Language Models for Law and Social Science

3. Dimensionality and Distance

Check for Understanding – Lectures 1 and 2

Different Goals, Different Methods

- Supervised Learning (next week)
  - pursuing a known goal, e.g., predicting whether a political speech is from a Democrat or a Republican.
  - machine learns to replicate labels for new data points

Different Goals, Different Methods

- Supervised Learning (next week)
  - pursuing a known goal, e.g., predicting whether a political speech is from a Democrat or a Republican.
  - machine learns to replicate labels for new data points
- Unsupervised Learning (today)
  - algorithm discovers themes/patterns in text (or other high-dimensional data)
  - human interprets the results (e.g. inspect content of topics or clsuters)

Different Goals, Different Methods

- Supervised Learning (next week)
  - pursuing a known goal, e.g., predicting whether a political speech is from a Democrat or a Republican.
  - machine learns to replicate labels for new data points
- Unsupervised Learning (today)
  - algorithm discovers themes/patterns in text (or other high-dimensional data)
  - human interprets the results (e.g. inspect content of topics or clsuters)
- Both strategies amplify human effort, each in different ways.

Different Goals, Different Methods

- Supervised Learning (next week)
  - pursuing a known goal, e.g., predicting whether a political speech is from a Democrat or a Republican.
  - machine learns to replicate labels for new data points
- Unsupervised Learning (today)
  - algorithm discovers themes/patterns in text (or other high-dimensional data)
  - human interprets the results (e.g. inspect content of topics or clsuters)
- Both strategies amplify human effort, each in different ways.
- Distinctions are not clear-cut:
  - supervised learning models can be used to discover themes/patterns
  - unsupervised learning models can be used in service of prediction or known goals.
  - "self-supervised" learning – models learn language from document structure (e.g. language modeling)

Objectives: Social-Science Research using Unsupervised Learning

1. **What is the research question?**

Objectives: Social-Science Research using Unsupervised Learning

1. **What is the research question?**
2. Corpus and Data:
   - obtain, clean, preprocess, and link.
   - Produce descriptive visuals and statistics on the text and metadata

Objectives: Social-Science Research using Unsupervised Learning

1. **What is the research question?**
2. Corpus and Data:
   - obtain, clean, preprocess, and link.
   - Produce descriptive visuals and statistics on the text and metadata
3. Unsupervised learning:
   - **What are we trying to measure?**

Objectives: Social-Science Research using Unsupervised Learning

1. **What is the research question?**
2. Corpus and Data:
   - obtain, clean, preprocess, and link.
   - Produce descriptive visuals and statistics on the text and metadata
3. Unsupervised learning:
   - **What are we trying to measure?**
   - Select a model and train it.
   - Probe sensitivity to hyperparameters.
   - Validate that the model is measuring what we want.

Objectives: Social-Science Research using Unsupervised Learning

1. **What is the research question?**
2. Corpus and Data:
   - obtain, clean, preprocess, and link.
   - Produce descriptive visuals and statistics on the text and metadata
3. Unsupervised learning:
   - **What are we trying to measure?**
   - Select a model and train it.
   - Probe sensitivity to hyperparameters.
   - Validate that the model is measuring what we want.
4. Empirical analysis
   - Produce statistics or predictions with the trained model.
   - **Answer the research question.**

Outline

Document Distance

Dimensionality Reduction

Topic Models

Text Re-Use

- Text Re-Use algorithms (like "Smith-Waterman") measure similarity by finding and counting shared sequences in two texts above some minimum length, e.g. 10 words.
  - useful for plagiarism detection, for example.
- precise but can be slow
  - shortcut: use hashed n-grams with N $\geq$ 5

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

- Recall in the previous lecture we represented each document i as a vector x_i,
  - for example $x_i =$ term counts or $x_i =$ term frequencies.

Cosine Similarity

- Recall in the previous lecture we represented each document i as a vector x_i,
  - for example $x_i =$ term counts or $x_i =$ term frequencies.
- 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

- Recall in the previous lecture we represented each document i as a vector x_i,
  - for example $x_i =$ term counts or $x_i =$ term frequencies.
- 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

- Recall in the previous lecture we represented each document i as a vector x_i,
  - for example $x_i =$ term counts or $x_i =$ term frequencies.
- 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:

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

(note: same as Pearson correlation)

```
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)
```

TF-IDF Weighting

- TF/IDF: "Term-Frequency / Inverse-Document-Frequency."
- The formula for word w in document k:

$$\underbrace{\frac{\text{Count of } w \text{ in } k}{\text{Total word count of } k}}_{\text{Term Frequency}} \times \underbrace{\log\left(\frac{\text{Number of documents in } D}{\text{Count of documents containing } w}\right)}_{\text{Inverse Document Frequency}}$$

TF-IDF Weighting

- TF/IDF: "Term-Frequency / Inverse-Document-Frequency."
- The formula for word w in document k:

$$\underbrace{\frac{\text{Count of } w \text{ in } k}{\text{Total word count of } k}}_{\text{Term Frequency}} \times \underbrace{\log\left(\frac{\text{Number of documents in } D}{\text{Count of documents containing } w}\right)}_{\text{Inverse Document Frequency}}$$

- The formula up-weights relatively rare words that do not appear in all documents.
  - These words are probably more distinctive of topics or differences between documents.
- Down-weights words showing up in all documents, eg stopwords
  - these words are less semantically/topically distinctive

TF-IDF Weighting

- TF/IDF: "Term-Frequency / Inverse-Document-Frequency."
- The formula for word w in document k:

$$\underbrace{\frac{\text{Count of } w \text{ in } k}{\text{Total word count of } k}}_{\text{Term Frequency}} \times \underbrace{\log\left(\frac{\text{Number of documents in } D}{\text{Count of documents containing } w}\right)}_{\text{Inverse Document Frequency}}$$

- The formula up-weights relatively rare words that do not appear in all documents.
  - These words are probably more distinctive of topics or differences between documents.
- Down-weights words showing up in all documents, eg stopwords
  - these words are less semantically/topically distinctive
- cosine similarities between TF-IDF weighted vectors (tf-idf similarity) is a workhorse in information retrieval
  - give robustly useful metrics of semantic/topical similarity.
  - used for example in bm25 and elasticsearch.

scikit-learn's TfidfVectorizer

https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction

https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

```
>>> from sklearn.feature_extraction.text import TfidfVectorizer
>>> vectorizer = TfidfVectorizer()
>>> vectorizer.fit_transform(corpus)
<4x9 sparse matrix of type '<... 'numpy.float64'>'
    with 19 stored elements in Compressed Sparse ... format>
```

scikit-learn's TfidfVectorizer

https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction

https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

```
>>> from sklearn.feature_extraction.text import TfidfVectorizer
>>> vectorizer = TfidfVectorizer()
>>> vectorizer.fit_transform(corpus)
<4x9 sparse matrix of type '<... 'numpy.float64'>'
    with 19 stored elements in Compressed Sparse ... format>
```

- `corpus` is a sequence of strings, e.g. pandas data-frame columns.
- pre-processing options: strip accents, lowercase, drop stopwords,
- n-grams: can produce phrases up to length n (words or characters).
- vocab options: min/max frequency, vocab size
- post-processing: binary, l2 norm, (smoothed) idf weighting, etc

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.

Alternative distance metrics:

- dot product (sensitive to document length)
- Euclidean distance, $||v_1 - v_2||$
- jaccard distance (works well with binary features)
- Jensen-Shannon Divergence
- etc.

hopefully empirical results are not sensitive to choice of distance metric.

Burgess et al, "Legislative Influence Detectors"

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

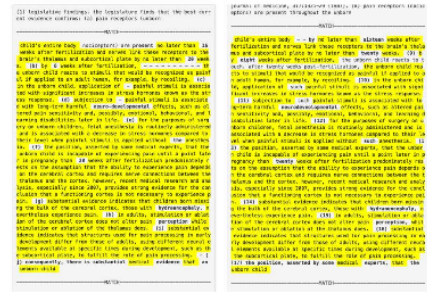


Figure 10: Match between Scott Walker’s bill and a highly similar bill from Louisiana. For a detailed view, please visit http://dssg.uchicago.edu/lid/.

Burgess et al, “Legislative Influence Detectors”

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

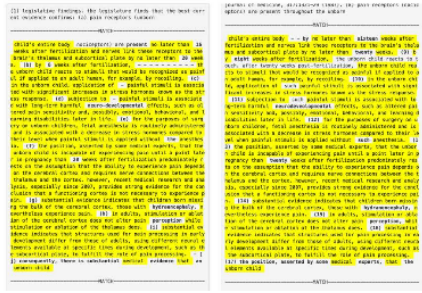


Figure 10: Match between Scott Walker’s bill and a highly similar bill from Louisiana. For a detailed view, please visit http://dssg.uchicago.edu/lid/.

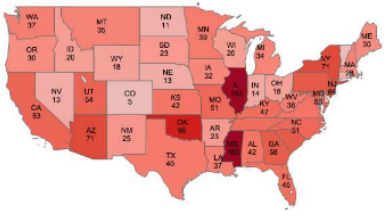


Figure 7: Introduced bills by state from ALEC model legislation

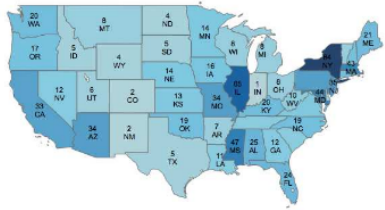


Figure 8: Introduced bills by state from ALICE model legislation

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.

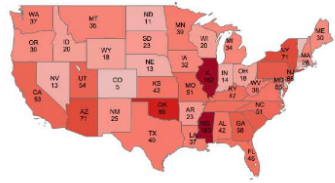


Figure 7: Introduced bills by state from ALEC model legislation

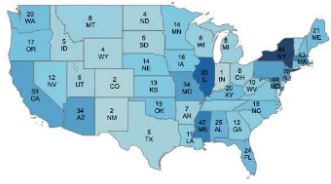


Figure 8: Introduced bills by state from ALICE model legislation

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?

Document Distance – not solved, not well-defined

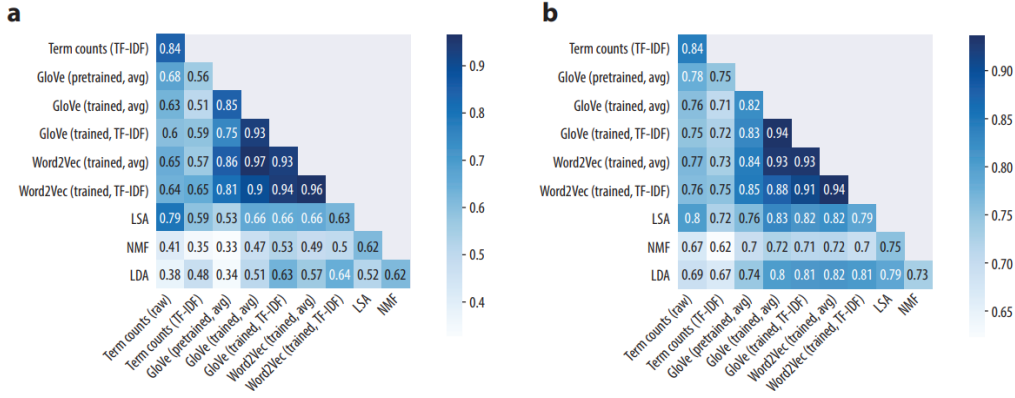


Figure 3

Comparison of algorithms for measuring document similarity. We begin with the corpus of Risk Factors sections of firms' 2019 10-K filings and compute pairwise cosine similarities across firms according to each of 10 different algorithms. Panel *a* presents the Pearson correlation between similarity scores produced by each pair of algorithms. For panel *b*, we draw 10,000 random document triplets, and for each triplet and algorithm we record whether the second or third document is closest to the first. Panel *b* presents agreement rates between algorithms in this ranking exercise. Two algorithms that produce independent rankings will agree in half of the cases, so the scale varies from 0.5 to 1. Abbreviations: LDA, latent Dirichlet allocation; LSA, latent semantic analysis; NMF, nonnegative matrix factorization; TF-IDF, term frequency-inverse document frequency.

Outline

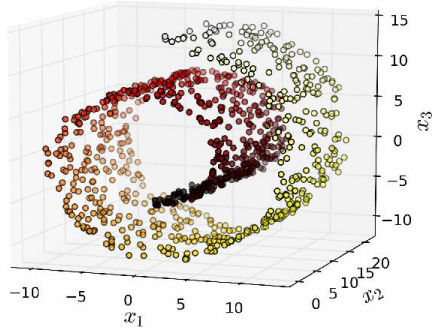
Document Distance

Dimensionality Reduction

Topic Models

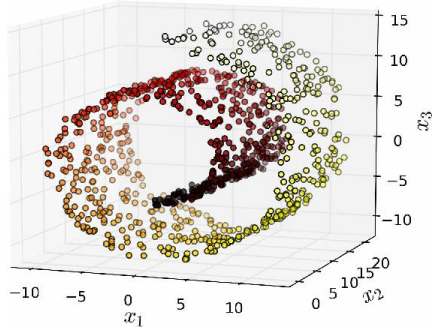
- Datasets are not distributed uniformly across the feature space.
- They have a lower-dimensional latent structure – a **manifold** – that can be learned.

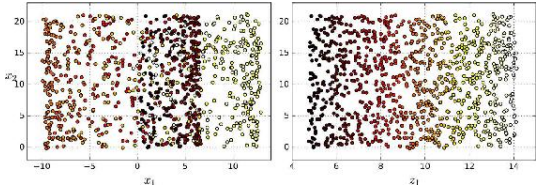
"The Swiss Roll"



"The Swiss Roll"

- Datasets are not distributed uniformly across the feature space.
- They have a lower-dimensional latent structure – a **manifold** – that can be learned.





- **Dimensonality reduction** makes data more interpretable – for example by projecting down to two dimensions for visualization.
- improves computational tractability.
- can improve model performance.

What dimension reductions have we already tried?

The Document-Term Matrix is high-dimensional

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
  - term vocabularies can be in the hundreds of thousands
- each word/column $\boldsymbol{X}_{[:,w]}$ is a distribution over documents.
  - many interesting corpora have millions of documents

$\rightarrow$ **$\boldsymbol{X}$ often has billions of cells.**

Dimension reduction of N-grams with collocation

- Conceptually, the goal of including n-grams is to featurize **collocations**:
  - Non-compositional: the meaning is not the sum of the parts (kick+the+bucket≠"kick the bucket")

Dimension reduction of N-grams with collocation

- Conceptually, the goal of including n-grams is to featurize **collocations**:
  - Non-compositional: the meaning is not the sum of the parts (kick+the+bucket≠"kick the bucket")
  - Non-substitutable: cannot substitute components with synonyms ("fast food"≠"quick food")

Dimension reduction of N-grams with collocation

- Conceptually, the goal of including n-grams is to featurize **collocations**:
  - Non-compositional: the meaning is not the sum of the parts (kick+the+bucket≠"kick the bucket")
  - Non-substitutable: cannot substitute components with synonyms ("fast food"≠"quick food")
  - Non-modifiable: cannot modify with additional words or grammar: (e.g., "kick around the bucket", "kick the buckets")

Dimension Reduction of N-grams: Point-wise mutual information

- A metric for identifying collocations is point-wise mutual information:

$$\begin{aligned}\mathrm{PMI}(w_1, w_2) &= \frac{\Pr(w_1\_w_2)}{\Pr(w_1)\Pr(w_2)} \\ &= \frac{\text{Prob. of collocation, actual}}{\text{Prob. of collocation, if independent}}\end{aligned}$$

where w_1 and w_2 are words in the vocabulary, and w_1, w_2 is the N-gram $w_1\_w_2$.

- ranks words by how often they collocate, relative to how often they occur apart.

Dimension Reduction of N-grams: Point-wise mutual information

- A metric for identifying collocations is point-wise mutual information:

$$\begin{aligned}\mathrm{PMI}(w_1, w_2) &= \frac{\Pr(w_1\_w_2)}{\Pr(w_1)\Pr(w_2)} \\ &= \frac{\text{Prob. of collocation, actual}}{\text{Prob. of collocation, if independent}}\end{aligned}$$

  where w_1 and w_2 are words in the vocabulary, and w_1, w_2 is the N-gram $w_1\_w_2$.
  - ranks words by how often they collocate, relative to how often they occur apart.

- Generalizes to longer phrases (length N) as the geometric mean of the probabilities:

$$\frac{\Pr(w_1, ..., w_N)}{\prod_{i=1}^{n} \sqrt[n]{\Pr(w_i)}}$$

- E.g., for trigrams:

$$\frac{\Pr(w_1, w_2, w_3)}{\sqrt[3]{\Pr(w_1)\Pr(w_2)\Pr(w_3)}}$$

Dimension Reduction of N-grams: Point-wise mutual information

- A metric for identifying collocations is point-wise mutual information:

$$\begin{aligned}\mathrm{PMI}(w_1, w_2) &= \frac{\Pr(w_1\_w_2)}{\Pr(w_1)\Pr(w_2)} \\ &= \frac{\text{Prob. of collocation, actual}}{\text{Prob. of collocation, if independent}}\end{aligned}$$

  where w_1 and w_2 are words in the vocabulary, and w_1, w_2 is the N-gram $w_1\_w_2$.
  - ranks words by how often they collocate, relative to how often they occur apart.

- Generalizes to longer phrases (length N) as the geometric mean of the probabilities:

$$\frac{\Pr(w_1, ..., w_N)}{\prod_{i=1}^{n} \sqrt[n]{\Pr(w_i)}}$$

- E.g., for trigrams:

$$\frac{\Pr(w_1, w_2, w_3)}{\sqrt[3]{\Pr(w_1)\Pr(w_2)\Pr(w_3)}}$$

- Warning: Rare words that appear together once or twice will have high PMI.
  - Address this with minimum frequency thresholds.

Constituency Parsing with Parts of Speech for N-Gram Dimension Reduction

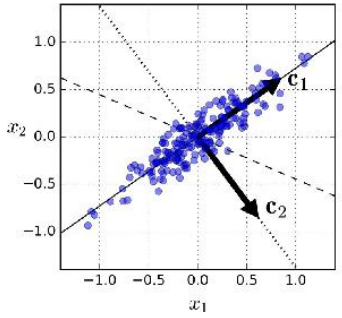
- A: Adjective, N: Noun, V: Verb, P: Preposition, D: Determinant, C: Conjunction.
- 2-grams: AN, NN, VN, VV, NV, VP.
  - tax credit, magistrate judge
- 3-grams: NNN, AAN, ANN, NAN, NPN, VAN, VNN, AVN, VVN, VPN, ANV, NVV, VDN, VVV, NNV, VVP, VAV, VVN, NCN, VCV, ACA, PAN.
  - armed and dangerous, stating the obvious

Constituency Parsing with Parts of Speech for N-Gram Dimension Reduction

- A: Adjective, N: Noun, V: Verb, P: Preposition, D: Determinant, C: Conjunction.
- 2-grams: AN, NN, VN, VV, NV, VP.
  - tax credit, magistrate judge
- 3-grams: NNN, AAN, ANN, NAN, NPN, VAN, VNN, AVN, VVN, VPN, ANV, NVV, VDN, VVV, NNV, VVP, VAV, VVN, NCN, VCV, ACA, PAN.
  - armed and dangerous, stating the obvious
- 4-grams: NCVN, ANNN, NNNN, NPNN, AANN, ANNN, ANPN, NNPN, NPAN, ACAN, NCNN, NNCN, ANCN, NCAN, PDAN, PNPN, VDNN, VDAN, VVDN.
  - `Beyond a reasonable doubt (preposition, article, adjective, noun)`
  - `Earned income tax credit (adjective, noun, noun, noun)`

PCA (principal component analysis) / SVD (singular value decomposition)

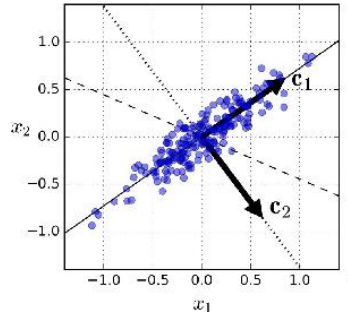
PCA (principal component analysis) / SVD (singular value decomposition)



- PCA computes the dimension in data explaining most variance.

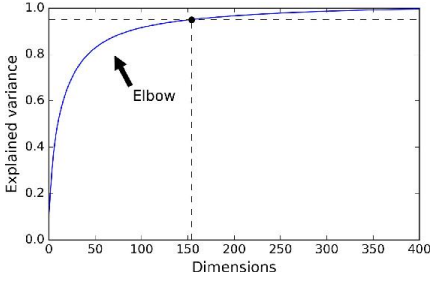
```
from sklearn.decomposition import PCA
pca = PCA(n_components=10)
X_train_pca = pca.fit_transform(X_train)
```

PCA (principal component analysis) / SVD (singular value decomposition)



- PCA computes the dimension in data explaining most variance.

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



- after the first component, subsequent components learn the (orthogonal) dimensions explaining most variance in dataset after projecting out first component.

PCA/NMF for Dimension Reduction

Data can be reduced by projecting down to first principal component dimensions.

- Distance metrics between observations (e.g. cosine similarity) are approximately preserved.

PCA/NMF for Dimension Reduction

Data can be reduced by projecting down to first principal component dimensions.

- Distance metrics between observations (e.g. cosine similarity) are approximately preserved.
- For supervised learning, reduced matrix be used as predictors instead of the original matrix.
  - but might destroy (a lot of) predictive information in your dataset.
  - compromise: use feature selection to keep strong predictors, and take principal components of weak predictors.

PCA/NMF for Dimension Reduction

Data can be reduced by projecting down to first principal component dimensions.

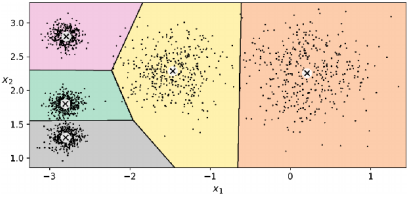
- Distance metrics between observations (e.g. cosine similarity) are approximately preserved.
- For supervised learning, reduced matrix be used as predictors instead of the original matrix.
  - but might destroy (a lot of) predictive information in your dataset.
  - compromise: use feature selection to keep strong predictors, and take principal components of weak predictors.
- PCA dimensions are not interpretable.
  - For non-negative data (e.g. counts or frequencies), **Non-negative Matrix Factorization (NMF)** provides more interpretable factors than PCA.

k-means clustering separates observations into k groups

- Matrix of predictors treated as a Euclidean space (should standardize all columns)
- algorithm: initialize cluster centroids randomly, then shift around to minimize sum of within-cluster squared distance

k-means clustering separates observations into k groups

- Matrix of predictors treated as a Euclidean space (should standardize all columns)
- algorithm: initialize cluster centroids randomly, then shift around to minimize sum of within-cluster squared distance

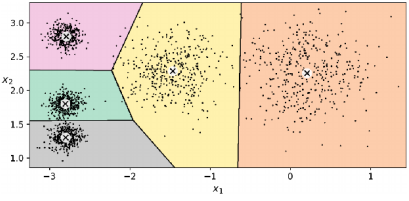


K-Means decision boundaries (Voronoi tessellation)

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

k-means clustering separates observations into k groups

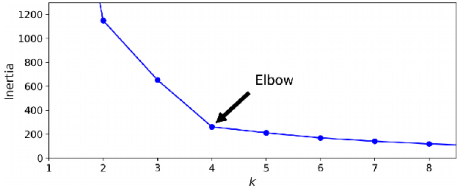
- Matrix of predictors treated as a Euclidean space (should standardize all columns)
- algorithm: initialize cluster centroids randomly, then shift around to minimize sum of within-cluster squared distance



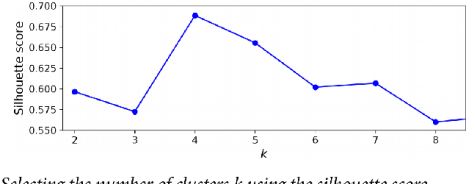
K-Means decision boundaries (Voronoi tessellation)

```
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

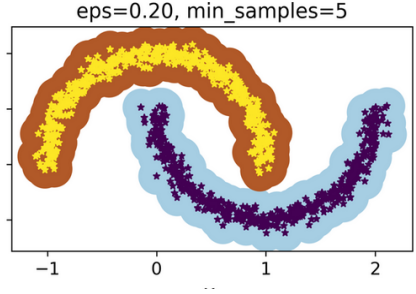
Other clustering algorithms

- "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 continous 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 represetnation: 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)

Outline

Document Distance

Dimensionality Reduction

Topic Models

Topic Models in Social Science

- 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

Topic Models in Social Science

- 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

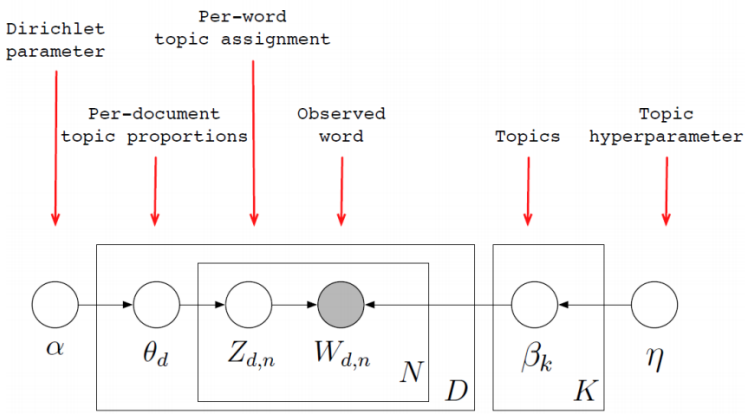
Topic Models in Social Science

- 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.

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

- 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, use coherence).
- Like PCA or NMF, LDA works by factorizing X into:
  - an $N \times K$ document-topic matrix
  - an $K \times M$ topic-term matrix.

- 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, use coherence).
- Like PCA or NMF, LDA works by factorizing X into:
  - an $N \times K$ document-topic matrix
  - an $K \times M$ topic-term matrix.



Variational inference setup (Brandon Stewart slides).

```
# train LDA with 10 topics and print
from gensim.models.ldamodel import LdaModel
lda = LdaModel(doc_term_matrix, num_topics=10,
               id2word = dictionary, passes=3)
lda.show_topics(formatted=False)
```

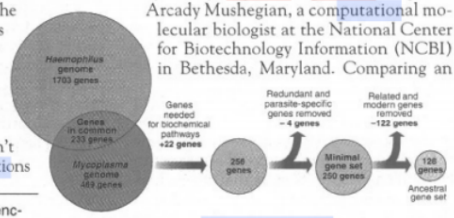
```
# to get the topic proportions for a document, use
# the corresponding row from the document-term matrix.
lda[doc_term_matrix[1]]
```

A statistical highlighter

Seeking Life's Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK—How many genes does an organism need to survive? Last week at the genome meeting here,* two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

Although the numbers don't match precisely, those predictions "are not all that far apart," especially in comparison to the 75,000 genes in the human genome, notes Siv Andersson of Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a genetic numbers game, particularly as more and more genomes are completely mapped and sequenced. "It may be a way of organizing any newly sequenced genome," explains Arcady Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an



Stripping down. Computer analysis yields an estimate of the minimum modern and ancient genomes.

* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

Image from Hanna Wallach

“Topic Modeling for the People” (Antoniak 2022)

“Topic Modeling for the People” (Antoniak 2022)

- LDA is still strong and reliable, especially for small and medium datasets
- Newer models are not automatically better for interpretability
- Human evaluation matters more than automated coherence scores
- Try multiple numbers of topics, there is no single “correct” choice
- Longer documents produce better topics than very short texts

Antoniak (2022): Modeling Choices

- Tune the document–topic prior more than the word–topic prior
- Gibbs sampling often gives clearer topics than variational methods
- Avoid stemming, it usually hurts interpretability
- Do not over-remove stopwords, they are less harmful than assumed
- Remove duplicate or near-duplicate documents
- Curate your corpus deliberately, topic models reflect what you feed them

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

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
- documents with highest share in a topic work as representative documents for the topic.

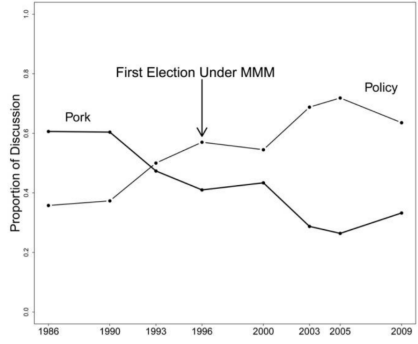
Using an LDA Model

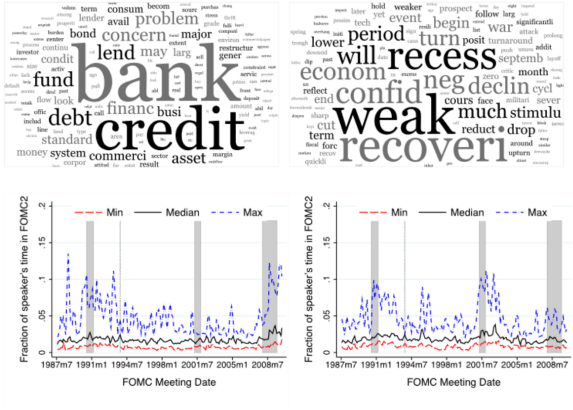
Once trained, can easily get topic proportions for a corpus.

- for any document – doesn't have to be in training corpus.
- main topic is the highest-probability topic
- documents with 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.

- e.g., Catalinac (2016) shows that after a Japanese political reform that reduced intraparty competition, candidate platforms reduced local pork and increased national policy.





(A) Topic 38 'Financial sector' **(B)** Topic 39 'Economic weakness'

Figure 1: Illustration of Output of Latent Dirichlet Allocation

This figure, from Hansen et al. (2018), illustrates the output of LDA estimated on the corpus of Federal Open Market Committee transcripts. The word clouds represent two topic-term distributions $\boldsymbol{\beta}_{38}$ and $\boldsymbol{\beta}_{39}$. The size of the words is approximately proportional to the frequency of the term in the topic. Document-topic distributions $\boldsymbol{\theta}_{it}$ are estimated for each FOMC member i and meeting t. The time series plots at the bottom show the maximum, median, and minimum value of these distributions for each meeting across members.

TABLE 1 A Summary of Common Assumptions and Relative Costs Across Different Methods of Discrete Text Categorization

	Method				
A. Assumptions	***Reading***	***Human Coding***	***Dictionaries***	***Supervised Learning***	***Topic Model***
Categories are known	No	Yes	Yes	Yes	No
Category nesting, if any, is known	No	Yes	Yes	Yes	No
Relevant text features are known	No	No	Yes	Yes	Yes
Mapping is known	No	No	Yes	No	No
Coding can be automated	No	No	Yes	Yes	Yes
B. Costs					
Preanalysis Costs					
Person-hours spent conceptualizing	Low	High	High	High	Low
Level of substantive knowledge	Moderate/High	High	High	High	Low
Analysis Costs					
Person hours spent per text	High	High	Low	Low	Low
Level of substantive knowledge	Moderate/High	Moderate	Low	Low	Low
Postanalysis Costs					
Person-hours spent interpreting	High	Low	Low	Low	Moderate
Level of substantive knowledge	High	High	High	High	High

Recommended: read this part of Quinn, Monroe, Colaresi, Crespin, and Radev (2010).

Structural Topic Model = LDA + Metadata

Roberts, Stewart, and Tingley

STM provides two ways to include contextual information:

- Topic prevalence can vary by metadata
  - e.g. Republicans talk about military issues more then Democrats

Structural Topic Model = LDA + Metadata

Roberts, Stewart, and Tingley

STM provides two ways to include contextual information:

- Topic prevalence can vary by metadata
  - e.g. Republicans talk about military issues more then Democrats
- Topic content can vary by metadata
  - e.g. Republicans talk about military issues more patriotically than Democrats.

Structural Topic Model = LDA + Metadata

Roberts, Stewart, and Tingley

STM provides two ways to include contextual information:

- Topic prevalence can vary by metadata
  - e.g. Republicans talk about military issues more then Democrats
- Topic content can vary by metadata
  - e.g. Republicans talk about military issues more patriotically than Democrats.
- Structural topic model is not a prediction model:
  - it will tell you which topics or features correlate with an outcome, but it will not provide an in-sample or out-of-sample prediction for an outcome

Structural Topic Model = LDA + Metadata

Roberts, Stewart, and Tingley

STM provides two ways to include contextual information:

- Topic prevalence can vary by metadata
  - e.g. Republicans talk about military issues more then Democrats
- Topic content can vary by metadata
  - e.g. Republicans talk about military issues more patriotically than Democrats.
- Structural topic model is not a prediction model:
  - it will tell you which topics or features correlate with an outcome, but it will not provide an in-sample or out-of-sample prediction for an outcome
- The main implementation is in R. gensim has a light-weight version called "author topic model" (see this week's notebook).

Objectives: Social-Science Research using Unsupervised Learning

1. **What is the research question?**
2. Corpus and Data:
   - obtain, clean, preprocess, and link.
   - Produce descriptive visuals and statistics on the text and metadata
3. Unsupervised learning:
   - **What are we trying to measure?**
   - Select a model and train it.
   - Probe sensitivity to hyperparameters.
   - Validate that the model is measuring what we want.
4. Empirical analysis
   - Produce statistics or predictions with the trained model.
   - **Answer the research question.**