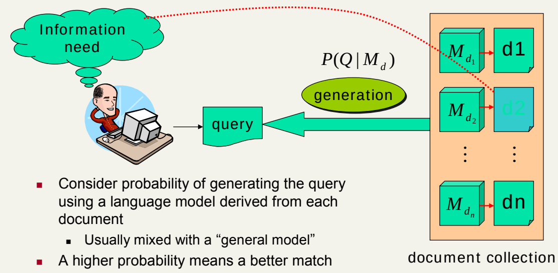
CS276A
Text Retrieval and Mining

Lecture 13

[Borrows slides from Ray Mooney and Soumen Chakrabarti]

Recap: The Language Model Approach to IR



- Consider probability of generating the query using a language model derived from each document
  - Usually mixed with a "general model"
- A higher probability means a better match
- Comparisons with vector space IR similarity

Today's Topic: Clustering 1

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

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

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
  - Effective "user recall" will be higher
- For speeding up vector space retrieval
  - Faster search

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

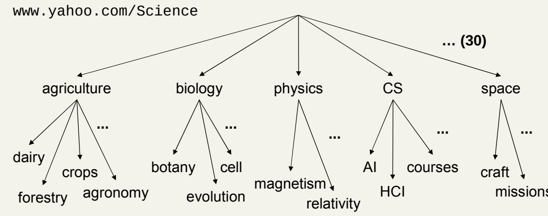
Index
Aardvark, 15
Blueberry, 200
Capricorn, 1, 45-55
Dog, 79-99
Egypt, 65
Falafel, 78-90
Giraffes, 45-59
…

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

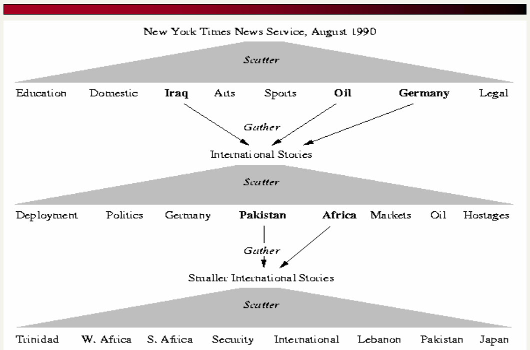
Corpus analysis/navigation

- 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

Yahoo! Hierarchy

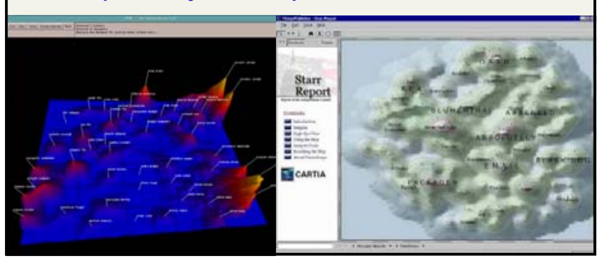


Scatter/Gather: Cutting, Karger, and Pedersen



For visualizing a document collection and its themes

- Wise et al, "Visualizing the non-visual" PNNL
- ThemeScapes, Cartia
  - [Mountain height = cluster size]



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.*

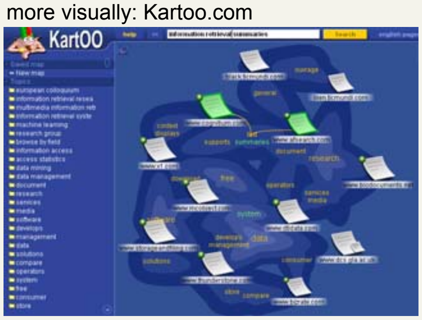
Why might this happen?

For better navigation of search results

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

For better navigation of search results

- And more visually: Kartoo.com



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 (say Jaguar, or ***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

For visualizing bookmarked pages

- Robertson, "Data Mountain" (Microsoft)

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

Exercise: Make up a simple example with points on a line in 2 clusters where this inexactness shows up.

Speeding up vector space retrieval

- Recall lecture 7 on leaders and followers
  - Effectively a fast simple clustering algorithm where documents are assigned to closest item in a set of randomly chosen leaders
- We could instead find natural clusters in the data
  - Cluster documents into *k* clusters
  - Retrieve closest cluster c_i to query
  - Rank documents in c_i and return to user

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

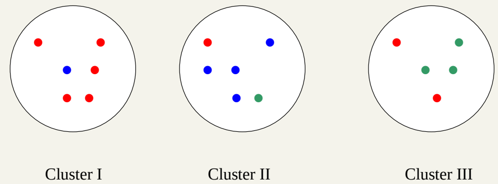
External Evaluation of Cluster Quality

- Assesses clustering with respect to ground truth
- Assume that there are C gold standard classes, while our clustering algorithms produce k clusters, $\pi_1, \pi_2, \ldots, \pi_k$ 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)

Purity



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

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.

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 similarity) between docs.
  - We will describe algorithms in terms of cosine similarity. Cosine similarity of normalized D_j, D_k:

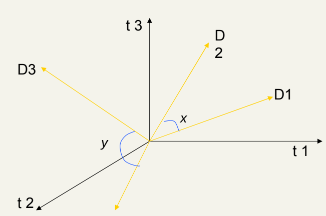
$$sim(D_j, D_k) = \sum_{i=1}^{m} w_{ij} \times w_{ik}$$

Aka normalized inner product.

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 axes - 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?

Intuition



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

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

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

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)

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*:

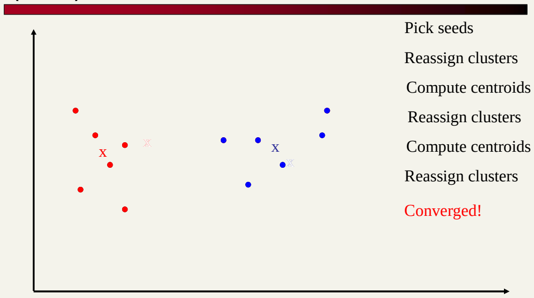
$$\vec{\mu}(c) = \frac{1}{|c|}\sum_{\vec{x} \in c} \vec{x}$$

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

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
        $s_j = \mu(c_j)$

K Means Example
(K=2)



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?

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 decreases 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:
  - $\Sigma -2(v_{in} - a) = 0$

Convergence of K-Means

- $\Sigma -2(v_{in} - a) = 0$
- $\Sigma v_{in} = \Sigma a$
- $m_k a = \Sigma v_{in}$
- $a = (1/ m_k) \Sigma v_{in} = c_{kn}$

- K-means typically converges quite quickly

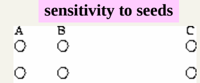
Time Complexity

- 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

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

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.

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]

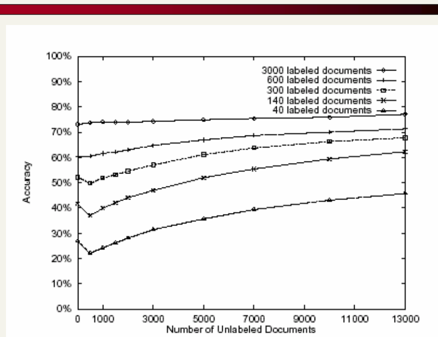
- Semi-supervised: some labeled and unlabeled data
- Take a completely labeled corpus D, and randomly select a subset as D_K.
- Also use the set $D^U \subseteq D$ of unlabeled documents in the EM procedure.
- Correct classification of a document
  => concealed class label = class with largest probability
- 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



Summary

- Two types of clustering
  - Flat, partional clustering
  - Hierarchical, agglomerative clustering
- How many clusters?
- Key issues
  - Representation of data points
  - Similarity/distance measure
- K-means: the basic partitional algorithm
- Model-based clustering and EM estimation