PURDUE
UNIVERSITY

CS54701: Information Retrieval

Text Clustering

24 March 2016

Prof. Chris Clifton

Borrows slides from Chris Manning, Ray Mooney and Soumen Chakrabarti

Indiana Center for Database Systems ™

Clustering

- Document clustering
  - Motivations
  - Document representations
  - Success criteria
- Clustering algorithms
  - K-means
  - Model-based clustering (EM clustering)

2

What is clustering?

- Clustering is the process of grouping a set of physical or abstract objects into classes of similar objects
  - It is the commonest form of unsupervised learning
    - Unsupervised learning = learning from raw data, as opposed to supervised data where the correct classification of examples is given
  - It is a common and important task that finds many applications in IR and other places

3

Why cluster documents?

- Whole corpus analysis/navigation
  - Better user interface
- For improving recall in search applications
  - Better search results
- For better navigation of search results
- For speeding up vector space retrieval
  - Faster search

4

Navigating document collections

- Standard IR is like a book index
- Document clusters are like a table of contents
- People find having a table of contents useful

Index
Aardvark, 15
Blueberry, 200
Capricorn, 1, 45-55
Dog, 79-99
Egypt, 65
Falafel, 78-90
Giraffes, 45-59
…

Table of Contents
1. Science of Cognition
    1.a. Motivations
        1.a.i. Intellectual Curiosity
        1.a.ii. Practical Applications
    1.b. History of Cognitive Psychology
2. The Neural Basis of Cognition
    2.a. The Nervous System
    2.b. Organization of the Brain
    2.c. The Visual System
3. Perception and Attention
    3.a. Sensory Memory
    3.b. Attention and Sensory Information Processing

5

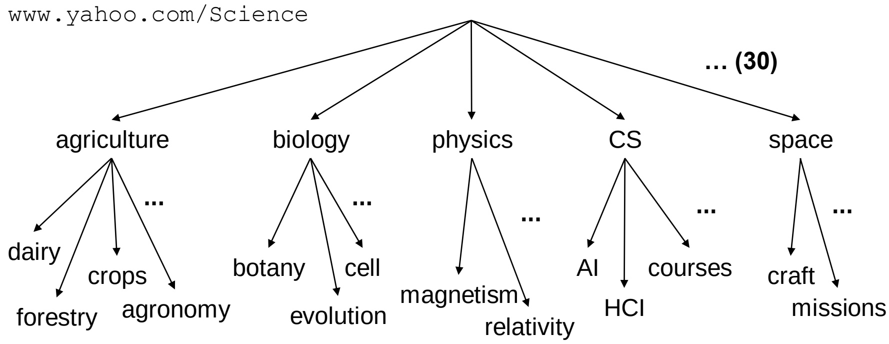
Corpus analysis/navigation

- Given a corpus, partition it into groups of related docs
    - Recursively, can induce a tree of topics
    - Allows user to browse through corpus to find information
    - Crucial need: meaningful labels for topic nodes.
- Yahoo!: manual hierarchy
    - Often not available for new document collection

6

Yahoo! Hierarchy

www.yahoo.com/Science



7

For improving search recall

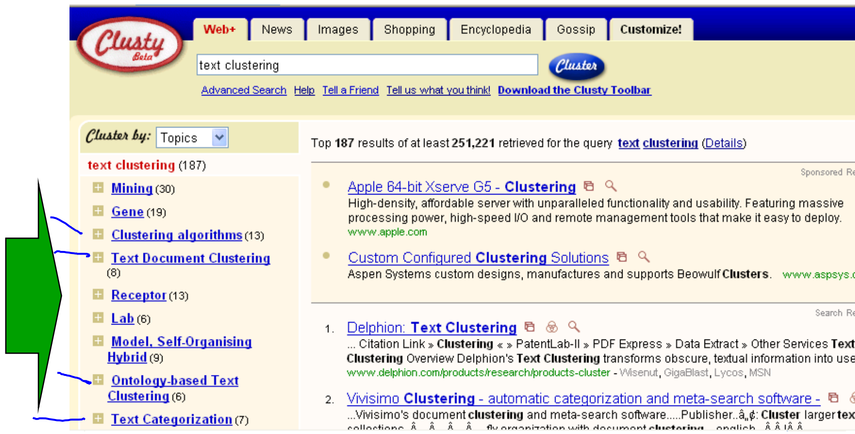
- *Cluster hypothesis* - Documents with similar text are related
- Therefore, to improve search recall:
  - Cluster docs in corpus a priori
  - When a query matches a doc *D*, also return other docs in the cluster containing *D*
- Hope if we do this: The query "car" will also return docs containing *automobile*
  - Because clustering grouped together docs containing *car* with those containing *automobile.*

Why might this happen?

8

For better navigation of search results

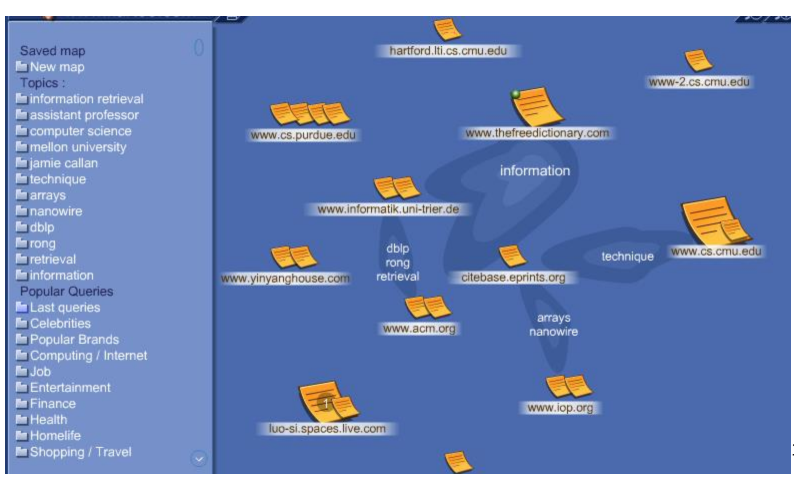
- For grouping search results thematically
  - clusty.com / Vivisimo



9

For better navigation of search results

- And more visually: Kartoo.com



10

Navigating search results (2)

- One can also view grouping documents with the same sense of a word as clustering
- Given the results of a search (e.g., jaguar, ***NLP***), partition into groups of related docs
- Can be viewed as a form of word sense disambiguation
- E.g., *jaguar* may have senses:
  - The car company
  - The animal
  - The football team
  - The video game
- Recall query reformulation/expansion discussion

11

Navigating search results (2)

12

For speeding up vector space retrieval

- In vector space retrieval, we must find nearest doc vectors to query vector
- This entails finding the similarity of the query to every doc – slow (for some applications)
- By clustering docs in corpus a priori
  - find nearest docs in cluster(s) close to query
  - inexact but avoids exhaustive similarity computation

13

What Is A Good Clustering?

- Internal criterion: A good clustering will produce high quality clusters in which:
  - the intra-class (that is, intra-cluster) similarity is high
  - the inter-class similarity is low
  - The measured quality of a clustering depends on both the document representation and the similarity measure used
- External criterion: The quality of a clustering is also measured by its ability to discover some or all of the hidden patterns or latent classes
  - Assessable with gold standard data

14

External Evaluation of Cluster Quality

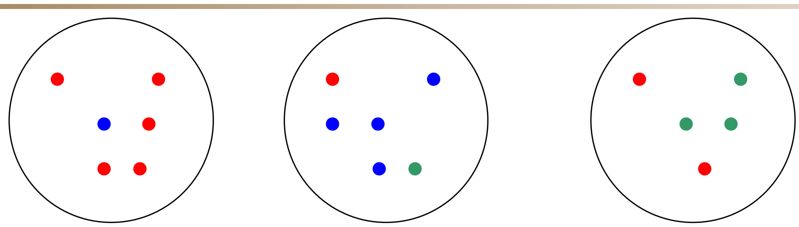
- Assesses clustering with respect to ground truth
- Assume that there are C gold standard classes, while our clustering algorithms produce *k* clusters, π_1, π_2, …, π_k with n_i members.
- Simple measure: purity, the ratio between the dominant class in the cluster π_i and the size of cluster π_i

$$Purity(\pi_i) = \frac{1}{n_i} \max_j (n_{ij}) \quad j \in C$$

- Others are entropy of classes in clusters (or mutual information between classes and clusters)

15

Purity



Cluster I Cluster II Cluster III

Cluster I: Purity = 1/6 (max(5, 1, 0)) = 5/6

Cluster II: Purity = 1/6 (max(1, 4, 1)) = 4/6

Cluster III: Purity = 1/5 (max(2, 0, 3)) = 3/5

16

Issues for clustering

- Representation for clustering
  - Document representation
    - Vector space? Normalization?
  - Need a notion of similarity/distance
- How many clusters?
  - Fixed a priori?
  - Completely data driven?
    - Avoid "trivial" clusters - too large or small
      - In an application, if a cluster's too large, then for navigation purposes you've wasted an extra user click without whittling down the set of documents much.

17

What makes docs "related"?

- Ideal: semantic similarity.
- Practical: statistical similarity
  - We will use cosine similarity.
  - Docs as vectors.
  - For many algorithms, easier to think in terms of a *distance* (rather than <u>similarity</u>) between docs.
  - We will describe algorithms in terms of cosine similarity.

Cosine similarity of normalized D_j, D_k:

$$sim(D_j, D_k) = \sum_{i=1}^{m} w_{ij} \times w_{ik}$$

Aka normalized inner product.

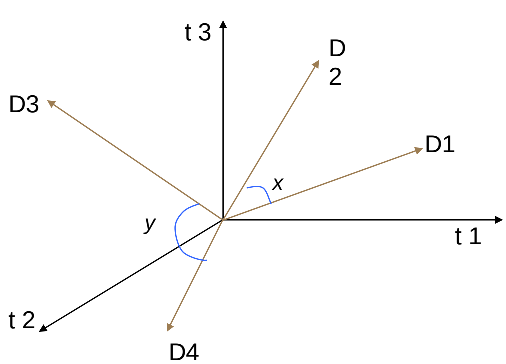
18

Recall doc as vector

- Each doc *j* is a vector of *tf*×*idf* values, one component for each term.
- Can normalize to unit length.
- So we have a vector space
  - terms are axis - aka *features*
  - *n* docs live in this space
  - even with stemming, may have 20,000+ dimensions
  - do we really want to use all terms?
    - Different from using vector space for search. Why?

19

Intuition



Postulate: Documents that are "close together" in vector space talk about the same things.

20

Clustering Algorithms

- Partitioning “flat” algorithms
  - Usually start with a random (partial) partitioning
  - Refine it iteratively
    - *k* means/medoids clustering
    - Model based clustering
- Hierarchical algorithms
  - Bottom-up, agglomerative
  - Top-down, divisive

21

Partitioning Algorithms

- Partitioning method: Construct a partition of *n* documents into a set of *k* clusters
- Given: a set of documents and the number *k*
- Find: a partition of *k* clusters that optimizes the chosen partitioning criterion
  - Globally optimal: exhaustively enumerate all partitions
  - Effective heuristic methods: k-means and k-medoids algorithms

22

How hard is clustering?

- One idea is to consider all possible clusterings, and pick the one that has best inter and intra cluster distance properties
- Suppose we are given n points, and would like to cluster them into k-clusters
  - How many possible clusterings?

$$\frac{k^n}{k!}$$

- Too hard to do it brute force or optimally
- Solution: Iterative optimization algorithms
  - Start with a clustering, iteratively improve it (eg. K-means)

23

K-Means

- Assumes documents are real-valued vectors.
- Clusters based on *centroids* (aka the *center of gravity* or mean) of points in a cluster, *c*:

$$\vec{\mu}(c) = \frac{1}{|c|}\sum_{\vec{x} \in c} \vec{x}$$

- Reassignment of instances to clusters is based on distance to the current cluster centroids.
  - (Or one can equivalently phrase it in terms of similarities)

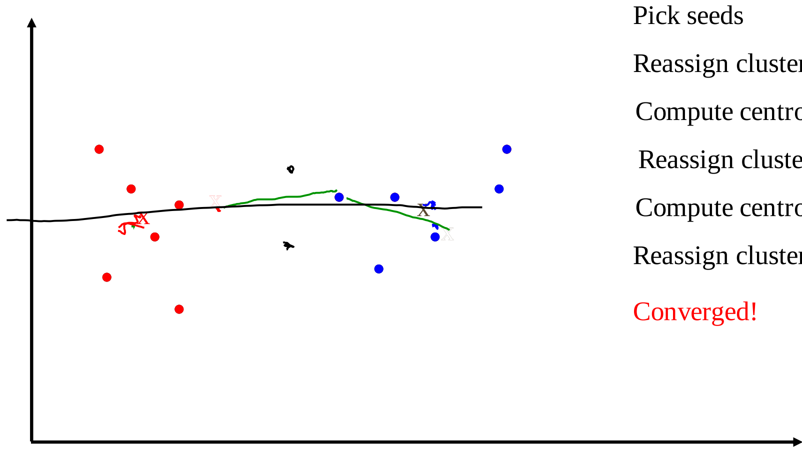
24

K-Means Algorithm

Let d be the distance measure between instances.
Select k random instances $\{s_1, s_2, \ldots s_k\}$ as seeds.
Until clustering converges or other stopping criterion:
    For each instance x_i:
        Assign x_i to the cluster c_j such that $d(x_i, s_j)$ is minimal.
    (Update the seeds to the centroid of each cluster)
    For each cluster c_j
        $s_j = \mu(c_j)$

25

K Means Example
(K=2)



Pick seeds

Reassign clusters

Compute centroids

Reassign clusters

Compute centroids

Reassign clusters

Converged!

26

Termination conditions

- Several possibilities, e.g.,
  - A fixed number of iterations.
  - Doc partition unchanged.
  - Centroid positions don't change.

Does this mean that the docs in a cluster are unchanged?

27

Time Complexity

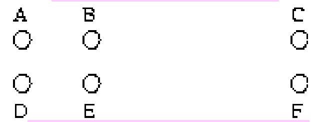
- Assume computing distance between two instances is *O(m)* where *m* is the dimensionality of the vectors.
- Reassigning clusters: *O(kn)* distance computations, or *O(knm)*.
- Computing centroids: Each instance vector gets added once to some centroid: *O(nm)*.
- Assume these two steps are each done once for *i* iterations: *O(iknm)*.
- Linear in all relevant factors, assuming a fixed number of iterations, more efficient than hierarchical agglomerative methods

28

Seed Choice

- Results can vary based on random seed selection.
- Some seeds can result in poor convergence rate, or convergence to sub-optimal clusterings.
  - Select good seeds using a heuristic (e.g., doc least similar to any existing mean)
  - Try out multiple starting points
  - Initialize with the results of another method.

Example showing sensitivity to seeds



In the above, if you start with B and E as centroids you converge to {A,B,C} and {D,E,F}
If you start with D and F you converge to {A,B,D,E} {C,F}

Exercise: find good approach for finding good starting points

29

PURDUE UNIVERSITY

CS54701: Information Retrieval

Text Clustering

29 March 2016

Prof. Chris Clifton

Borrows slides from Chris Manning, Ray Mooney and Soumen Chakrabarti

Indiana Center for Database Systems ™

Recap

- Why cluster documents?
  - For improving recall in search applications
  - For speeding up vector space retrieval
  - Navigation
  - Presentation of search results
- *k*-means basic iteration
  - At the start of the iteration, we have *k* centroids.
  - Each doc assigned to the nearest centroid.
  - All docs assigned to the same centroid are averaged to compute a new centroid;
    - thus have *k* new centroids.

How Many Clusters?

- Number of clusters *k* is given
  - Partition *n* docs into predetermined number of clusters
- Finding the "right" number of clusters is part of the problem
  - Given docs, partition into an "appropriate" number of subsets.
  - E.g., for query results - ideal value of *k* not known up front - though UI may impose limits.
- Can usually take an algorithm for one flavor and convert to the other.

k not specified in advance

- Say, the results of a query.
- Solve an optimization problem: penalize having lots of clusters
  - application dependent, e.g., compressed summary of search results list.
- Tradeoff between having more clusters (better focus within each cluster) and having too many clusters

k not specified in advance

- Given a clustering, define the Benefit for a doc to be the cosine similarity to its centroid
- Define the Total Benefit to be the sum of the individual doc Benefits.

Why is there always a clustering of Total Benefit *n*?

Penalize lots of clusters

- For each cluster, we have a Cost *C*.
- Thus for a clustering with *k* clusters, the Total Cost is *kC*.
- Define the Value of a clustering to be =

  Total Benefit - Total Cost.
- Find the clustering of highest value, over all choices of *k*.
  - Total benefit increases with increasing K. But can stop when it doesn't increase by "much". The Cost term enforces this.

Convergence

- Why should the K-means algorithm ever reach a *fixed point*?
  - A state in which clusters don't change.
- K-means is a special case of a general procedure known as the *Expectation Maximization (EM) algorithm*.
  - EM is known to converge.
  - Number of iterations could be large.

Convergence of K-Means

- Define goodness measure of cluster k as sum of squared distances from cluster centroid:
  - $G_k = \Sigma_i (v_i - c_k)^2$ (sum all v_i in cluster k)
- $G = \Sigma_k G_k$
- Reassignment monotonically reduces G since each vector is assigned to the closest centroid.
- Recomputation monotonically decreases each G_k since: (m_k is number of members in cluster)
  - $\Sigma (v_{in} - a)^2$ reaches minimum for:
  - $\Sigma -2(v_{in} - a) = 0$

K-means issues, variations, etc.

- Recomputing the centroid after every assignment (rather than after all points are re-assigned) can improve speed of convergence of K-means
- Assumes clusters are spherical in vector space
  - Sensitive to coordinate changes, weighting etc.
- Disjoint and exhaustive
  - Doesn't have a notion of "outliers"

Soft Clustering

- Clustering typically assumes that each instance is given a “hard” assignment to exactly one cluster.
- Does not allow uncertainty in class membership or for an instance to belong to more than one cluster.
- *Soft clustering* gives probabilities that an instance belongs to each of a set of clusters.
- Each instance is assigned a probability distribution across a set of discovered categories (probabilities of all categories must sum to 1).

Model based clustering

- Algorithm optimizes a probabilistic model criterion
- Clustering is usually done by the Expectation Maximization (EM) algorithm
  - Gives a soft variant of the K-means algorithm
  - Assume k clusters: $\{c_1, c_2, \ldots c_k\}$
  - Assume a probabilistic model of categories that allows computing $P(c_i \mid E)$ for each category, c_i, for a given example, E.
  - For text, typically assume a naïve Bayes category model.
  - Parameters $\theta = \{P(c_i), P(w_j \mid c_i): i \in \{1, \ldots k\}, j \in \{1, \ldots, |V|\}\}$

Expectation Maximization (EM) Algorithm

- Iterative method for learning probabilistic categorization model from unsupervised data.
- Initially assume random assignment of examples to categories.
- Learn an initial probabilistic model by estimating model parameters θ from this randomly labeled data.
- Iterate following two steps until convergence:
  - Expectation (E-step): Compute $P(c_i \mid E)$ for each example given the current model, and probabilistically re-label the examples based on these posterior probability estimates.
  - Maximization (M-step): Re-estimate the model parameters, θ, from the probabilistically re-labeled data.

EM Experiment
[Soumen Chakrabarti]

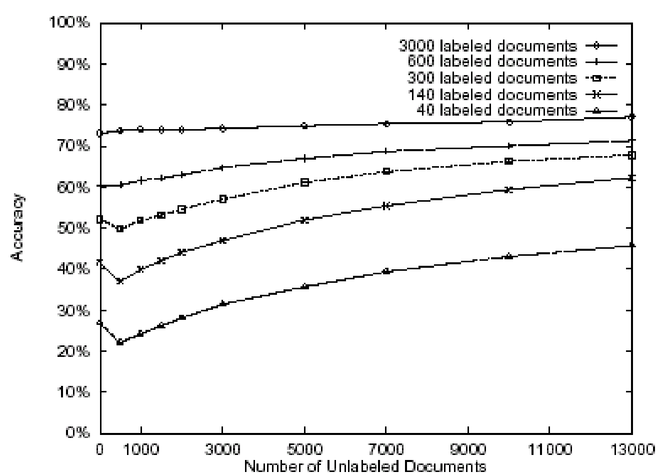
- Semi-supervised: some labeled and unlabeled data
- Take a completely labeled corpus D, and randomly select a subset as D_K.
- Also use the set $D^U \subseteq D$ of unlabeled documents in the EM procedure.
- Correct classification of a document
  => concealed class label = class with largest probability
- Accuracy with unlabeled documents > accuracy without unlabeled documents
  - Keeping labeled set of same size
- EM beats naïve Bayes with same size of labeled document set
  - Largest boost for small size of labeled set
  - Comparable or poorer performance of EM for large labeled sets

Belief in labeled documents

- Depending on one's faith in the initial labeling
  - Set before 1st iteration:
    - $\Pr(c_d \mid d) = 1 - \varepsilon$ and $\Pr(c' \mid d) = \varepsilon/(n-1)$ for all $c' \neq c_d$
  - With each iteration
    - Let the class probabilities of the labeled documents `smear' in reestimation process
- To limit 'drift' from initial labeled documents, one can add a damping factor in the E step to the contribution from unlabeled documents

Increasing D^U while holding D^K fixed also shows the advantage of using large unlabeled sets in the EM-like algorithm.

Purity

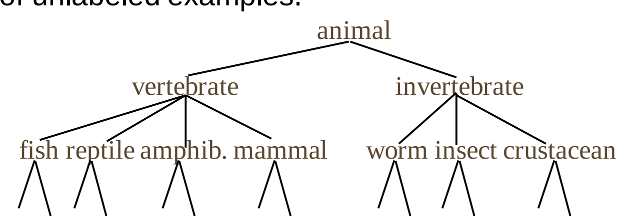


“The Curse of Dimensionality”

- Why document clustering is difficult
  - While clustering looks intuitive in 2 dimensions, many of our applications involve 10,000 or more dimensions…
  - High-dimensional spaces look different: the probability of random points being close drops quickly as the dimensionality grows.
  - One way to look at it: in large-dimension spaces, random vectors are almost all almost perpendicular. Why?
- Solution: Dimensionality reduction … important for text

Hierarchical Clustering

- Build a tree-based hierarchical taxonomy (*dendrogram*) from a set of unlabeled examples.



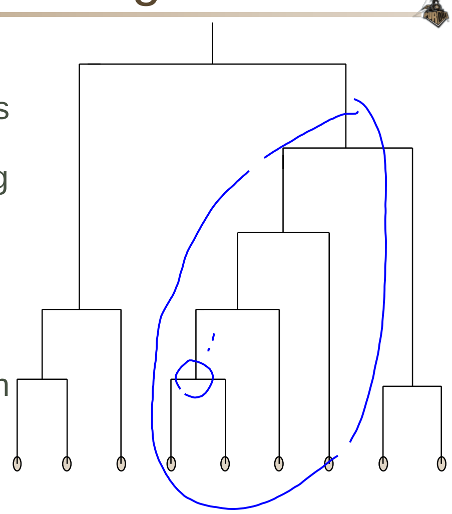
- One option to produce a hierarchical clustering is recursive application of a partitional clustering algorithm to produce a hierarchical clustering.

Hierarchical Agglomerative Clustering (HAC)

- Assumes a similarity function for determining the similarity of two instances.
- Starts with all instances in a separate cluster and then repeatedly joins the two clusters that are most similar until there is only one cluster.
- The history of merging forms a binary tree or hierarchy.

A Dendogram: Hierarchical Clustering

- Dendrogram: Decomposes data objects into a several levels of nested partitioning (tree of clusters).
- Clustering of the data objects is obtained by cutting the dendrogram at the desired level, then each connected component forms a cluster.



HAC Algorithm

Start with all instances in their own cluster.
Until there is only one cluster:

Among the current clusters, determine the two clusters, c_i and c_j, that are most similar.

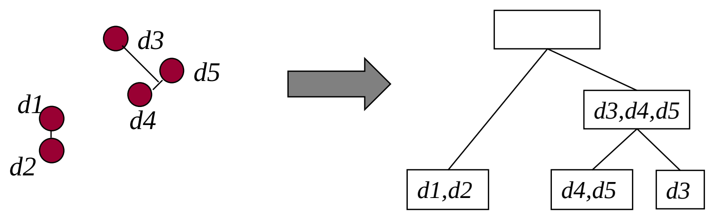
Replace c_i and c_j with a single cluster $c_i \cup c_j$

Hierarchical Clustering algorithms

- **Agglomerative (bottom-up):**
  - Start with each document being a single cluster.
  - Eventually all documents belong to the same cluster.
- **Divisive (top-down):**
  - Start with all documents belong to the same cluster.
  - Eventually each node forms a cluster on its own.
- Does not require the number of clusters ***k*** in advance
- Needs a termination/readout condition
  - The final mode in both Agglomerative and Divisive is of no use.

Dendrogram: Document Example

- As clusters *agglomerate*, docs likely to fall into a hierarchy of “topics” or concepts.



“Closest pair” of clusters

- Many variants to defining closest pair of clusters
- “Center of gravity”
  - Clusters whose centroids (centers of gravity) are the most cosine-similar
- Average-link
  - Average cosine between pairs of elements
- Single-link
  - Similarity of the most cosine-similar (single-link)
- Complete-link
  - Similarity of the “furthest” points, the least cosine-similar

Hierarchical Clustering

- Key problem: as you build clusters, how do you represent the location of each cluster, to tell which pair of clusters is closest?
- Euclidean case: each cluster has a *centroid* = average of its points.
  - Measure intercluster distances by distances of centroids.

Single Link Agglomerative Clustering

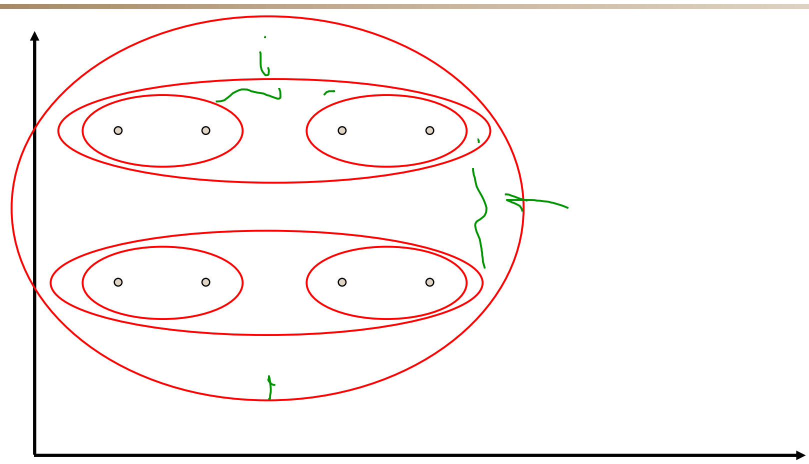
- Use maximum similarity of pairs:

$$sim(c_i, c_j) = \max_{x \in c_i, y \in c_j} sim(x, y)$$

- Can result in "straggly" (long and thin) clusters due to chaining effect.
  - Appropriate in some domains, such as clustering islands: "Hawaii clusters"
- After merging c_i and c_j, the similarity of the resulting cluster to another cluster, c_k, is:

$$sim((c_i \cup c_j), c_k) = \max(sim(c_i, c_k), sim(c_j, c_k))$$

Single Link Example



Complete Link Agglomerative Clustering

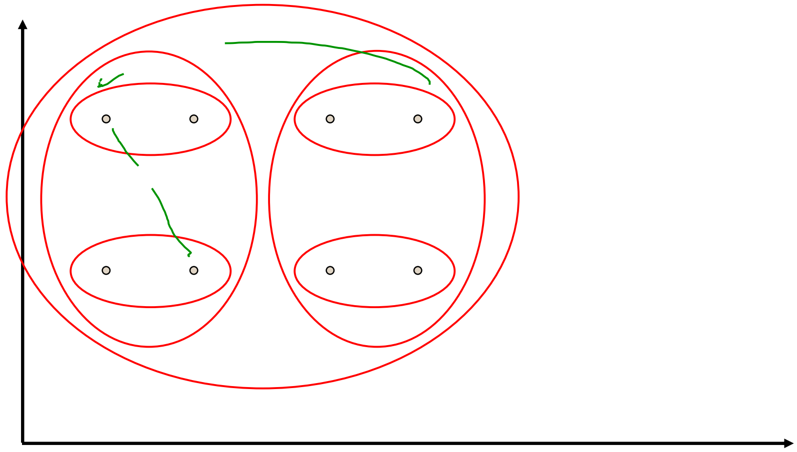
- Use minimum similarity of pairs:

$$sim(c_i, c_j) = \min_{x \in c_i, y \in c_j} sim(x, y)$$

- Makes "tighter," spherical clusters that are typically preferable.
- After merging c_i and c_j, the similarity of the resulting cluster to another cluster, c_k, is:

$$sim((c_i \cup c_j), c_k) = \min(sim(c_i, c_k), sim(c_j, c_k))$$

Complete Link Example



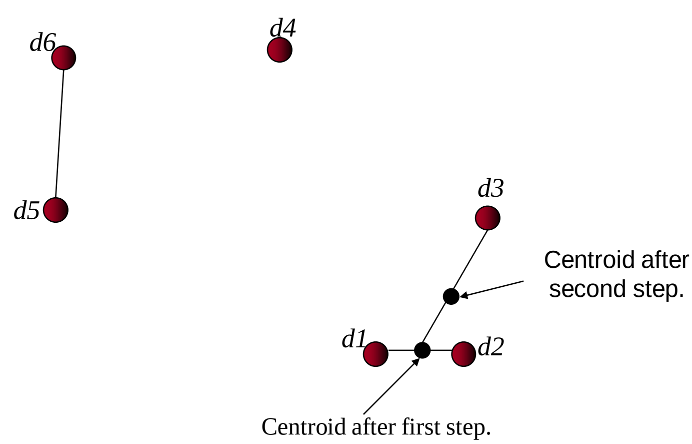
Computational Complexity

- In the first iteration, all HAC methods need to compute similarity of all pairs of n individual instances which is $O(n^2)$.
- In each of the subsequent n–2 merging iterations, it must compute the distance between the most recently created cluster and all other existing clusters.
  - Since we can just store unchanged similarities
- In order to maintain an overall $O(n^2)$ performance, computing similarity to each other cluster must be done in constant time.
  - Else $O(n^2 \log n)$ or $O(n^3)$ if done naively

Key notion: *cluster representative*

- We want a notion of a representative point in a cluster
- Representative should be some sort of "typical" or central point in the cluster, e.g.,
  - point inducing smallest radii to docs in cluster
  - smallest squared distances, etc.
  - point that is the "average" of all docs in the cluster
    - Centroid or center of gravity

Example: *n=6, k=3,* closest pair of centroids



Outliers in centroid computation

- Can ignore outliers when computing centroid.
- What is an outlier?
  - Lots of statistical definitions, e.g.

    Say 10.
  - *moment* of point to centroid > M × some cluster *moment*.

Group Average Agglomerative Clustering

- Use average similarity across all pairs within the merged cluster to measure the similarity of two clusters.

$$sim(c_i, c_j) = \frac{1}{|c_i \cup c_j|(|c_i \cup c_j| - 1)} \sum_{\vec{x} \in (c_i \cup c_j)} \sum_{\vec{y} \in (c_i \cup c_j): \vec{y} \neq \vec{x}} sim(\vec{x}, \vec{y})$$

- Compromise between single and complete link.
- Two options:
  - Averaged across all ordered pairs in the merged cluster
  - Averaged over all pairs between the two original clusters
- Some previous work has used one of these options; some the other. No clear difference in efficacy

Computing Group Average Similarity

- Assume cosine similarity and normalized vectors with unit length.
- Always maintain sum of vectors in each cluster.

$$\vec{s}(c_j) = \sum_{\vec{x} \in c_j} \vec{x}$$

- Compute similarity of clusters in constant time:

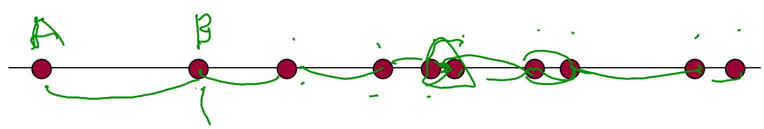
$$sim(c_i, c_j) = \frac{(\vec{s}(c_i) + \vec{s}(c_j)) \bullet (\vec{s}(c_i) + \vec{s}(c_j)) - (|c_i| + |c_j|)}{(|c_i| + |c_j|)(|c_i| + |c_j| - 1)}$$

Efficiency: Medoid As Cluster Representative

- The centroid does not have to be a document.
- Medoid: A cluster representative that is one of the documents
- For example: the document closest to the centroid
- One reason this is useful
  - Consider the representative of a large cluster (>1000 documents)
  - The centroid of this cluster will be a dense vector
  - The medoid of this cluster will be a sparse vector
- Compare: mean/centroid vs. median/medoid

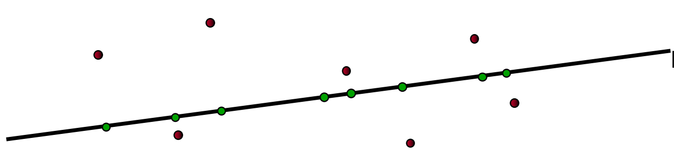
Exercise

- Consider agglomerative clustering on *n* points on a line. Explain how you could avoid n^3 distance computations - how many will your scheme use?



Efficiency: "Using approximations"

- In standard algorithm, must find closest pair of centroids at each step
- Approximation: instead, find nearly closest pair
  – use some data structure that makes this approximation easier to maintain
  – simplistic example: maintain closest pair based on distances in projection on a random line



Term vs. document space

- So far, we clustered docs based on their similarities in term space
- For some applications, e.g., topic analysis for inducing navigation structures, can “dualize”:
  - use docs as axes
  - represent (some) terms as vectors
  - proximity based on co-occurrence of terms in docs
  - now clustering terms, not docs

Term vs. document space

- Cosine computation
  - Constant for docs in term space
  - Grows linearly with corpus size for terms in doc space
- Cluster labeling
  - clusters have clean descriptions in terms of noun phrase co-occurrence
  - Easier labeling?
- Application of term clusters
  - Sometimes we want term clusters (example?)
  - If we need doc clusters, left with problem of binding docs to these clusters

Multi-lingual docs

- E.g., Canadian government docs.
- Every doc in English and equivalent French.
  - Must cluster by concepts rather than language
- Simplest: pad docs in one language with dictionary equivalents in the other
  - thus each doc has a representation in both languages
- Axes are terms in both languages

Feature selection

- Which terms to use as axes for vector space?
- Large body of (ongoing) research
- IDF is a form of feature selection
  - Can exaggerate noise e.g., mis-spellings
- Better is to use highest weight mid-frequency words – the most discriminating terms
- Pseudo-linguistic heuristics, e.g.,
  - drop stop-words
  - stemming/lemmatization
  - use only nouns/noun phrases
- Good clustering should “figure out” some of these

Major issue - labeling

- After clustering algorithm finds clusters - how can they be useful to the end user?
- Need pithy label for each cluster
  - In search results, say “Animal” or “Car” in the ***jaguar*** example.
  - In topic trees (Yahoo), need navigational cues.
    - Often done by hand, a posteriori.

How to Label Clusters

- Show titles of typical documents
  - Titles are easy to scan
  - Authors create them for quick scanning!
  - But you can only show a few titles which may not fully represent cluster
- Show words/phrases prominent in cluster
  - More likely to fully represent cluster
  - Use distinguishing words/phrases
    - Differential labeling
  - But harder to scan

Labeling

- Common heuristics - list 5-10 most frequent terms in the centroid vector.
  - Drop stop-words; stem.
- Differential labeling by frequent terms
  - Within a collection "Computers", clusters all have the word computer as frequent term.
  - Discriminant analysis of centroids.
- Perhaps better: distinctive noun phrase

PURDUE UNIVERSITY

CS54701: Information Retrieval

Text Clustering

31 March 2016

Prof. Chris Clifton

Borrows slides from Chris Manning, Ray Mooney and Soumen Chakrabarti

Indiana Center for Database Systems ™

Evaluation of clustering

- Perhaps the most substantive issue in data mining in general:
  - how do you measure goodness?
- Most measures focus on computational efficiency
  - Time and space
- For application of clustering to search:
  - Measure retrieval effectiveness

Approaches to evaluating

- Anecdotal
- User inspection
- Ground "truth" comparison
  - Cluster retrieval
- Purely quantitative measures
  - Probability of generating clusters found
  - Average distance between cluster members
- Microeconomic / utility

Anecdotal evaluation

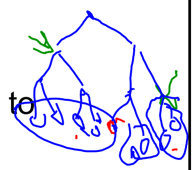
- Probably the commonest (and surely the easiest)
  - “I wrote this clustering algorithm and look what it found!”
- No benchmarks, no comparison possible
- Any clustering algorithm will pick up the easy stuff like partition by languages
- Generally, unclear scientific value.

User inspection

- Induce a set of clusters or a navigation tree
- Have subject matter experts evaluate the results and score them
  - some degree of subjectivity
- Often combined with search results clustering
- Not clear how reproducible across tests.
- Expensive / time-consuming

Ground “truth” comparison

- Take a union of docs from a taxonomy & cluster
  - Yahoo!, ODP, newspaper sections …
- Compare clustering results to baseline
  - e.g., 80% of the clusters found map “cleanly” to taxonomy nodes
  - How would we measure this?
- But is it the “right” answer?
  - There can be several equally right answers
- For the docs given, the static prior taxonomy may be incomplete/wrong in places
  - the clustering algorithm may have gotten right things not in the static taxonomy



“Subjective”

Ground truth comparison

- Divergent goals
- Static taxonomy designed to be the “right” navigation structure
  - somewhat independent of corpus at hand
- Clusters found have to do with vagaries of corpus
- Also, docs put in a taxonomy node may not be the most representative ones for that topic
  - cf Yahoo!

Microeconomic viewpoint

- Anything - including clustering - is only as good as the economic utility it provides
- For clustering: net economic gain produced by an approach (vs. another approach)
- Strive for a concrete optimization problem
- Examples
  - recommendation systems
  - clock time for interactive search
    - expensive

Evaluation example: Cluster retrieval

- Ad-hoc retrieval
- Cluster docs in returned set
- Identify best cluster & only retrieve docs from it
- How do various clustering methods affect the quality of what's retrieved?
- Concrete measure of quality:
  - Precision as measured by user judgements for these queries
- Done with TREC queries

Evaluation

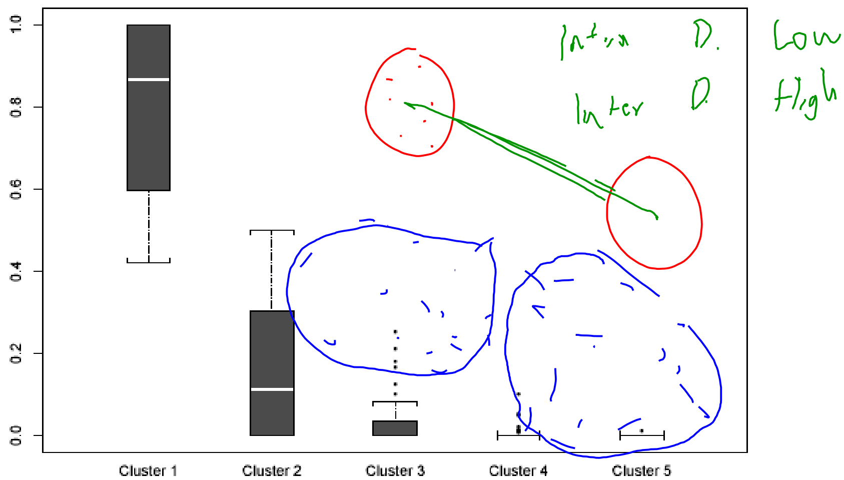
- Compare two IR algorithms
  - 1. send query, present ranked results
  - 2. send query, cluster results, present clusters
- Experiment was simulated (no users)
  - Results were clustered into 5 clusters
  - Clusters were ranked according to percentage relevant documents
  - Documents within clusters were ranked according to similarity to query

Sim-Ranked vs. Cluster-Ranked

	Precision at Cutoffs		
CutOff	Sim-Ranked	Cluster-Ranked	% Increase
5	.342	.428	.252
10	.314	.401	.277
20	.276	.363	.312

Table 4: Precision at small document cutoff levels for the one-step algorithm.

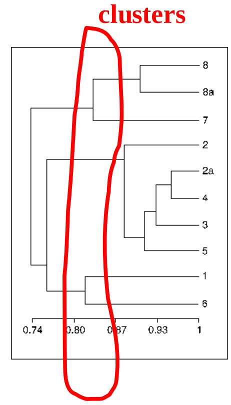
Relevance Density of Clusters



Buckshot Algorithm

- Another way to an efficient implementation:
  - Cluster a sample, then assign the entire set
- Buckshot combines HAC and K-Means clustering.
- First randomly take a sample of instances of size $\sqrt{n}$
- Run group-average HAC on this sample, which takes only O(*n*) time.
- Use the results of HAC as initial seeds for K-means.
- Overall algorithm is O(*n*) and avoids problems of bad seed selection.

Cut where You have k clusters



Uses HAC to bootstrap K-means

Bisecting K-means

- Divisive hierarchical clustering method using K-means
- For I=1 to k-1 do {
  - Pick a leaf cluster C to split
  - For J=1 to ITER do {
    - Use K-means to split C into two sub-clusters, C_1 and C_2
    - Choose the best of the above splits and make it permanent}

    }

  }

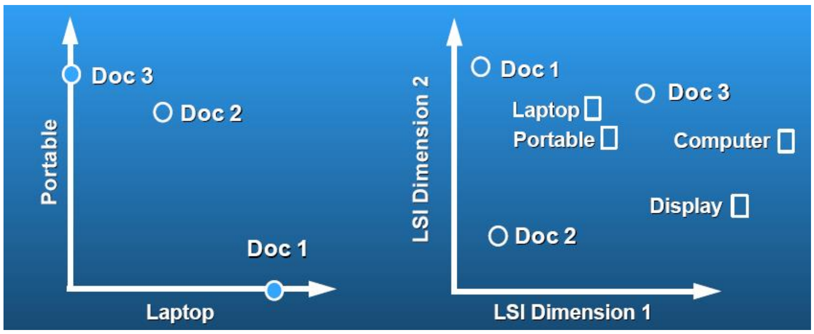
- Steinbach et al. suggest HAC is better than k-means but Bisecting K-means is better than HAC for their text experiments

Resources

- Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections (1992)
  - Cutting/Karger/Pedersen/Tukey
  - http://citeseer.ist.psu.edu/cutting92scattergather.html
- Data Clustering: A Review (1999)
  - Jain/Murty/Flynn
  - http://citeseer.ist.psu.edu/jain99data.html
- A Comparison of Document Clustering Techniques
  - Michael Steinbach, George Karypis and Vipin Kumar. TextMining Workshop. KDD. 2000.

Latent Semantic Analysis

- **Latent semantic space**: illustrative example



courtesy of Susan Dumais

Performing the maps

- Each row and column of *A* gets mapped into the *k*-dimensional LSI space, by the SVD.
- Claim – this is not only the mapping with the best (Frobenius error) approximation to *A*, but in fact *improves* retrieval.
- A query q is also mapped into this space, by

$$q_k = q^T U_k \Sigma_k^{-1}$$

  – Query NOT a sparse vector.

Empirical evidence

- Experiments on TREC 1/2/3 – Dumais
- Lanczos SVD code (available on netlib) due to Berry used in these expts
  - Running times of ~ one day on tens of thousands of docs
- Dimensions – various values 250-350 reported
  - (Under 200 reported unsatisfactory)
- Generally expect recall to improve – what about precision?

Empirical evidence

- Precision at or above median TREC precision
  - Top scorer on almost 20% of TREC topics
- Slightly better on average than straight vector spaces
- Effect of dimensionality:

Dimensions	Precision
250	0.367
300	0.371
346	0.374

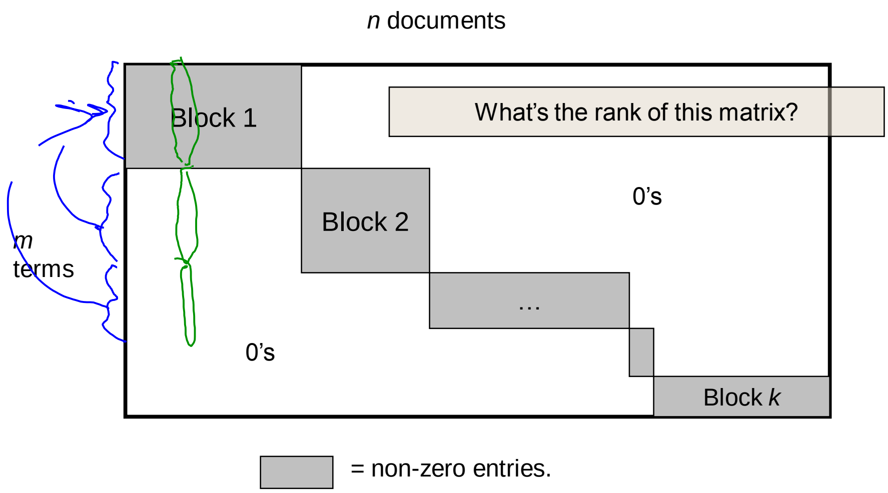
Failure modes

- Negated phrases
  - TREC topics sometimes negate certain query/terms phrases – automatic conversion of topics to
- Boolean queries
  - As usual, freetext/vector space syntax of LSI queries precludes (say) “Find any doc having to do with the following 5 companies”
- See Dumais for more.

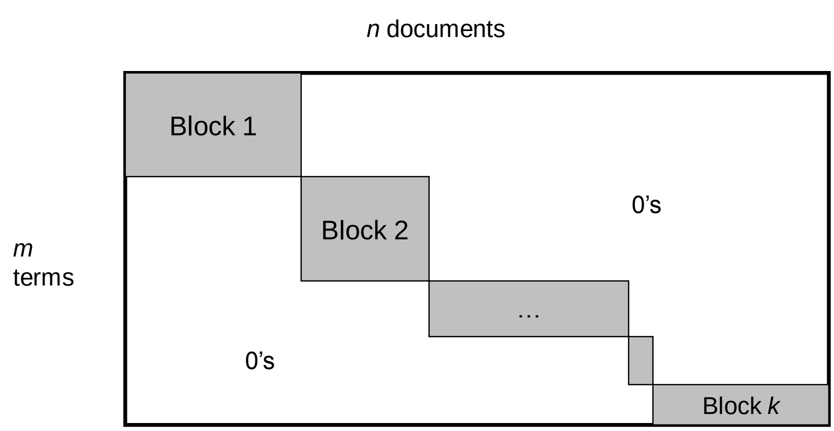
But why is this clustering?

- We’ve talked about docs, queries, retrieval and precision here.
- What does this have to do with clustering?
- Intuition: Dimension reduction through LSI brings together “related” axes in the vector space.

Intuition from block matrices

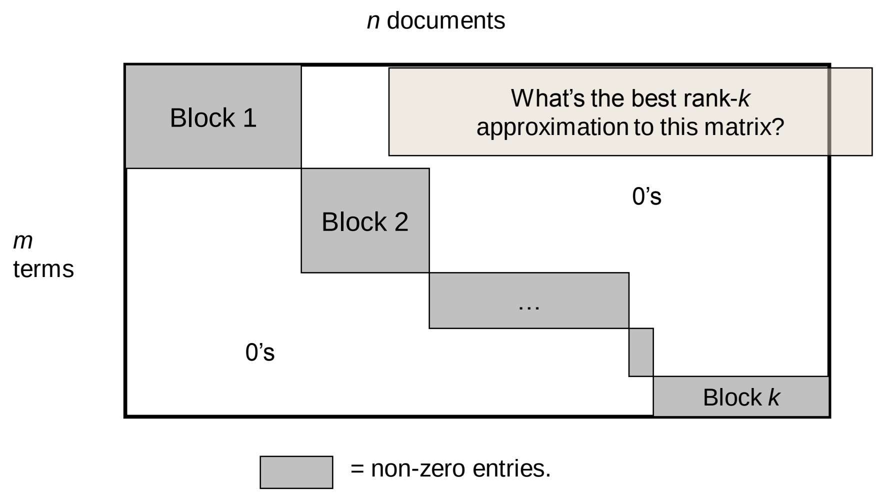


Intuition from block matrices

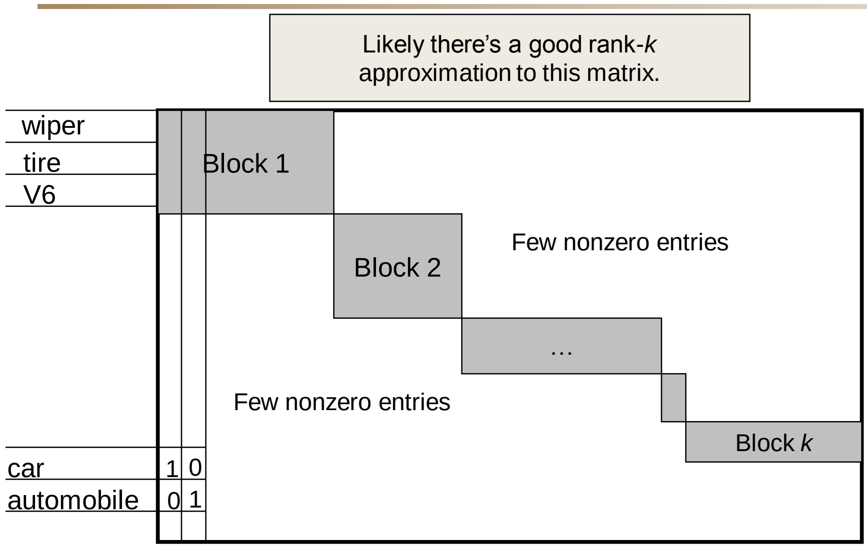


Vocabulary partitioned into *k* topics (clusters); each doc discusses only one topic.

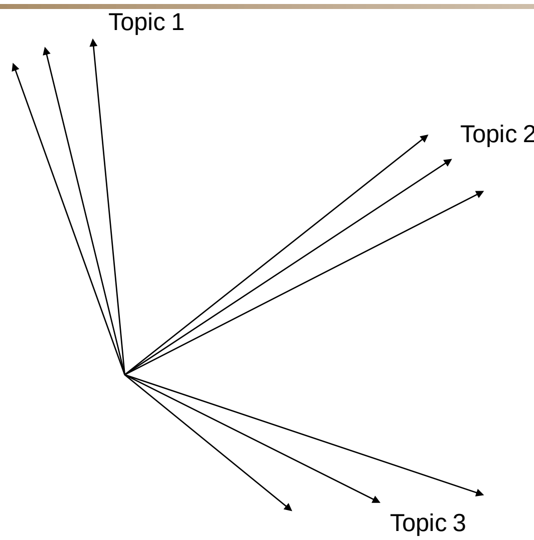
Intuition from block matrices



Intuition from block matrices



Simplistic picture



Some wild extrapolation

- The “dimensionality” of a corpus is the number of distinct topics represented in it.
- More mathematical wild extrapolation:
  – if A has a rank k approximation of low Frobenius error, then there are no more than k distinct topics in the corpus.

LSI has many other applications

- In many settings in pattern recognition and retrieval, we have a feature-object matrix.
  - For text, the terms are features and the docs are objects.
  - Could be opinions and users … more in 276B.
- This matrix may be redundant in dimensionality.
  - Can work with low-rank approximation.
  - If entries are missing (e.g., users' opinions), can recover if dimensionality is low.
- Powerful general analytical technique
  - Close, principled analog to clustering methods.

Resources

- http://www.cs.utk.edu/~berry/lsi++/
- http://lsi.argreenhouse.com/lsi/LSIpapers.html
- Dumais (1993) LSI meets TREC: A status report.
- Dumais (1994) Latent Semantic Indexing (LSI) and TREC-2.
- Dumais (1995) Using LSI for information filtering: TREC-3 experiments.
- M. Berry, S. Dumais and G. O'Brien. *Using linear algebra for intelligent information retrieval*. SIAM Review, 37(4):573--595, 1995.