Text Data in Business and Economics

Basel University - Autumn 2023

4. Document Distance

Outline

Document Distance

Clustering

Document Distance/Proximity

- In economics, we often want to compare documents to one another.
- Example, how close is a political speech to the party leader?

Document Distance/Proximity

- In economics, we often want to compare documents to one another.
- Example, how close is a political speech to the party leader?
- Today, we will focus on methods designed to measure document distance/proximity.
- But almost everything we do in this class can be framed as measuring document distance in some way.

Text Re-Use

- Text Re-Use algorithms (like "Smith-Waterman" or BLAST) measure similarity by finding and counting shared sequences in two texts above some minimum length, e.g. 5 words.
  - useful for plagiarism detection, for example.
- precise but slow
  - shortcut: look at proportion of shared (hashed) 5-grams across texts

Document-Term Matrix

The **document-term matrix $\boldsymbol{X}$**:

- each row d represents a **document,** while each column w represents a word (or term more generally, e.g. n-grams).
  - A matrix entry $\boldsymbol{X}_{[d,w]}$ quantifies the strength of association between a document and a word, generally its count or frequency

Document-Term Matrix

The **document-term matrix $\boldsymbol{X}$**:

- each row d represents a **document,** while each column w represents a word (or term more generally, e.g. n-grams).
  - A matrix entry $\boldsymbol{X}_{[d,w]}$ quantifies the strength of association between a document and a word, generally its count or frequency
- each document/row $\boldsymbol{X}_{[d,:]}$ 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

The **document-term matrix** $\boldsymbol{X}$:

- each row d represents a **document,** while each column w represents a word (or term more generally, e.g. n-grams).
  - A matrix entry $\boldsymbol{X}_{[d,w]}$ quantifies the strength of association between a document and a word, generally its count or frequency
- each document/row $\boldsymbol{X}_{[d,:]}$ 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 $\boldsymbol{X}_{[:,w]}$ 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, e.g. term counts or TF-IDF frequencies.
- $\rightarrow$ Each document is a non-negative vector in an n_x-space, where $n_x =$ vocabulary size.
  - that is, documents are rays, and similar documents have similar vectors.

Cosine Similarity

- Each document is a vector x_d, e.g. term counts or TF-IDF frequencies.
- $\rightarrow$ Each document is a non-negative vector in an n_x-space, where $n_x =$ vocabulary size.
  - that is, documents are rays, and similar documents have similar vectors.
- Can measure similarity between documents i and j by the cosine of the angle between x_i and x_j :
  - With perfectly collinear documents (that is, $x_i = \alpha x_j$, $\alpha > 0$), $\cos(0) = 1$
  - For orthogonal documents (no words in common), $\cos(\pi/2)=0$

Cosine Similarity

- Each document is a vector x_d, e.g. term counts or TF-IDF frequencies.
- $\rightarrow$ Each document is a non-negative vector in an n_x-space, where $n_x =$ vocabulary size.
  - that is, documents are rays, and similar documents have similar vectors.
- Can measure similarity between documents i and j by the cosine of the angle between x_i and x_j :
  - With perfectly collinear documents (that is, $x_i = \alpha x_j$, $\alpha > 0$), $\cos(0) = 1$
  - For orthogonal documents (no words in common), $\cos(\pi/2){=}0$

Cosine similarity is computable as the normalized dot product between the vectors:

$$\text{cos\_sim}(x_1, x_2) = \frac{x_1 \cdot x_2}{||x_1||||x_2||}$$

```
from sklearn.metrics.pairwise import
cosine_similarity
# between two vectors:
sim = cosine_similarity(x, y)[0,0]
# between all rows of a matrix:
sims = cosine_similarity(X)
```

Notes on Cosine Similarity

- For a corpus with n rows, the pairwise similarities give $n \times (n-1)$ similarity scores.

Notes on Cosine Similarity

- For a corpus with n rows, the pairwise similarities give $n \times (n-1)$ similarity scores.
- tf-idf down-weights terms that appear in many documents, usually gives better results.
    - tf-idf similarity is the workhorse, used for example in bm25 and elasticsearch.

Notes on Cosine Similarity

- For a corpus with n rows, the pairwise similarities give $n \times (n-1)$ similarity scores.
- tf-idf down-weights terms that appear in many documents, usually gives better results.
  - tf-idf similarity is the workhorse, used for example in bm25 and elasticsearch.

Alternative distance metrics:

- dot product and Euclidean distance are too sensitive to document length
- Jensen-Shannon Divergence
- Jaccard distance
- etc.

Burgess et al, "Legislative Influence Detectors"

- Compare bill texts across states in two-step process:
  (1) find candidates using elasticsearch (tf-idf similarity);
  (2) compare candidates using text reuse score.

Burgess et al, “Legislative Influence Detectors”

- Compare bill texts across states in two-step process:
  (1) find candidates using elasticsearch (tf-idf similarity);
  (2) compare candidates using text reuse score.

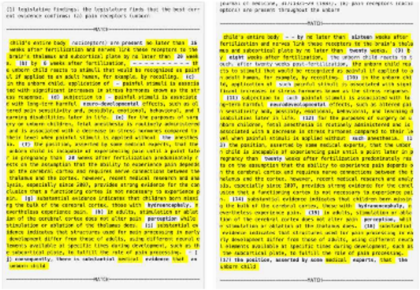


Figure 10: Match between Scott Walker's bill and a highly similar bill from Louisiana. For a detailed view, please visit http://dssg.uchicago.edu/lid/.

Burgess et al, “Legislative Influence Detectors”

- Compare bill texts across states in two-step process:
  (1) find candidates using elasticsearch (tf-idf similarity);
  (2) compare candidates using text reuse score.

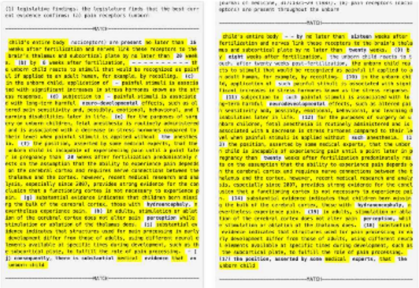


Figure 10: Match between Scott Walker's bill and a highly similar bill from Louisiana. For a detailed view, please visit http://dssg.uchicago.edu/lid/.

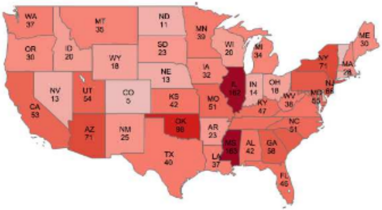


Figure 7: Introduced bills by state from ALEC model legislation

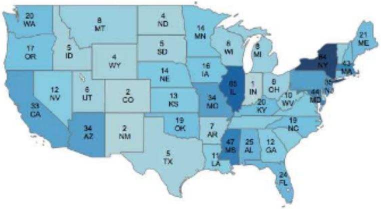


Figure 8: Introduced bills by state from ALICE model legislation

ABSTRACT

State legislatures introduce at least 45,000 bills each year. However, we lack a clear understanding of who is actually writing those bills. As legislators often lack the time and staff to draft each bill, they frequently copy text written by other states or interest groups.

However, existing approaches to detect text reuse are slow, biased, and incomplete. Journalists or researchers who want to know where a particular bill originated must perform a largely manual search. Watchdog organizations even hire armies of volunteers to monitor legislation for matches. Given the time-consuming nature of the analysis, journalists and researchers tend to limit their analysis to a subset of topics (e.g. abortion or gun control) or a few interest groups.

This paper presents the Legislative Influence Detector (LID). LID uses the Smith-Waterman local alignment algorithm to detect sequences of text that occur in model legislation and state bills. As it is computationally too expensive to run this algorithm on a large corpus of data, we use a search engine built using Elasticsearch to limit the number of comparisons. We show how LID has found 45,405 instances of bill-to-bill text reuse and 14,137 instances of model-legislation-to-bill text reuse. LID reduces the time it takes to manually find text reuse from days to seconds.

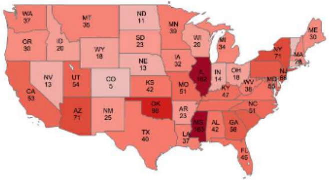


Figure 7: Introduced bills by state from ALEC model legislation

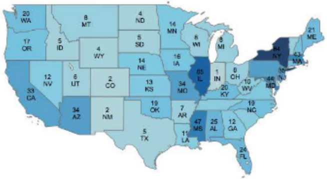


Figure 8: Introduced bills by state from ALICE model legislation

1. What is the research question?
2. Why is it important?
3. What is the problem solved?
4. What is being measured?
5. How does the measurement help answer the research question?

Outline

Document Distance

Clustering

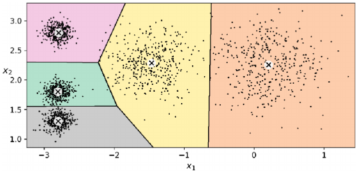
- k-**means clustering**: divide documents $\boldsymbol{x}$ into sets $S_1, \ldots, S_k$ with associated centroids $\mu_1, \ldots, \mu_k$; each doc is in set with closest centroid.

- k-**means clustering**: divide documents $\boldsymbol{x}$ into sets $S_1, ..., S_k$ with associated centroids $\mu_1, ..., \mu_k$; each doc is in set with closest centroid.
- algorithm: initialize cluster centroids randomly, then shift around to minimize sum of within-cluster squared distance (features should be standardized)

$$\arg\min_S = \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in S_i} ||\boldsymbol{x} - \mu_i||^2$$

- **k-means clustering**: divide documents $\boldsymbol{x}$ into sets $S_1, \ldots, S_k$ with associated centroids $\mu_1, \ldots, \mu_k$; each doc is in set with closest centroid.
- algorithm: initialize cluster centroids randomly, then shift around to minimize sum of within-cluster squared distance (features should be standardized)

$$\arg\min_S = \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in S_i} ||\boldsymbol{x} - \mu_i||^2$$

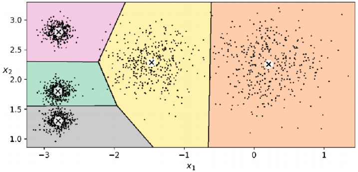


K-Means decision boundaries (Voronoi tessellation)

```
from sklearn.cluster import Kmeans
kmeans = KMeans(n_clusters=10)
kmeans.fit(X)
assigned_cluster = kmeans.labels_
```

- **k-means clustering**: divide documents $\boldsymbol{x}$ into sets $S_1, \ldots, S_k$ with associated centroids $\mu_1, \ldots, \mu_k$; each doc is in set with closest centroid.
- algorithm: initialize cluster centroids randomly, then shift around to minimize sum of within-cluster squared distance (features should be standardized)

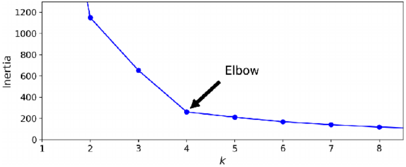
$$\arg\min_S = \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in S_i} ||\boldsymbol{x} - \mu_i||^2$$



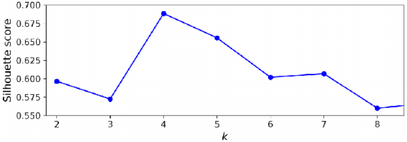
K-Means decision boundaries (Voronoi tessellation)

```
from sklearn.cluster import Kmeans
kmeans = KMeans(n_clusters=10)
kmeans.fit(X)
assigned_cluster = kmeans.labels_
```

k (number of clusters) is the only hyperparameter, can select using:



Selecting the number of clusters k using the "elbow rule"



Selecting the number of clusters k using the silhouette score

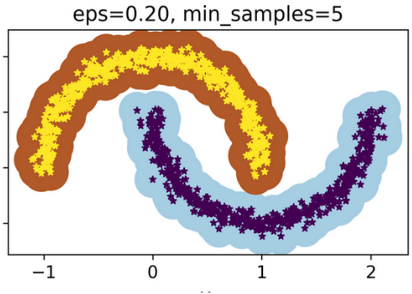
Other clustering algorithms

- "k-medoid" clustering use L1 distance rather than Euclidean distance; produces the "medoid" (median vector) for each cluster rather than "centroid" (mean vector).
  - less sensitive to outliers, and medoid can be used as representative data point.

Other clustering algorithms

- “k-medoid” clustering use L1 distance rather than Euclidean distance; produces the “medoid” (median vector) for each cluster rather than “centroid” (mean vector).
  - less sensitive to outliers, and medoid can be used as representative data point.

- DBSCAN defines clusters as continuous regions of high density.
  - detects and excludes outliers automatically



- Agglomerative (hierarchical) clustering makes nested clusters.

Applications

Ganglmair and Wardlaw, "Complexity, Standardization, and the Design of Loan Agreements"

- use k-medoid clustering to identify different types of debt contracts, and analyze customization.
- used for descriptive analysis $\rightarrow$ e.g., that larger deals have more customization.

Applications

Ganglmair and Wardlaw, "Complexity, Standardization, and the Design of Loan Agreements"

- use k-medoid clustering to identify different types of debt contracts, and analyze customization.
- used for descriptive analysis $\rightarrow$ e.g., that larger deals have more customization.

Hoberg and Phillips, "Text-Based Network Industries and Endogenous Product Differentiation"

- "business description" section from annual regulatory filings, preprocessed by extracting nouns, drop words appearing in more than 25% of documents.
- vector representation: binary for whether word appears (rather than counts)
- clusters of these vectors are "industries" – sets of firms with similar lists of nouns in their business descriptions.

Note on terms and documents

- Recall that in $\boldsymbol{X}$,
  - each row / document $\boldsymbol{X}_{[d,:]}$ is a distribution over terms
  - each column / term $\boldsymbol{X}_{[:,w]}$ is a distribution over documents.
- The same methods we used on the rows can be used on the columns:
  - apply cosine similarity to the columns to compare words (rather than compare documents)
  - apply k-means clustering to the columns to get clusters of similar words (rather than clusters of documents)