CS276B

Text Information Retrieval, Mining, and Exploitation

Lecture 1

Jan 7 2003

Restaurant recommendations

- We have a list of all Palo Alto restaurants
  - with ↑ and ↓ ratings for *some*
  - as provided by *some* Stanford students
- Which restaurant(s) should I recommend to you?

Input

Alice	Il Fornaio	Yes
Bob	Ming's	No
Cindy	Straits Café	No
Dave	Ming's	Yes
Alice	Straits Café	No
Estie	Zao	Yes
Cindy	Zao	No
Dave	Brahma Bull	No
Dave	Zao	Yes
Estie	Ming's	Yes
Fred	Brahma Bull	No
Alice	Mango Café	No
Fred	Ramona's	No
Dave	Homma's	Yes
Bob	Higashi West	Yes
Estie	Straits Café	Yes

Algorithm 0

- Recommend to you the most popular restaurants
  - say # positive votes minus # negative votes
- Ignores your culinary preferences
  - *And* judgements of those with similar preferences
- How can we exploit the wisdom of "like-minded" people?

Another look at the input - a matrix

	Brahma Bull	Higashi West	Mango	Il Fornaio	Zao	Ming's	Ramona's	Straits	Homma's
Alice		Yes	No	Yes				No	
Bob		Yes				No		No	
Cindy				Yes	No			No	
Dave	No			No	Yes	Yes			Yes
Estie				No	Yes	Yes		Yes	
Fred	No						No		

Now that we have a matrix

	Brahma Bull	Higashi West	Mango	Il Fornaio	Zao	Ming's	Ramona's	Straits	Homma's
Alice		1	-1	1				-1	
Bob		1				-1		-1	
Cindy				1	-1			-1	
Dave	-1			-1	1	1			1
Estie				-1	1	1		1	
Fred	-1						-1		

View all other entries as zeros for now.

Similarity between two people

- Similarity between their preference vectors.
- Inner products are a good start.
- Dave has similarity 3 with Estie
  - but -2 with Cindy.
- Perhaps recommend Straits Cafe to Dave
  - and Il Fornaio to Bob, etc.

Algorithm 1.1

- You give me your preferences and I need to give you a recommendation.
- I find the person "most similar" to you in my database and recommend something he likes.
- Aspects to consider:
  - No attempt to discern cuisines, etc.
  - What if you've been to all the restaurants he has?
  - Do you want to rely on one person's opinions?

Algorithm 1.*k*

- You give me your preferences and I need to give you a recommendation.
- I find the *k* people "most similar" to you in my database and recommend what's most popular amongst them.
- Issues:
  - A priori unclear what *k* should be
  - Risks being influenced by "unlike minds"

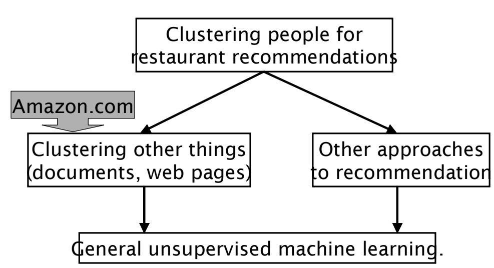
Slightly more sophisticated attempt

- Group similar users together into *clusters*
- You give your preferences and seek a recommendation, then
  - Find the "nearest cluster" (what's this?)
  - Recommend the restaurants most popular in this cluster
- Features:
  - avoids data sparsity issues
  - still no attempt to discern why you're recommended what you're recommended
  - how do you cluster?

How do you cluster?

- Must keep similar people together in a cluster
- Separate dissimilar people
- Factors:
  - Need a notion of similarity/distance
  - Vector space? Normalization?
  - How many clusters?
    - Fixed a priori?
    - Completely data driven?
  - Avoid "trivial" clusters - too large or small

Looking beyond



Why cluster documents?

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

Improving search recall

- *Cluster hypothesis* - Documents with similar text are related
- Ergo, 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: docs containing *automobile* returned on a query for *car* because
  - clustering grouped together docs containing *car* with those containing *automobile.*

Why might this happen?

Speeding up vector space retrieval

- In vector space retrieval, must find nearest doc vectors to query vector
- This would entail finding the similarity of the query to every doc - slow!
- 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.

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 home in on information
  - Crucial need: meaningful labels for topic nodes.
- Screenshot.

Navigating search results

- Given the results of a search (say ***jaguar***), partition into groups of related docs
  - sense disambiguation
- See for instance vivisimo.com

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

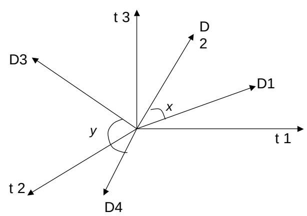
What makes docs "related"?

- 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 <u>similarity</u>) between docs.
  - We will describe algorithms in terms of cosine similarity.

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 10000+ dimensions
  - do we really want to use all terms?

Intuition



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

Cosine similarity

Cosine similarity of D_j, D_k:

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

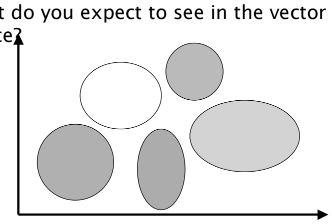
Aka normalized inner product.

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

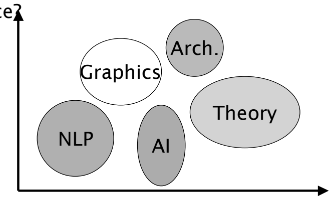
Thought experiment

- Consider clustering a large set of computer science documents
  - what do you expect to see in the vector space?



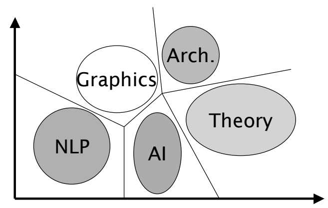
Thought experiment

- Consider clustering a large set of computer science documents
  - what do you expect to see in the vector space?



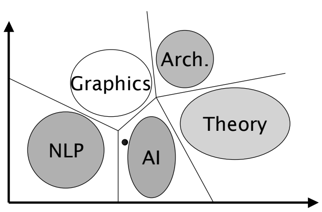
Decision boundaries

- Could we use these blobs to infer the subject of a new document?



Deciding what a new doc is about

- Check which region the new doc falls into
  - can output "softer" decisions as well.



• = AI

Setup

- Given "training" docs for each category
  - Theory, AI, NLP, etc.
- Cast them into a <u>decision space</u>
  - generally a vector space with each doc viewed as a bag of words
- Build a classifier that will classify new docs
  - Essentially, partition the decision space
- Given a new doc, figure out which partition it falls into

Supervised vs. unsupervised learning

- This setup is called *supervised learning* in the terminology of Machine Learning
- In the domain of text, various names
  - Text classification, text categorization
  - Document classification/categorization
  - "Automatic" categorization
  - Routing, filtering …
- In contrast, the earlier setting of clustering is called *unsupervised learning*
  - Presumes no availability of training samples
  - Clusters output may not be thematically unified.

"Which is better?"

- Depends
  - on your setting
  - on your application
- Can use in combination
  - Analyze a corpus using clustering
  - Hand-tweak the clusters and label them
  - Use clusters as training input for classification
  - Subsequent docs get classified
- Computationally, methods quite different

What more can these methods do?

- Assigning a category label to a document is one way of adding structure to it.
- Can add others, e.g.,extract from the doc
  - people
  - places
  - dates
  - organizations …
- This process is known as *information extraction*
  - can also be addressed using supervised learning.

Information extraction - methods

- Simple dictionary matching
- Supervised learning
  - e.g., train using URL's of universities
  - classifier learns that the portion before *.edu* is likely to be the University name.
- Regular expressions
  - Dates, prices
- Grammars
  - Addresses
- Domain knowledge
  - Resume/invoice field extraction

Information extraction - why

- Adding structure to unstructured/semi-structured documents
- Enable more structured queries without imposing strict semantics on document creation - why?
  - distributed authorship
  - legacy
- Enable "mining"

Course preview

- Document Clustering:
- Next time:
  - algorithms for clustering
  - term vs. document space
  - hierarchical clustering
  - labeling
- Jan 16: finish up document clustering
  - some implementation aspects for text
  - link-based clustering on the web

Course preview

- Text classification
  - Features for text classification
  - Algorithms for decision surfaces
- Information extraction
- More text classification methods
  - incl link analysis
- Recommendation systems
  - Voting algorithms
  - Matrix reconstruction
  - Applications to expert location

Course preview

- Text mining
  - Ontologies for information extraction
  - Topic detection/tracking
  - Document summarization
  - Question answering
- Bio-informatics
  - IR with textual and non-textual data
  - Gene functions; gene-drug interactions

Course administrivia

- Course URL: http://www.stanford.edu/class/cs276b/
- Grading:
  - 20% from midterm
  - 40% from final
  - 40% from project.

Course staff

- **Professor:** Christopher Manning
  Office: Gates 418
  manning@cs.stanford.edu
- **Professor:** Prabhakar Raghavan

  pragh@db.stanford.edu
- **Professor:** Hinrich Schütze
  schuetze@csli.stanford.edu
- Office Hours: F 10-12
- **TA: Teg Grenager**
  Office:
  Office Hours:
  grenager@cs.stanford.edu

Course Project

- This quarter we're doing a structured project
  - The whole class will work on a system to search/cluster/classify/extract/mine research papers
    - Citeseer on uppers [http://citeseer.com/]
  - This domain provides opportunities for exploring almost all the topics of the course:
    - text classification, clustering, information extraction, linkage algorithms, collaborative filtering, textbase visualization, text mining
  - … as well as opportunities to learn about building a large real working system

Course Project

- Two halves:
  - In first half (divided into two phases), people will build basic components, infrastructure, and data sets/databases for project
  - Second half: student-designed project related to goals of this project
- In general, work in groups of 2 on projects
- Reuse existing code where available
  - Lucene IR, ps/pdf to text converters, …
- 40% of the grade (distributed over phases)
- Watch for more details in Tue 14 Jan lecture

Resources

- Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections (1992)
  - Cutting/Karger/Pederesen/Tukey
  - http://citeseer.nj.nec.com/cutting92scattergather.html
- Data Clustering: A Review (1999)
  - Jain/Murty/Flynn
  - http://citeseer.nj.nec.com/jain99data.html