Today’s topic

• Clustering documents
Why cluster documents

• Given a corpus, partition it into groups of related docs
  – Recursively, can induce a tree of topics
• Given the set of docs from the results of a search (say *jaguar*), partition into groups of related docs
  – semantic disambiguation
Results list clustering example

• Cluster 1:
  • Jaguar Motor Cars’ home page
  • Mike’s XJS resource page
  • Vermont Jaguar owners’ club

• Cluster 2:
  • Big cats
  • My summer safari trip
  • Pictures of jaguars, leopards and lions

• Cluster 3:
  • Jacksonville Jaguars’ Home Page
  • AFC East Football Teams
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 distance
Recall doc as vector

- Each doc $j$ is a vector of $tf \times idf$ values, one component for each term.
- Can normalize to unit length.
- So we have a vector space
  - terms are axes
  - $n$ docs live in this space
  - even with stemming, may have 10000+ dimensions
Postulate: Documents that are “close together” in vector space talk about the same things.
Cosine similarity

Cosine similarity of $D_j, D_k$:

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

Aka normalized inner product.
Two flavors of clustering

• Given \( n \) docs and a positive integer \( k \), partition docs into \( k \) (disjoint) subsets.

• Given docs, partition into an “appropriate” number of subsets.
  – E.g., for query results - ideal value of \( k \) not known up front.

• Can usually take an algorithm for one flavor and convert to the other.
Cluster centroid

- **Centroid** of a cluster = average of vectors in a cluster - is a vector.
  - Need not be a doc.
- Centroid of (1,2,3); (4,5,6); (7,2,6) is (4,3,5).
Outliers in centroid computation

- Ignore outliers when computing centroid.
  - What is an outlier?
  - Distance to centroid > $M \times \text{average}$.

Say 10.
Agglomerative clustering

• Given target number of clusters $k$.
• Initially, each doc viewed as a cluster
  – start with $n$ clusters;
• Repeat:
  – while there are $> k$ clusters, find the “closest pair” of clusters and merge them.
“Closest pair” of clusters

- Many variants to defining closest pair of clusters.
- Closest pair \(\iff\) two clusters whose centroids are the most cosine-similar.
Example; $n=6$, $k=3$

Centroid after first step.
Issues

• Have to discover closest pairs
  – compare all pairs?
    • $n^3$ cosine similarity computations.
    • Avoid: recall techniques from lecture 4.
  – points are changing as centroids change.
• Changes at each step are not localized
  – on a large corpus, memory management becomes an issue.

How would you adapt sampling/pre-grouping?
Consider agglomerative clustering on $n$ points on a line. Explain how you could avoid $n^3$ distance computations - how many will your scheme use?
Hierarchical clustering

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

\[ d1, d2 \]
\[ d4, d5 \]
\[ d3 \]
\[ d3, d4, d5 \]
Different algorithm: $k$-means

- Iterative algorithm.
- More locality within each iteration.
- Hard to get good bounds on the number of iterations.
Basic iteration

• At the start of the iteration, we have $k$ centroids.
  – Need not be docs, just some $k$ points.
• 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.
Iteration example

- Docs
- Current centroids
Iteration example

- Docs
- New centroids
$k$-means clustering

- Begin with $k$ docs as centroids
  - could be any $k$ docs, but $k$ random docs are better.
- Repeat Basic Iteration until termination condition satisfied.
Termination conditions

- Several possibilities, e.g.,
  - A fixed number of iterations.
  - Centroid positions don’t change.

Does this mean that the docs in a cluster are unchanged?
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 *EM algorithm*.

- Under reasonable conditions, known to converge.
- Number of iterations could be large.
Exercise

• Consider running 2-means clustering on a corpus, each doc of which is from one of two different languages. What are the two clusters we would expect to see?

• Is agglomerative clustering likely to produce different results?
Multi-lingual docs

- Canadian/Belgian government docs.
- Every doc in English and equivalent French.
  - Cluster by concepts rather than language.
  - Cross-lingual retrieval.
\( k \) not specified in advance

- Say, the results of a query.
- Solve an optimization problem: penalize having lots of clusters
  - compressed summary of list of docs.
- Tradeoff between having more clusters (better focus within each cluster) and having too many clusters
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 cluster to be $= - $ Total Benefit - Total Cost.
- Find the clustering of highest Value, over all choices of $k$. 
Back to agglomerative clustering

- In a run of agglomerative clustering, we can try all values of $k=n,n-1,n-2, \ldots 1$.
- At each, we can measure our Value, then pick the best choice of $k$. 
Exercises

• Suppose a run of agglomerative clustering finds $k=7$ to have the highest Value amongst all $k$. Have we found the highest-Value clustering amongst all clusterings with $k=7$?
Using clustering in applications
Clustering to speed up scoring

- From Lecture 4, recall sampling and pre-grouping
  - Wanted to find, given a query $Q$, the nearest docs in the corpus
  - Wanted to avoid computing cosine similarity of $Q$ to each of $n$ docs in the corpus.
Sampling and pre-grouping
(Lecture 4)

• First run a pre-processing phase:
  – pick $\sqrt{n}$ docs at random: call these leaders
  – For each other doc, pre-compute nearest leader
    • Docs attached to a leader: its followers;
    • Likely: each leader has $\sim \sqrt{n}$ followers.

• Process a query as follows:
  – Given query $Q$, find its nearest leader $L$.
  – Seek nearest docs from among $L$’s followers.
Instead of random leaders, cluster

• First run a pre-processing phase:
  – Cluster docs into $\sqrt{n}$ clusters.
  – For each cluster, its centroid is the leader.

• Process a query as follows:
  – Given query $Q$, find its nearest leader $L$.
  – Seek nearest docs from among $L$’s followers.
Given a corpus, agglomerate into a hierarchy

- Throw away lower layers so you don’t have $n$ leaf topics each having a single doc.
Navigation structure

• Deciding how much to throw away needs human judgement.
• Can also induce hierarchy top-down - e.g., use $k$-means, then recur on the clusters.
• Topics induced by clustering need human ratification.
• Need to address issues like partitioning at the top level by language.
Major issue - labelling

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

• Common heuristics - list 5-10 most frequent terms in the centroid vector.
  – Drop stop-words; stem.

• Differential labeling by frequent terms
  – Within the cluster “Computers”, child clusters all have the word *computer* as frequent terms.
  – Discriminant analysis of centroids for peer clusters.
Supervised vs. unsupervised learning

• Unsupervised learning:
  – Given corpus, infer structure implicit in the docs, without prior training.

• Supervised learning:
  – Train system to recognize docs of a certain type (e.g., docs in Italian, or docs about religion)
  – Decide whether or not new docs belong to the class(es) trained on
Resources

• Good demo of results-list clustering:
  \texttt{cluster.cs.yale.edu}