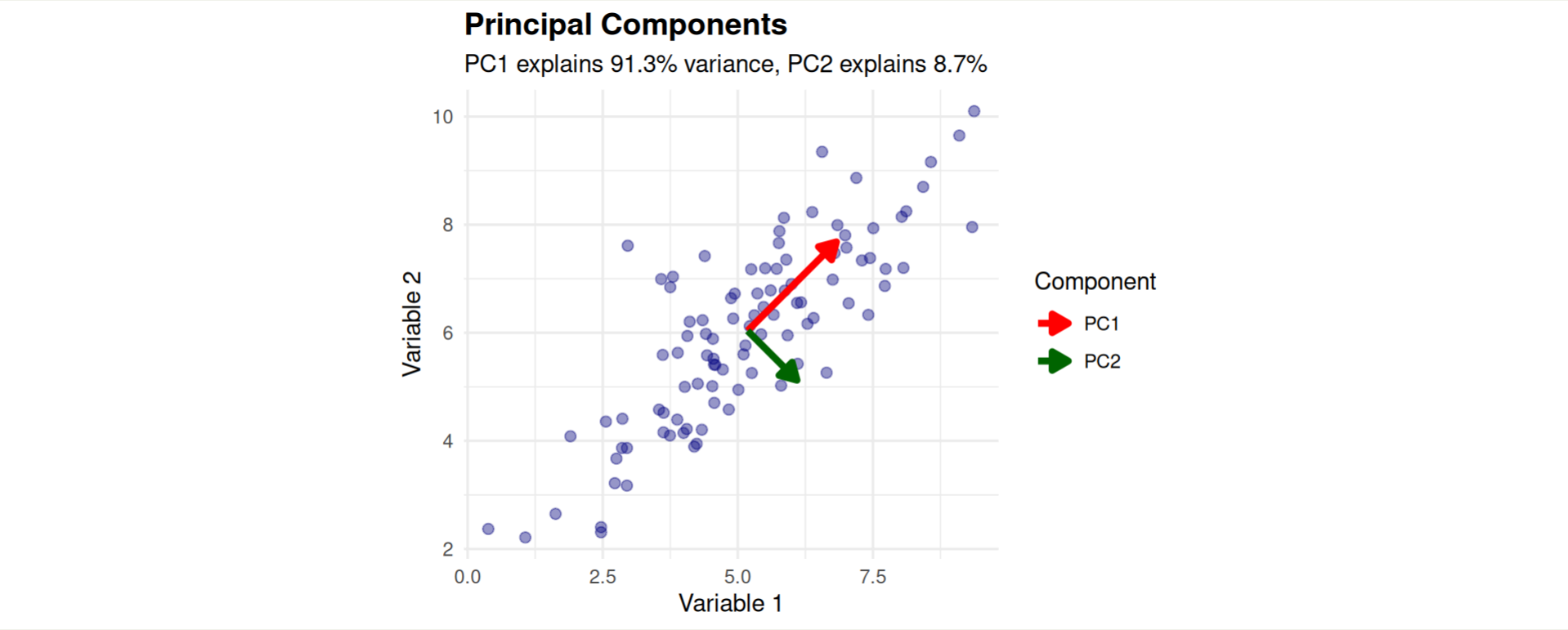
```
         PC1        PC2
x1 0.7071068  0.7071068
x2 0.7071068 -0.7071068
```

Interpreting PCA Results

- **Eigenvalues**: Indicate the amount of variance explained by each component.
- Proportion of variance explained: $\frac{\lambda_i}{\sum_{j=1}^{p} \lambda_j}$
- Typically report cumulative variance explained by first few components.
- **Loadings**: Weights showing how original features contribute to each PC.
- Principal components are **orthogonal** (uncorrelated with each other).
- PCA assumes **linear relationships** between variables.

PCA Results

Plot | Code



- **PC1** (red) captures the direction of maximum variance.
- **PC2** (green) captures the remaining variance, orthogonal to PC1.

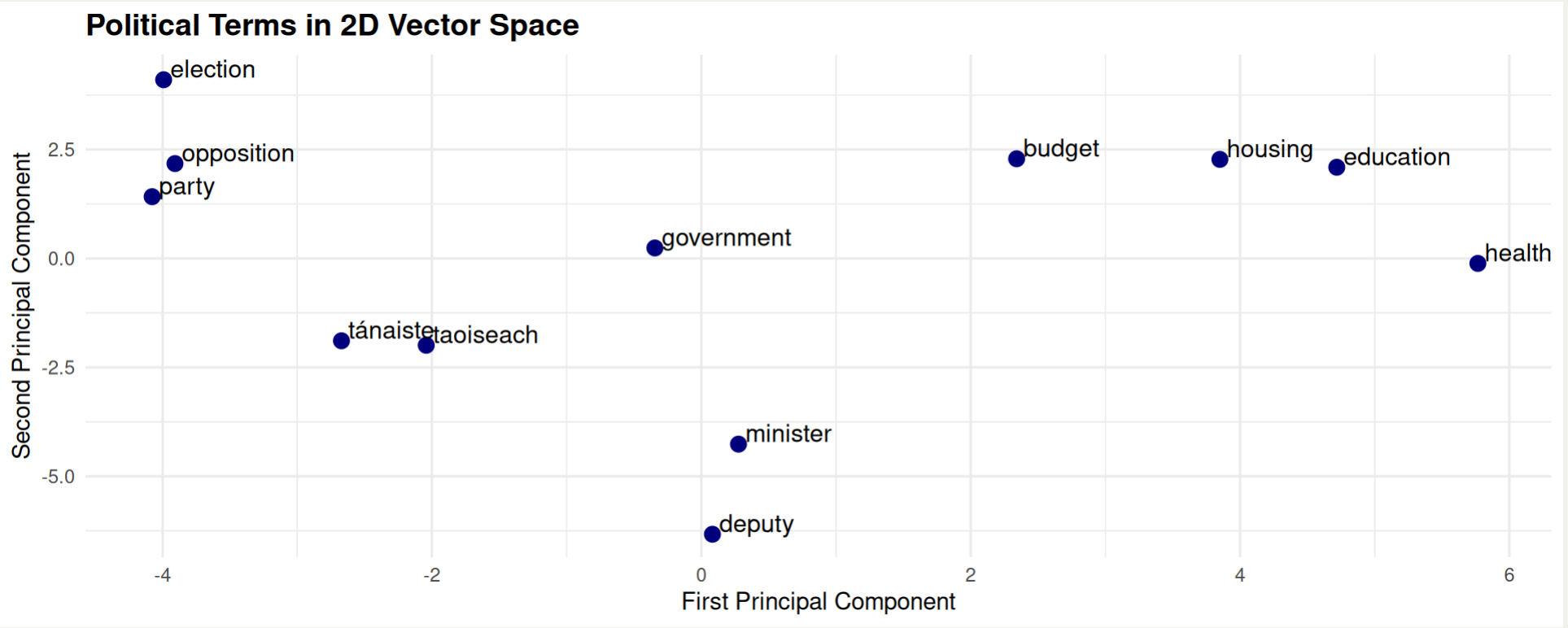
PCA for Word Embeddings

- For word embeddings:
  - Each word is an observation (n = vocabulary size)
  - Each embedding dimension is a feature (p = embedding dimensionality, e.g., 50-300)
- Reducing to 2D allows us to visualize semantic relationships.
- Words close together in the original high-dimensional space should remain close in 2D.
- **Limitation**: PCA may not capture all nuances of semantic similarity.

Visualizing Word Distances

- We can visualise word embeddings by reducing dimensionality to 2D using PCA.

Plot | Code



Next

- Tutorial: Validating embeddings
- Next Week: Neural networks
- Assignment 3: Due 15:59 on Wednesday, 1st April (submission on Blackboard)