CS276B

Text Information Retrieval, Mining, and Exploitation

Lecture 2

Recap: Why cluster documents?

- For improving recall in search applications
- For speeding up vector space retrieval
- Corpus analysis/navigation
  - Sense disambiguation in search results

Recap: 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/features
  - *n* docs live in this space
  - even with stemming, may have 10000+ dimensions
  - do we really want to use all terms?

Recap: 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 - though UI may impose limits.
- Can usually take an algorithm for one flavor and convert to the other.

Today's topics

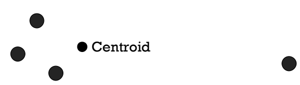
- Top-down and bottom-up clustering algorithms
- Issues peculiar to text

Key notion: *cluster representative*

- In the algorithms to follow, will generally need 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
- Need not be a document

Key notion: cluster centroid

- Centroid of a cluster = component-wise 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)

- Can ignore outliers when computing centroid.
- What is an outlier?
  - Lots of statistical definitions, e.g.
  - *moment* of point to centroid > M × some cluster *moment*.

Say 10.

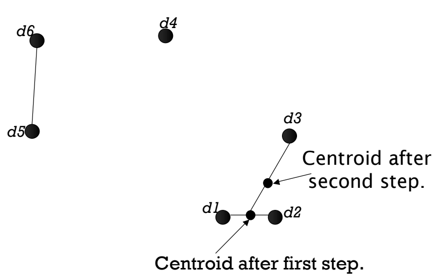
Agglomerative clustering

- 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
  - Clusters whose centroids are the most cosine-similar
  - … whose "closest" points are the most cosine-similar
  - … whose "furthest" points are the most cosine-similar

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



Issues

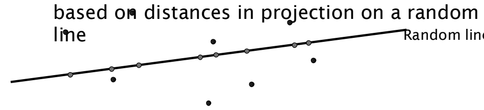
- Have to support finding closest pairs continually
  - compare all pairs? Why?
    - What's this? Potentially n^3 cosine similarity computations
    - To avoid: use approximations.
  - "points" are changing as centroids change.
- Changes at each step are not localized
  - on a large corpus, memory management an issue
  - sometimes addressed by clustering a sample.

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?

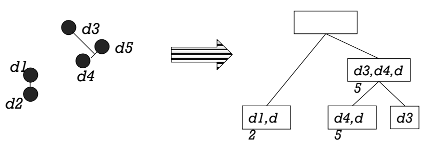
"Using approximations"

- 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



Hierarchical clustering

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



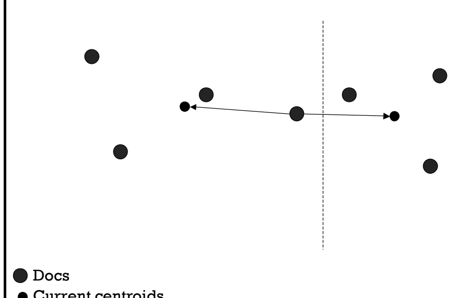
Different algorithm: *k*-means

- Given k - the number of clusters desired.
- Iterative algorithm.
- More locality within each iteration.
- Hard to get good bounds on the number of iterations.

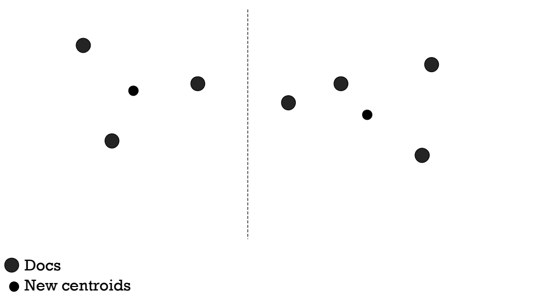
Basic iteration

- At the start of the iteration, we have k centroids.
- 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



Iteration example



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.
  - 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 *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?

k not specified in advance

- Say, the results of a query.
- Solve an optimization problem: penalize having lots of clusters
  - application dependant, 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 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, … 1.*
- At each, we can measure our Value, then pick the best choice of *k*.

Exercise

- 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*?

Text clustering issues and applications

List of issues/applications covered

- Term vs. document space clustering
- Multi-lingual docs
- Feature selection
- Speeding up scoring
- Building navigation structures
  - "Automatic taxonomy induction"
- Labeling

Term vs. document space

- Thus far, we clustered docs based on their similarities in terms 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

- If terms carefully chosen (say nouns)
  - fixed number of pairs for distance computation
    - independent of corpus size
  - clusters have clean descriptions in terms of noun phrase co-occurrence - easier labeling?
  - 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 lang 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?
- Huge body of (ongoing) research
- IDF is a form of feature selection
  - can exaggerate noise e.g., mis-spellings
- Pseudo-linguistic heuristics, e.g.,
  - drop stop-words
  - stemming/lemmatization
  - use only nouns/noun phrases
- Good clustering should "figure out" some of these

Clustering to speed up scoring

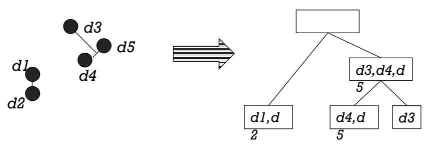
- From CS276a, recall sampling and pre-grouping
  - Wanted to find, given a query *Q*, the nearest docs in the corpus
  - Wanted to avoid computing cosine similarity of *Q* to each of n docs in the corpus.

Sampling and pre-grouping

- First run a clustering phase
  - pick a representative *leader* for each cluster.
- Process a query as follows:
  - Given query *Q*, find its nearest *leader L*.
  - Seek nearest docs from *L*'s followers only
    - avoid cosine similarity to all docs.

Navigation structure

- Given a corpus, agglomerate into a hierarchy
- Throw away lower layers so you don't have *n* leaf topics each having a single doc
  - Many principled methods for this pruning such as MDL.



Navigation structure

- Can also induce hierarchy top-down - e.g., use *k*-means, then recur on the clusters.
  - Need to figure out what *k* should be at each point.
- Topics induced by clustering need human ratification
  - can override mechanical pruning.
- Need to address issues like partitioning at the top level by language.

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 "Football" or "Car" in the ***jaguar*** example.
  - In topic trees, need navigational cues.
    - Often done by hand, a posteriori.

From 276a: 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
  - 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 the cluster "Computers", child clusters all have the word ***computer*** as frequent terms.
  - Discriminant analysis of sub-tree centroids.

The biggest issue in clustering?

- How do you compare two alternatives?
- Computation (time/space) is only one metric of performance
- How do you look at the "goodness" of the clustering produced by a method
- Next time ...

Resources

- Initialization of iterative refinement clustering algorithms. (1998)
  - Fayyad, Reina, and Bradley
  - http://citeseer.nj.nec.com/fayyad98initialization.html
- Scaling Clustering Algorithms to Large Databases (1998)
  - Bradley, Fayyad, and Reina
  - http://citeseer.nj.nec.com/bradley98scaling.html