Carnegie Mellon University
Heinz College

Unstructured Data Analysis

Lecture 9: Wrap up clustering; topic modeling

George Chen

Clustering Methods

There's a whole zoo of clustering methods

Several main categories (although there are other categories!):

Generative models

1. Pretend data generated by specific model with parameters
2. Learn the parameters ("fit model to data")
3. Use fitted model to determine cluster assignments

Hierarchical clustering

Top-down: Start with everything in 1 cluster and decide on how to recursively split

Bottom-up: Start with everything in its own cluster and decide on how to iteratively merge clusters

Density-based clustering

Based on finding parts of the data with higher density

Clustering Methods

Generative models

1. Pretend data generated by specific model with parameters

2. Learn the parameters ("fit model to data")

3. Use fitted model to determine cluster assignments

Most popular models assume Euclidean distance…

You learn a model
➔ can predict cluster assignments for points not seen in training

Hierarchical clustering

Top-down: Start with everything in 1 cluster and decide on how to recursively split

Bottom-up: Start with everything in its own cluster and decide on how to iteratively merge clusters

Density-based clustering

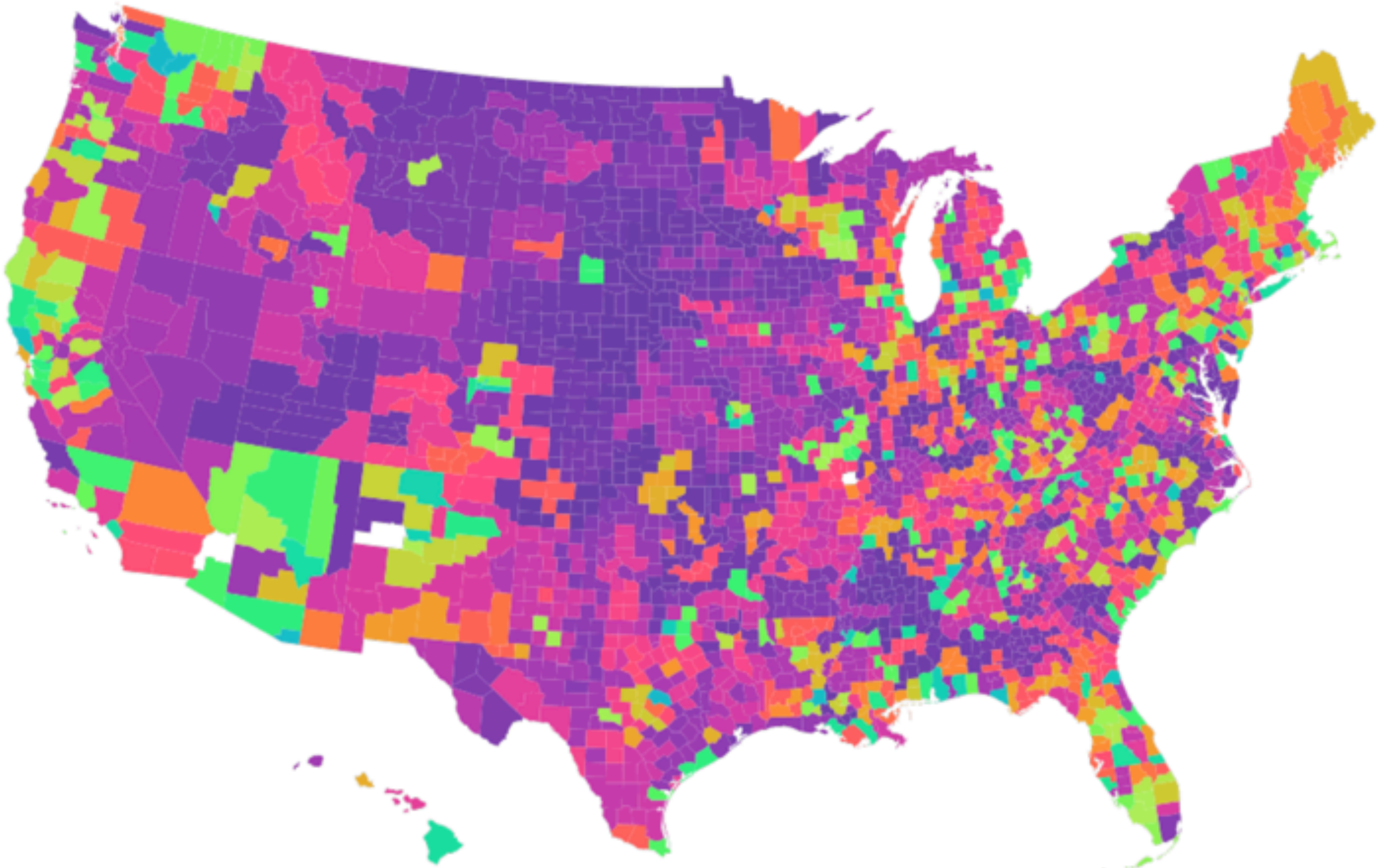
Based on finding parts of the data with higher density

These accommodate user-specified distances/similarity functions

Does not inherently model future points

Example: Clustering on US Counties

(using opioid death rate data across 37 years)



No need to predict which cluster new counties should belong to, since we're already looking at all U.S. counties!

Image source: Amanda Coston (using hierarchical clustering)

Recall: the code demo used *structured* data to hopefully make the material coverage clearer

Thoughts on Interpreting Clustering Results for *Unstructured* Data

Key idea: first run analyses on each cluster separately

- Clustering on text documents:
    - Per cluster: for text documents in the cluster, conduct frequency & co-occurrence analysis (e.g., find most frequent words, pairs of words with highest PMI)
    - If using a k-means/GMM/DP-means/DP-GMM, can look at what text documents are closest to the cluster center

Compare findings from individual clusters to try to understand in what ways the clusters are different

Also works for images instead of text

What About When Using Distances/Similarities Where “Cluster Mean” is Not Obvious?

For example, consider hierarchical clustering or DBSCAN

- Per cluster:
  - Find “centroid” to be the point that has sum of distances to all other points within the cluster to be as small as possible (or sum of similarities as large as possible)
  - Can then see what points within the cluster are closest to the centroid (and also ones that are far away)

This strategy works for structured and unstructured data represented as feature vectors

Clustering Last Remarks

Ultimately, *you* have to decide on which clustering method and number of clusters make sense for your data

- After you run a clustering algorithm, make visualizations to interpret the clusters *in the context of your application*!
- Do not just blindly rely on numerical metrics (e.g., CH index)
- Some times it makes more sense to define your own score function for how good a clustering assignment is

If you can set up a prediction task, then you can use the prediction task to guide the clustering

Is clustering structure enough?

- Fundamentally, clustering models assume each point comes from a <u>single</u> cluster
- In reality, a data point could have “mixed” membership and belong to <u>multiple</u> clusters

Topic Modeling

Text

Each document is part of multiple topics

Each topic consists of a bunch of regularly co-occurring words
(example topics: “sports”, “medicine”, “movies”, “finance”)

Movie recommendation

Each user is part of multiple “clusters”/topics

Each cluster/topic consists of a bunch of movies
(example clusters: “sci-fi epics”, “cheesy rom-coms”)

Health care

Each patient’s health records explained by multiple “topics”

Each topic consists of co-occurring “events”
(example topics: “heart condition”, “severe pancreatitis”)

Topic Modeling

Text

Each document is part of multiple topics

Each topic consists of a bunch of regularly co-occurring words
(example topics: “sports”, “medicine”, “movies”, “finance”)

In all of these examples:

- Each data point (a feature vector) is part of multiple topics
- Each topic corresponds to specific feature values in the feature vector likely appearing

(ex s”)

Health care

Each patient’s health records explained by multiple “topics”

Each topic consists of co-occurring “events”
(example topics: “heart condition”, “severe pancreatitis”)

Latent Dirichlet Allocation (LDA)

- Easy to describe in terms of text (but works for not just text)
- A generative model
- Input: "document-word" matrix, and pre-specified # topics *k*

Word

		1	2	…	*d*
Document	1 2 ⋮ *n*	Each row is a feature vector representing a raw counts histogram!			

i-th row, *j*-th column: # times word *j* appears in doc *i*

- Output: what the *k* topics are (details on this shortly)

LDA Generative Model Example

		Topic	
		weather	food
Document	Alice's text	0.1	0.9
	Bob's text	0.5	0.5

		Word			
		cold	hot	apple	pie
Topic	weather	0.3	0.7	0.0	0.0
	food	0.1	0.3	0.5	0.1

Each word in Alice's text is generated by:

1. Flip 2-sided coin for Alice
2. If weather: flip 4-sided coin for weather
   If food: flip 4-sided coin for food

LDA Generative Model Example

		Topic	
		weather	food
Document	Alice’s text	0.1	0.9
	Bob’s text	0.5	0.5

		Word			
		cold	hot	apple	pie
Topic	weather	0.3	0.7	0.0	0.0
	food	0.1	0.3	0.5	0.1

Each word in Bob’s text is generated by:

1. Flip 2-sided coin for Bob
2. If weather: flip 4-sided coin for weather
   If food: flip 4-sided coin for food

LDA Generative Model Example

		Topic	
		weather	food
Document	Alice's text	0.1	0.9
	Bob's text	0.5	0.5

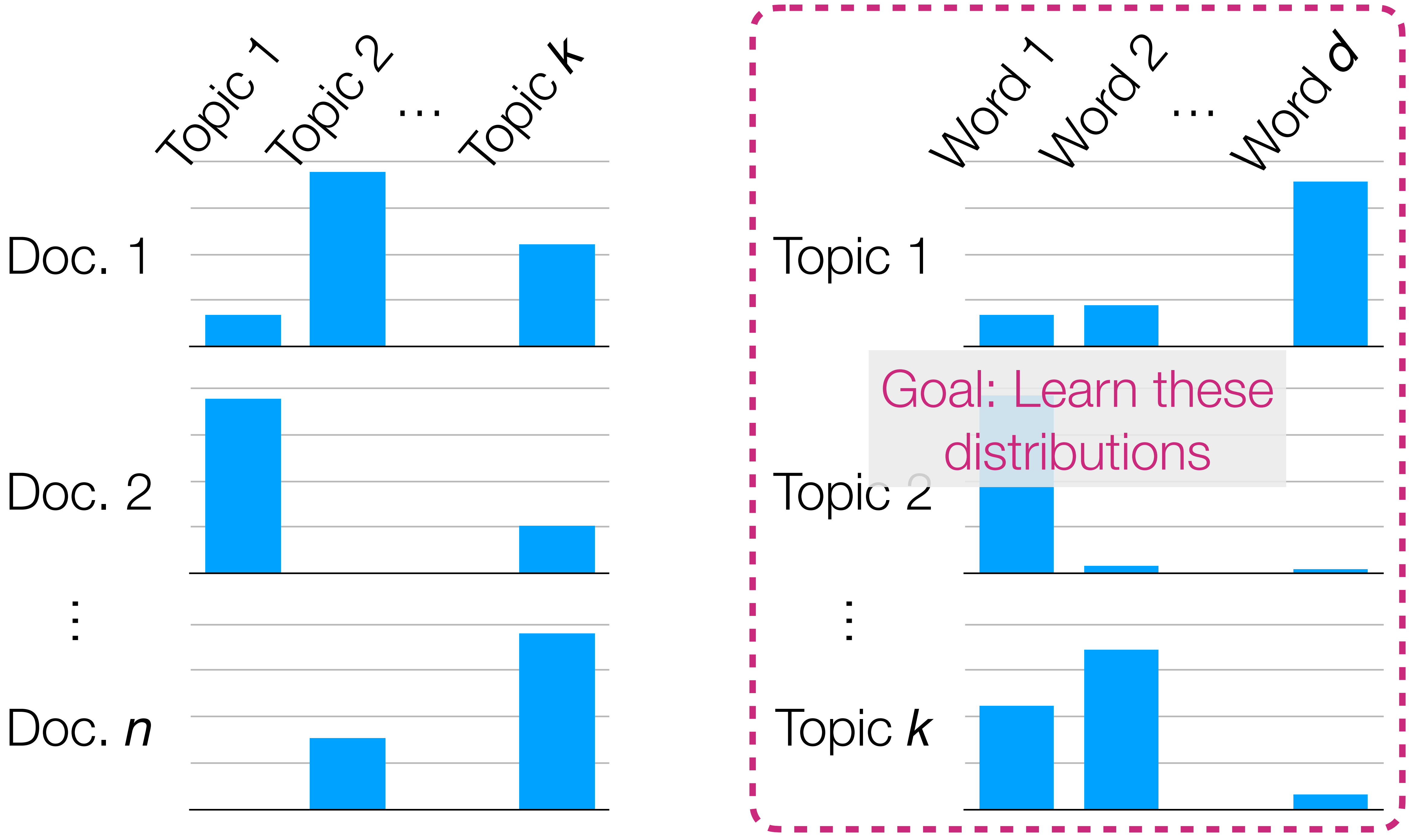
		Word			
		cold	hot	apple	pie
Topic	weather	0.3	0.7	0.0	0.0
	food	0.1	0.3	0.5	0.1

Each word in doc *i* is generated by:

1. Flip 2-sided coin for doc *i*
2. If weather: flip 4-sided coin for weather
   If food: flip 4-sided coin for food

"Learning the topics" means figuring out these 4-sided coin probabilities

LDA Generative Model



LDA models each word in document *i* to be generated as:

- Randomly choose a topic *Z* (use topic distribution for doc *i*)
- Randomly choose a word (use word distribution for topic *Z*)

LDA

- Easy to describe in terms of text (but works for not just text)
- A generative model
- Input: “document-word” matrix, and pre-specified # topics *k*

Word

1 2 … *d*

Document 1, 2, ⋮, *n*

Each row is a feature vector representing a raw counts histogram!

i-th row, *j*-th column: # times word *j* appears in doc *i*

- Output: the *k* topics’ distribution of words

LDA

Demo

An Alternative Feature Vector Representation for Text: TF-IDF

Intuition: words that appear in more documents are likely less useful (same intuition as stop words!) — let's *downweight* these words!

Word

$$\begin{array}{cc|cccc} & & 1 & 2 & \ldots & d \\ \hline & 1 & & & & \\ \textbf{Document} & 2 & & & & \\ & \vdots & & & & \\ & n & & & & \end{array}$$

Term frequency (TF)

i-th row, *j*-th column: # times word *j* appears in doc *i*

multiply TF by $\log \dfrac{1}{\mathbb{P}(\text{document has word } j)} = \log \dfrac{1}{\left(\frac{\text{\# documents that have word } j}{n}\right)}$

$$= \log \boxed{\frac{n}{\text{\# documents that have word } j}}$$

Hack: avoid dividing by 0 by adding 1 to numerator & 1 to denominator

An Alternative Feature Vector Representation for Text: TF-IDF

Intuition: words that appear in more documents are likely less useful (same intuition as stop words!) — let's *downweight* these words!

There are many TF-IDF variants! (Lots of hacks!)

Word

	1	2	…	d
Document 1				
2				
⋮				
n				

Term frequency (TF)

i-th row, j-th column: # times word j appears in doc i

$$\times \left[1 + \log \frac{1 + n}{1 + \text{\# documents that have word } j} \right]$$

Default TF-IDF weighting in sklearn

sklearn's default behavior further normalizes each row to have Euclidean norm 1

TF-IDF is in your HW2

(usage is similar to `CountVectorizer` from the demo)

How to Choose Number of Topics *k*?

Something sort of like CH index is also possible:

avoid numerical issues

For a specific topic, look at the *m* most probable words ("top words")

Coherence (within cluster/topic variability):

$$\sum_{\substack{\text{top words } v,w \\ \text{that are not the same}}} \log \frac{\#\text{ documents that contain both } v \text{ and } w + 0.1}{\#\text{ documents that contain } w}$$

log of P(see word v | see word w)

Inter-topic similarity (between cluster/topic variability):

Can average each of these across the topics

Count # top words that do not appear in any of the other topics' *m* top words

(number of "unique words")

Can plot average coherence vs *k*, and average # unique words vs *k*
(for values of k you are willing to try)

LDA

Demo

Topic Modeling: Last Remarks

- There are actually *many* topic models, not just LDA
  - Hierarchical Dirichlet Process (Bayesian nonparametric version of LDA), correlated topic models, SAGE, anchor word topic models, ProdLDA, embedded topic model, …
- Dynamic topic models: tracks how topics change *over time*
  - Example: for text over time, figure out how topics change
  - Example: for recommendation system, figure out how user tastes change over time
- Trivial to add supervision to topic models! Can have topics learned help with prediction tasks!
- Caution: learning topic models can be very sensitive to random initialization