Text Analytics (Text Mining)

Concepts and Algorithms

Duen Horng (Polo) Chau
Georgia Tech

Some lectures are partly based on materials by
Professors Guy Lebanon, Jeffrey Heer, John Stasko, Christos Faloutsos, Le Song

Text is everywhere

We use documents as primary information artifact in our lives

Our access to documents has grown tremendously thanks to the Internet

- **WWW**: webpages, Twitter, Facebook, Wikipedia, Blogs, ...
- **Digital libraries**: Google books, ACM, IEEE, ...
- Lyrics, closed caption... (youtube)
- Police case reports
- legislation (law)
- reviews (products, rotten tomatoes)
- medical reports (EHR - electronic health records)
- job descriptions

Big (Research) Questions

... in understanding and gathering information from text and document collections

- establish authorship, authenticity; plagiarism detection
- finding patterns in human genome
- classification of genres for narratives (e.g., books, articles)
- tone classification; sentiment analysis (online reviews, twitter, social media)
- code: syntax analysis (e.g., find common bugs from students' answers)

Outline

- Storage (full text storage and full text search in SQLite, MySQL)
- **Preprocessing** (e.g., stemming, remove stop words)
- **Document representation** (most common: bag-of-words model)
- **Word importance** (e.g., word count, TF-IDF)
- Word disambiguation/entity resolution
- Document importance (e.g., PageRank)
- Document similarity (e.g., cosine similarity, Apolo/Belief Propagation, etc.)
- **Retrieval (Latent Semantic Indexing)**

To learn more:
Prof. Jacob Eisenstein's CS 4650/7650 Natural Language Processing

Stemming

Reduce words to their **stems** (or base forms)

Words: compute, computing, computer, ...

Stem: comput

Several classes of algorithms to do this:

- Stripping suffixes, lookup-based, etc.

http://en.wikipedia.org/wiki/Stemming
Stop words: http://en.wikipedia.org/wiki/Stop_words

Bags-of-words model

Represent each **document** as a **bag of words**, ignoring words' ordering. Why?

- Unstructured text -> a vector of numbers
- e.g., docs: "I like visualization", "I like data".
  - "I": 1,
  - "like": 2,
  - "data": 3,
  - "visualization": 4
- "I like visualization" -> [1, 1, 0, 1]
- "I like data" -> [1, 1, 1, 0]

TF-IDF

(**a word's importance score** in a **document**, among **N documents**)

When to use it? Everywhere you use "word count", you may use TF-IDF.

- **TF**: term frequency
  = #appearance a document
- **IDF**: inverse document frequency
  = log(**N** / #document containing that term)
- **Score = TF * IDF**
  (higher score -> more important)

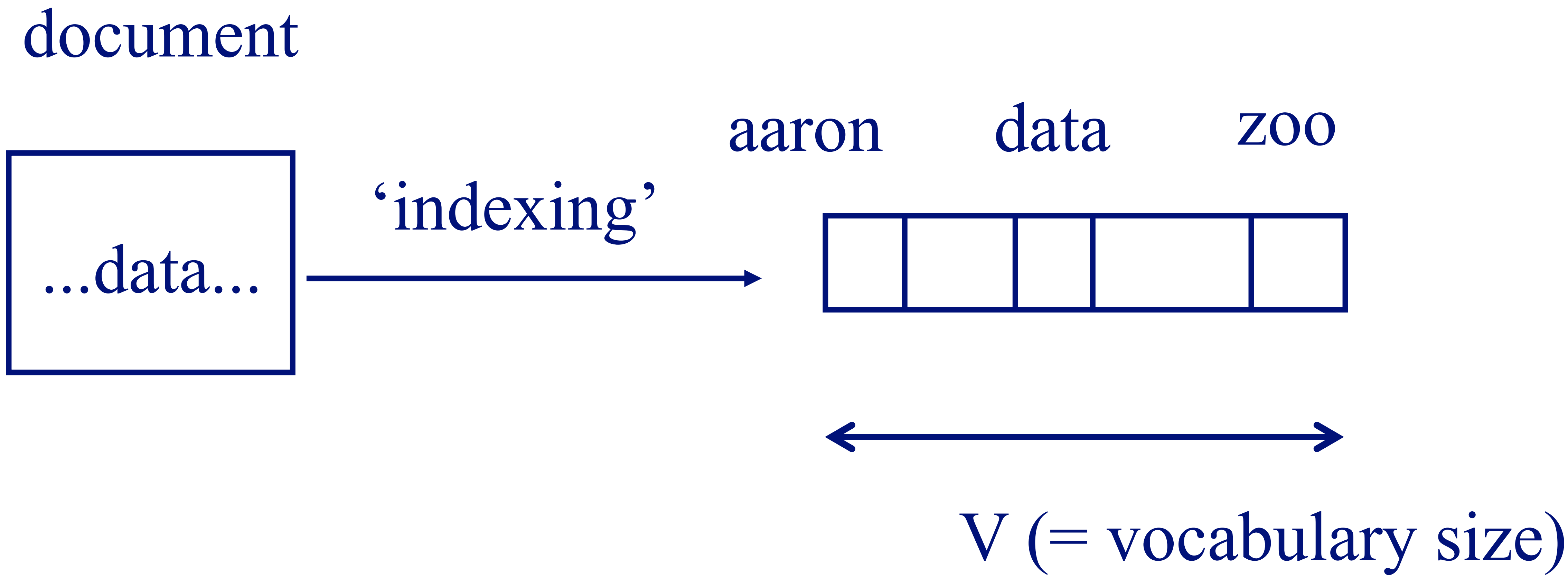
Example: http://en.wikipedia.org/wiki/Tf–idf#Example_of_tf.E2.80.93idf

Vector Space Model and Clustering

- keyword queries (vs Boolean)
- each document: -> vector (HOW?)
- each query: -> vector
- search for ‘similar’ vectors

Vector Space Model and Clustering

- main idea:



Vector Space Model and Clustering

Then, group nearby vectors together

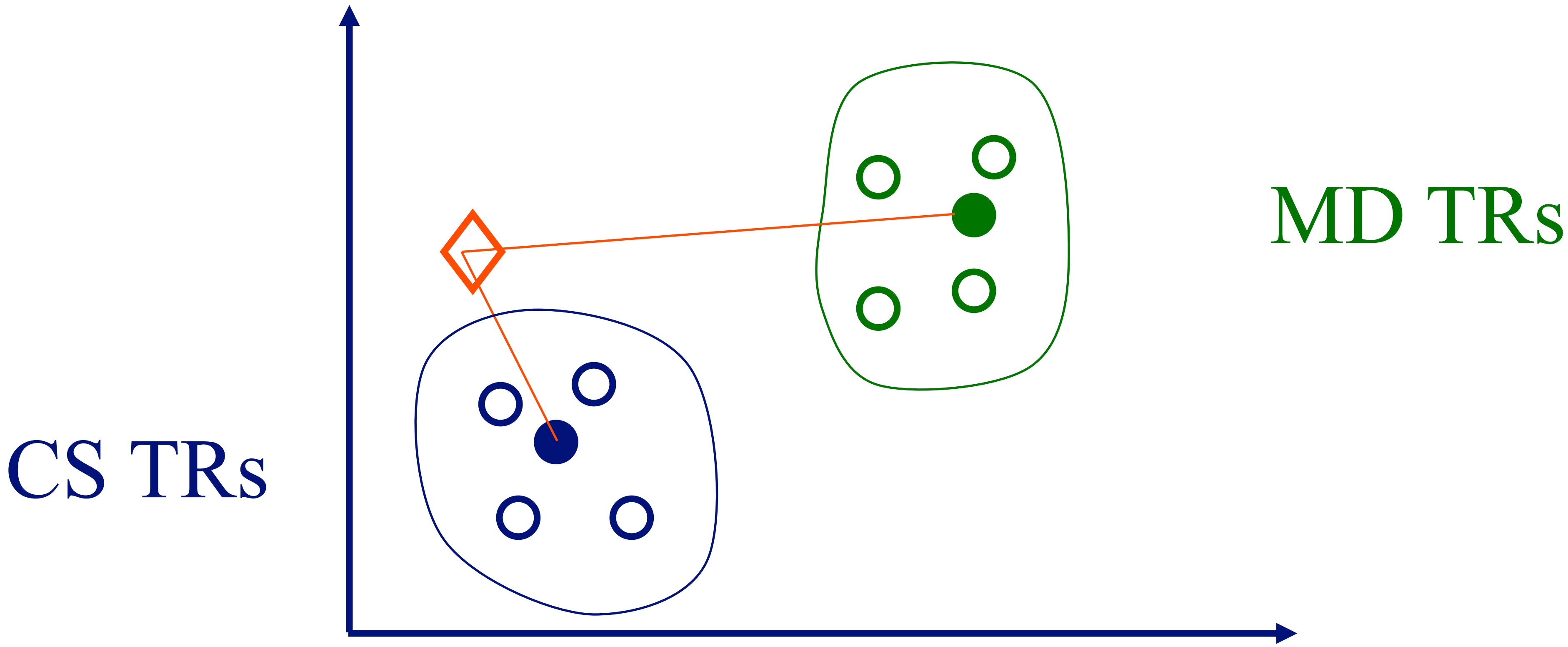
- Q1: cluster search?
- Q2: cluster generation?

Two significant contributions

- ranked output
- relevance feedback

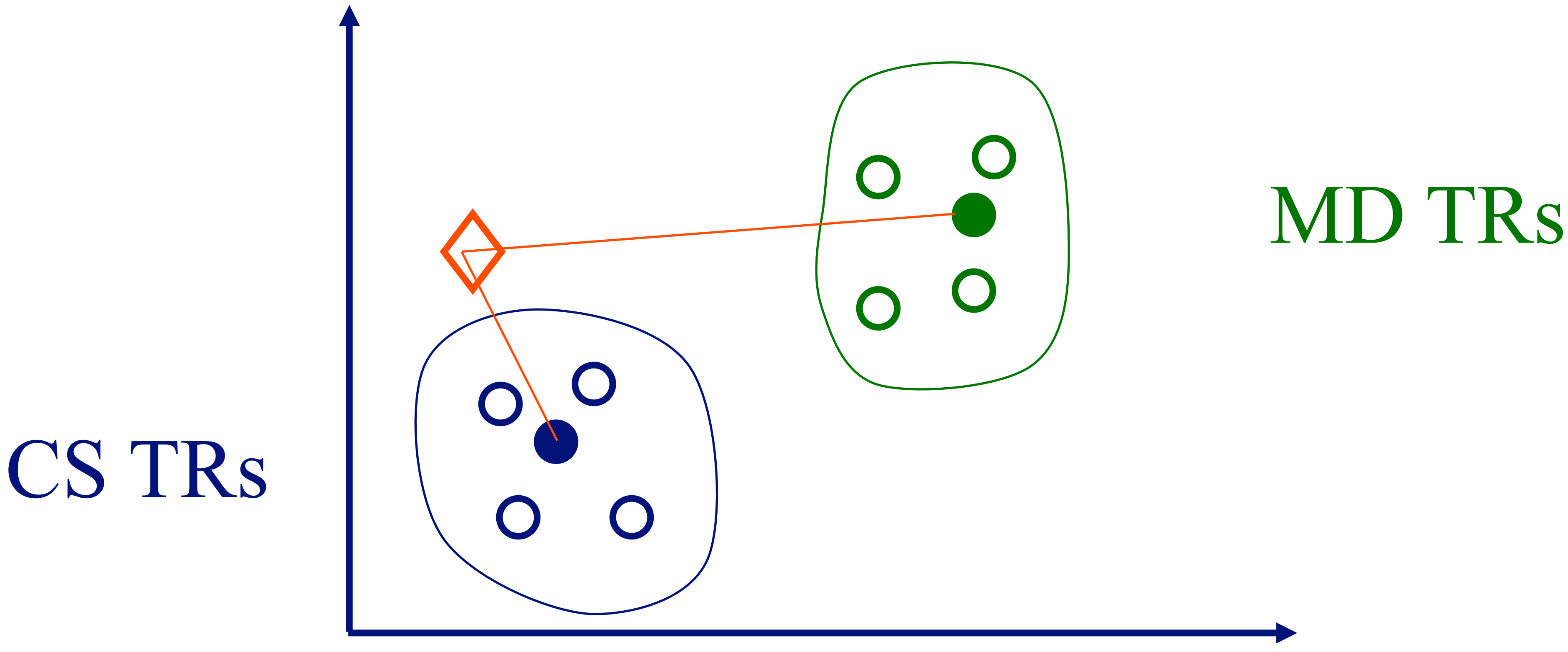
Vector Space Model and Clustering

- cluster search: visit the (k) closest superclusters; continue recursively



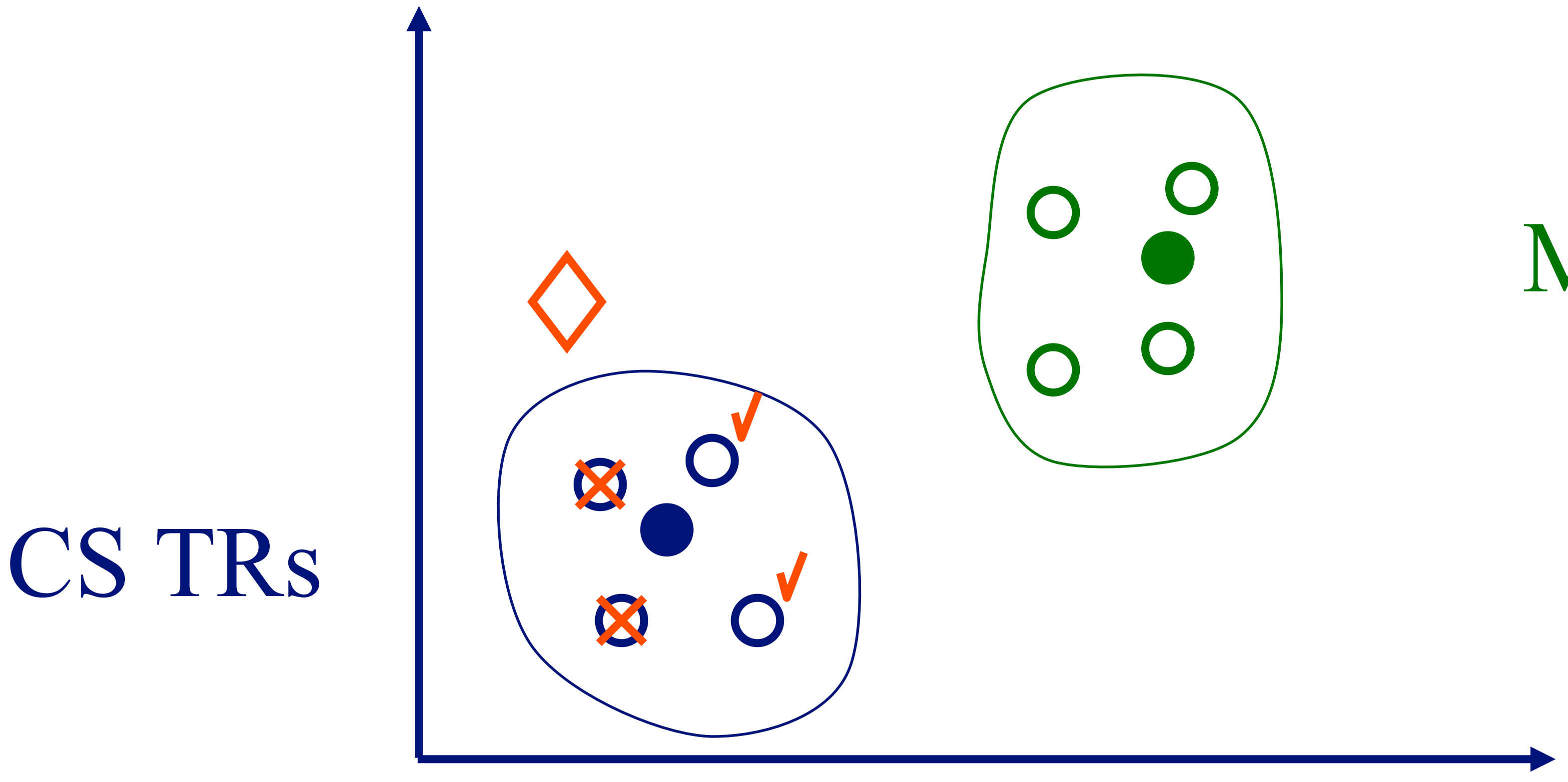
Vector Space Model and Clustering

- ranked output: easy!



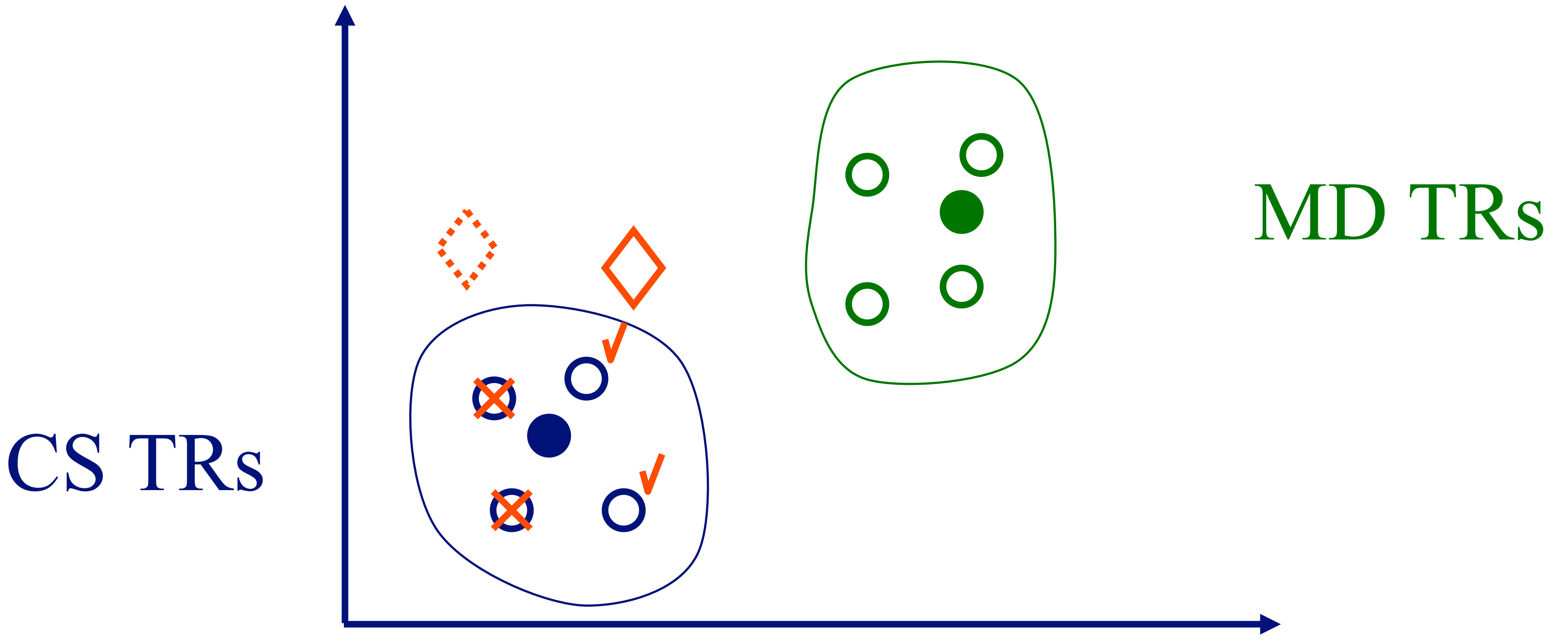
Vector Space Model and Clustering

- relevance feedback (brilliant idea) [Roccio'73]



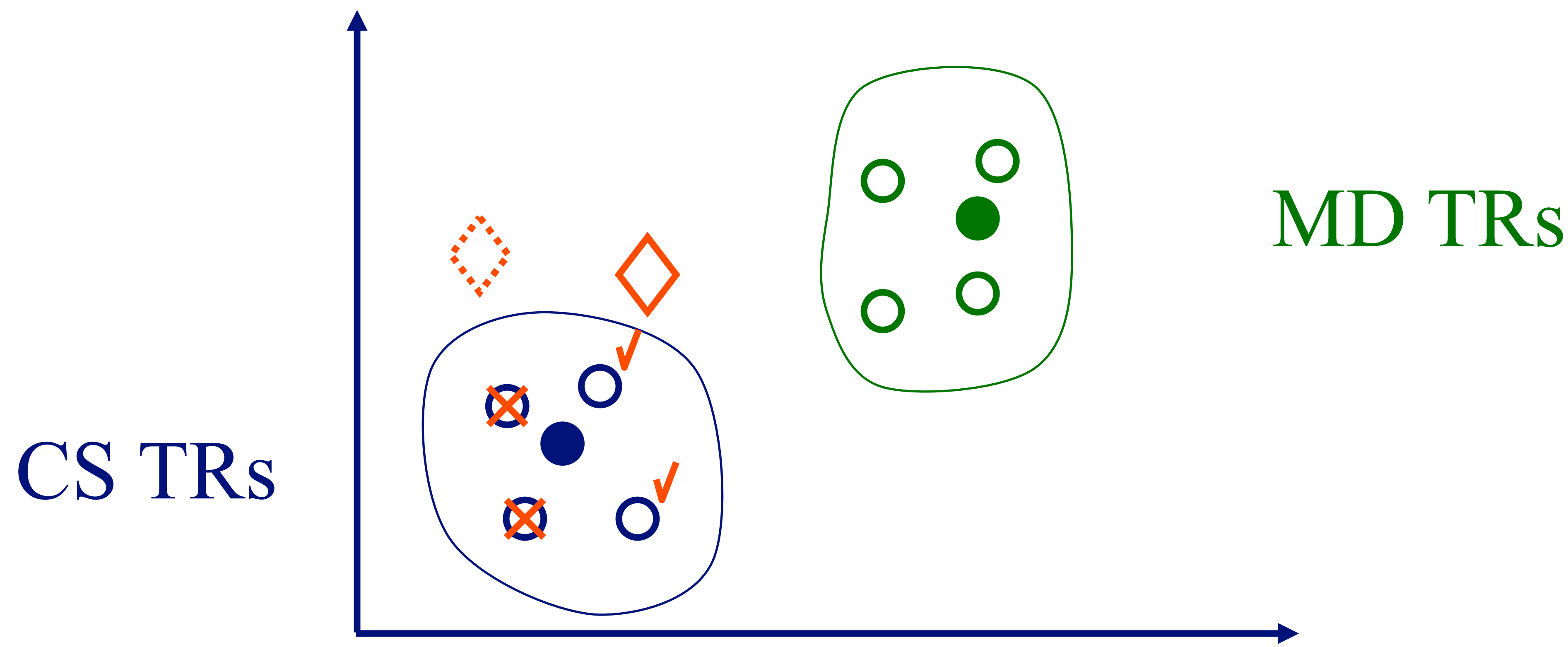
Vector Space Model and Clustering

- relevance feedback (brilliant idea) [Roccio'73]
- How?



Vector Space Model and Clustering

- How? A: by adding the 'good' vectors and subtracting the 'bad' ones

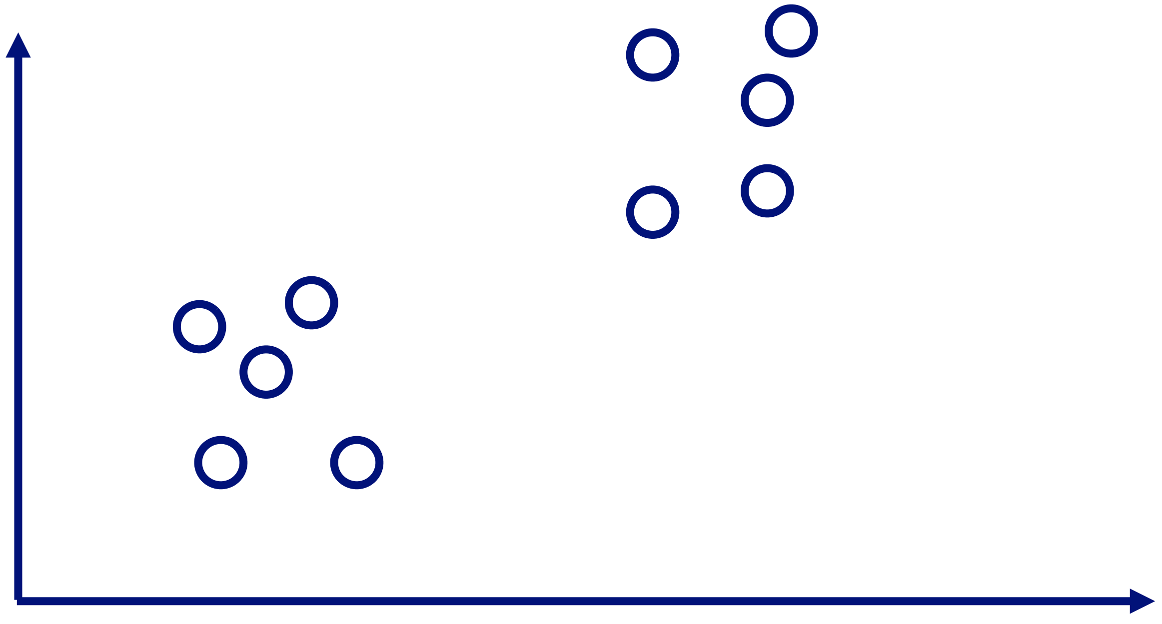


Outline - detailed

- main idea
- cluster search
- cluster generation
- evaluation

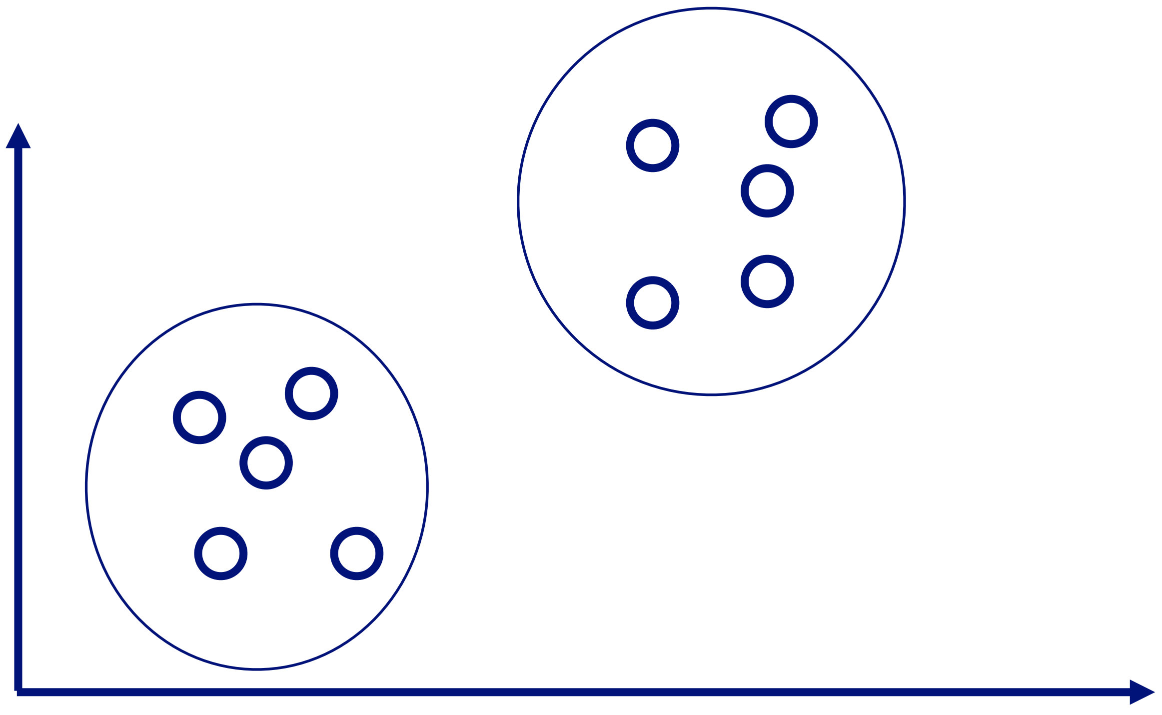
Cluster generation

- Problem:
  - given N points in V dimensions,
  - group them



Cluster generation

- Problem:
  - given N points in V dimensions,
  - group them



Cluster generation

We need

- Q1: document-to-document similarity
- Q2: document-to-cluster similarity

Cluster generation

Q1: document-to-document similarity

(recall: ‘bag of words’ representation)

- D1: {‘data’, ‘retrieval’, ‘system’}
- D2: {‘lung’, ‘pulmonary’, ‘system’}
- distance/similarity functions?

Cluster generation

A1: # of words in common

A2: .... normalized by the vocabulary sizes

A3: .... etc

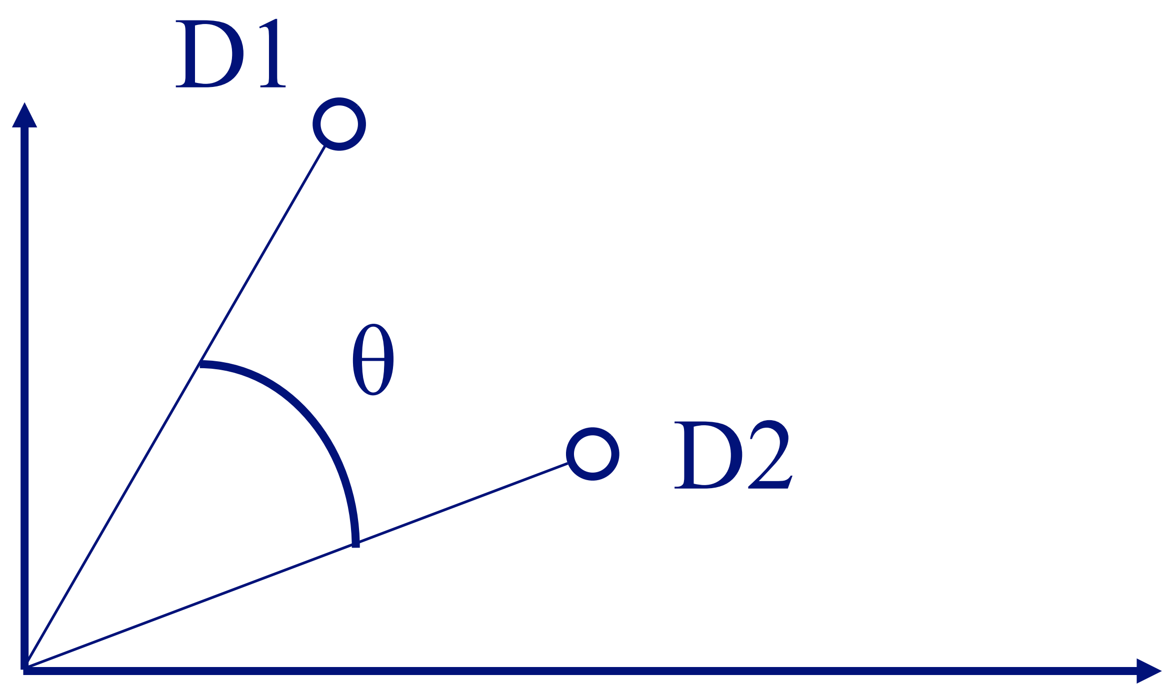
About the same performance - prevailing one: cosine similarity

Cluster generation

cosine similarity:

$$\text{similarity}(D1, D2) = \cos(\theta) =$$

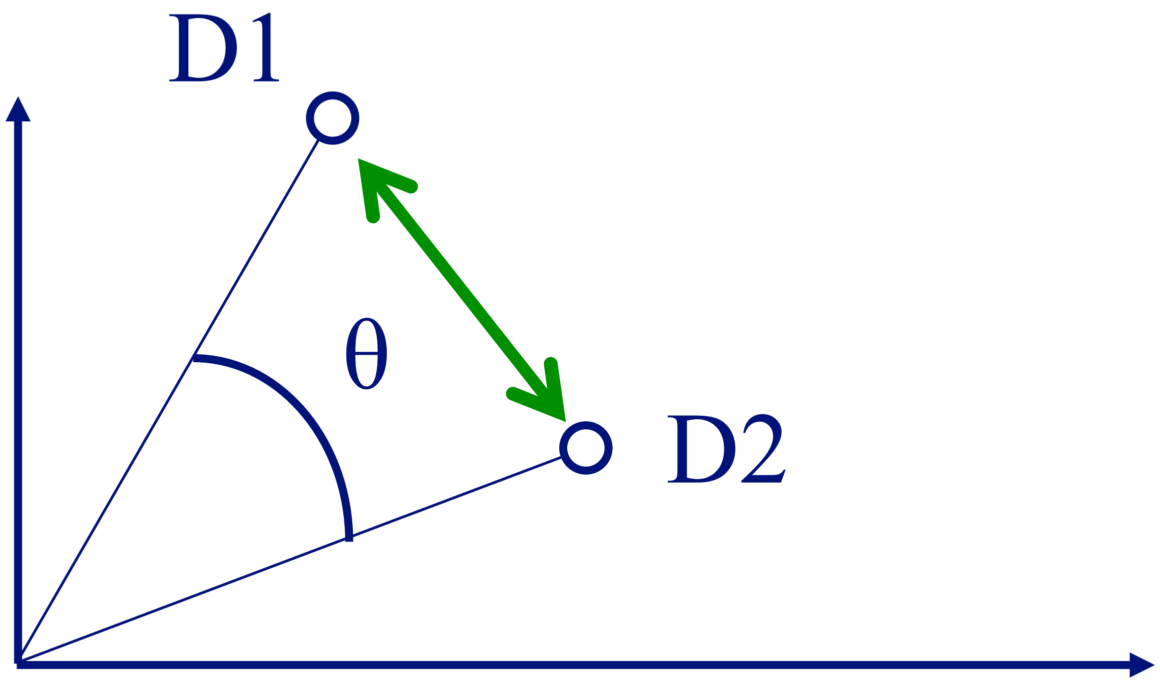
$$\text{sum}(v_{1,i} * v_{2,i}) / [\text{len}(v_1) * \text{len}(v_2)]$$



Cluster generation

cosine similarity - observations:

- related to the Euclidean distance
- weights $v_{i,j}$: according to tf/idf



Cluster generation

tf (‘term frequency’)

high, if the term appears very often in this document.

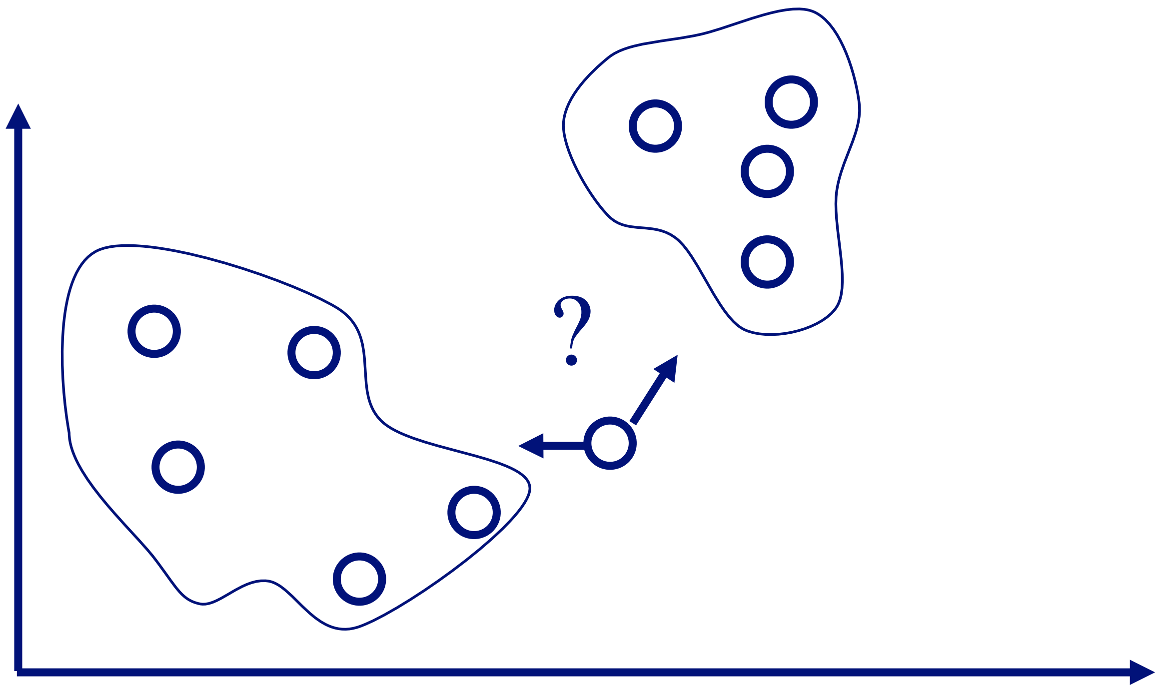
idf (‘inverse document frequency’)

penalizes ‘common’ words, that appear in almost every document

Cluster generation

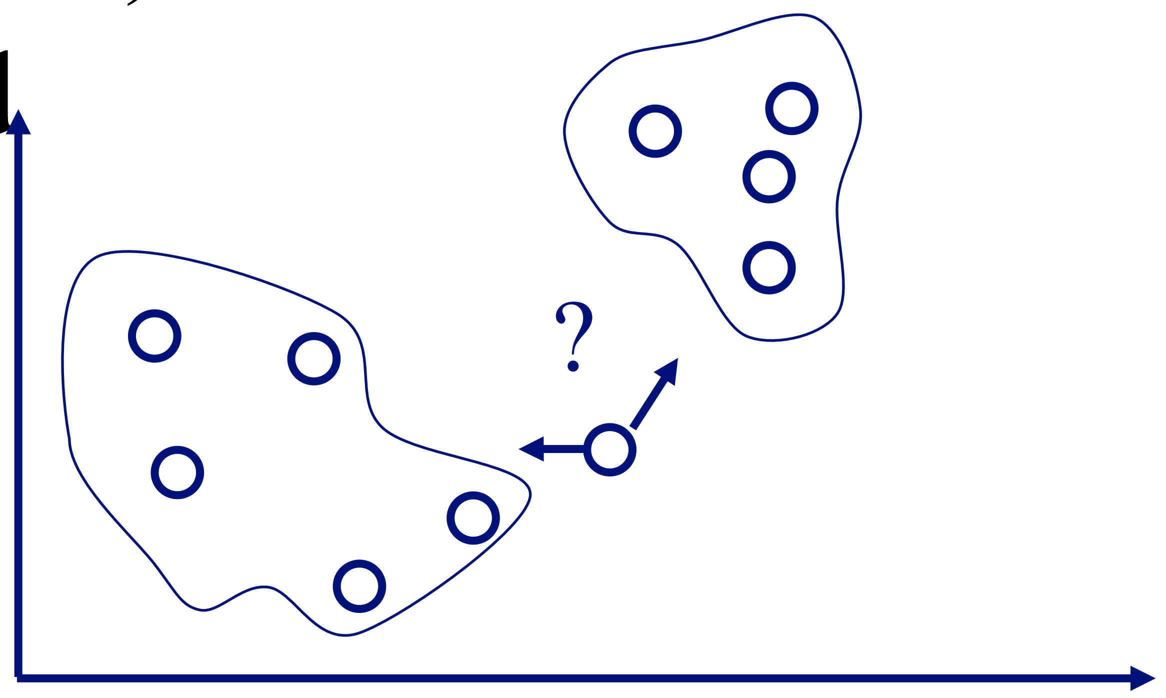
We need

- Q1: document-to-document similarity
- Q2: document-to-cluster similarity



Cluster generation

- A1: min distance (‘single-link’)
- A2: max distance (‘all-link’)
- A3: avg distance (gives same cluster ranking as A4, but different values)
- A4: distance to centroid



Cluster generation

- A1: min distance (‘single-link’)
  - leads to elongated clusters
- A2: max distance (‘all-link’)
  - many, small, tight clusters
- A3: avg distance
  - in between the above
- A4: distance to centroid
  - fast to compute

Cluster generation

We have

- document-to-document similarity
- document-to-cluster similarity

Q: How to group documents into ‘natural’ clusters

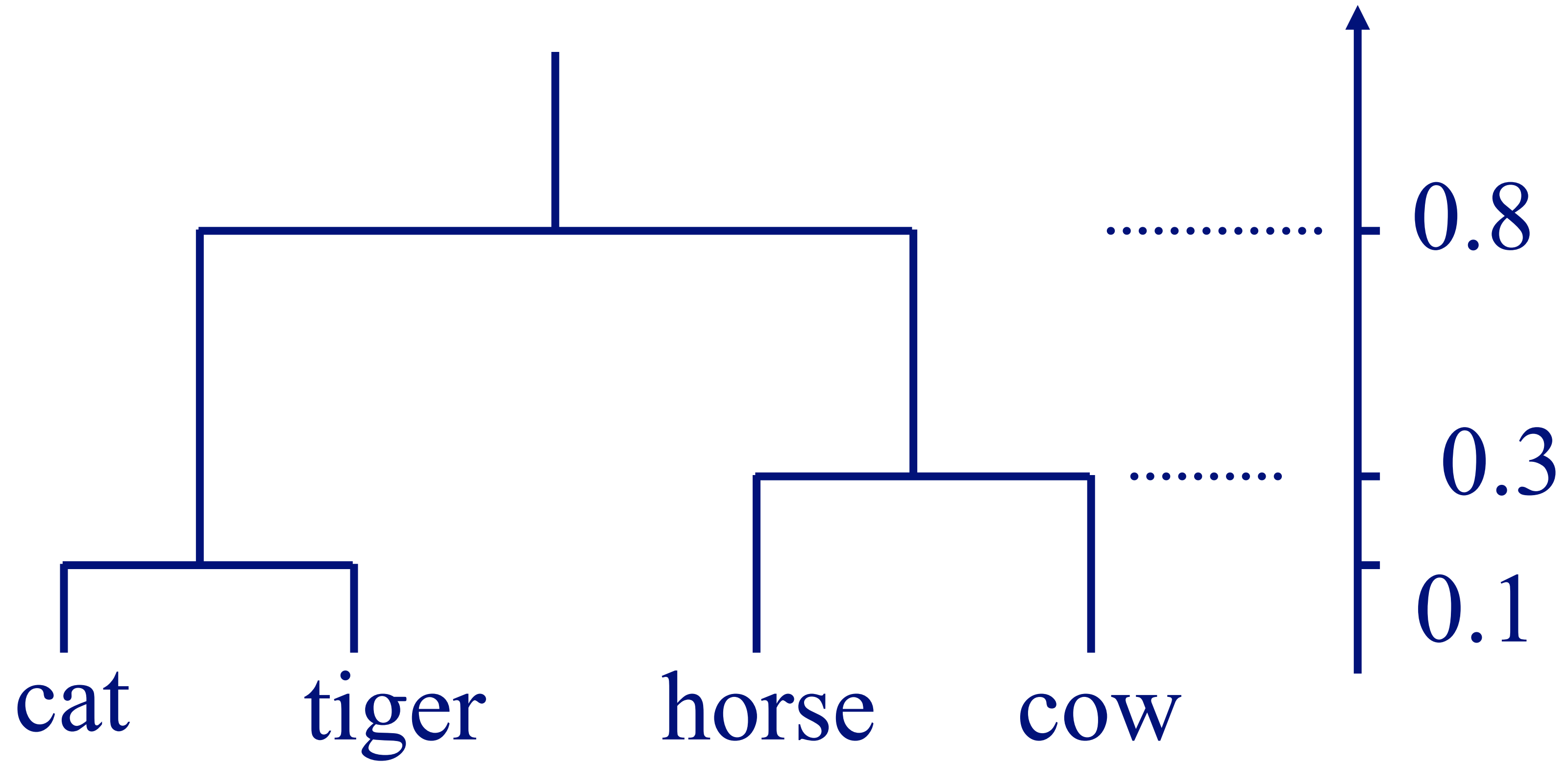
Cluster generation

A: *many-many* algorithms - in two groups [VanRijsbergen]:

- theoretically sound (O(N^2))
  - independent of the insertion order
- iterative (O(N), O(N log(N))

Cluster generation - ‘sound’ methods

- Approach#1: dendrograms - create a hierarchy (bottom up or top-down) - choose a cut-off (how?) and cut

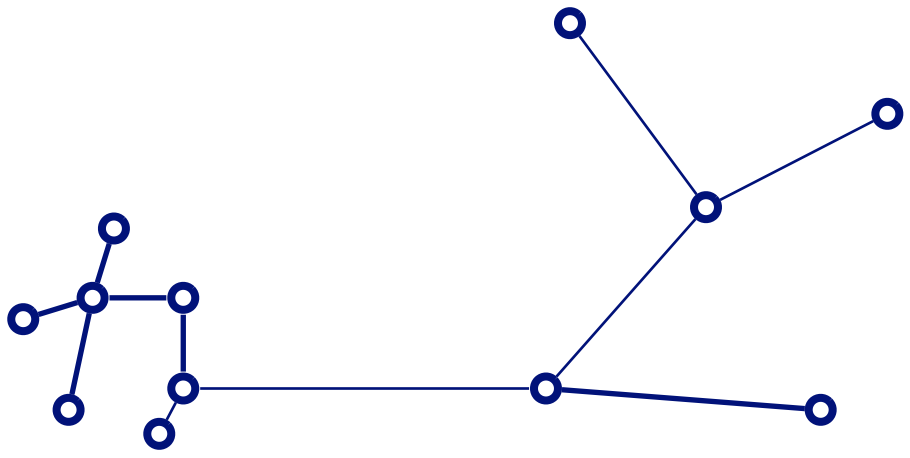


Cluster generation - ‘sound’ methods

- Approach#2: min. some statistical criterion (eg., sum of squares from cluster centers)
  - like ‘k-means’
  - but how to decide ‘k’?

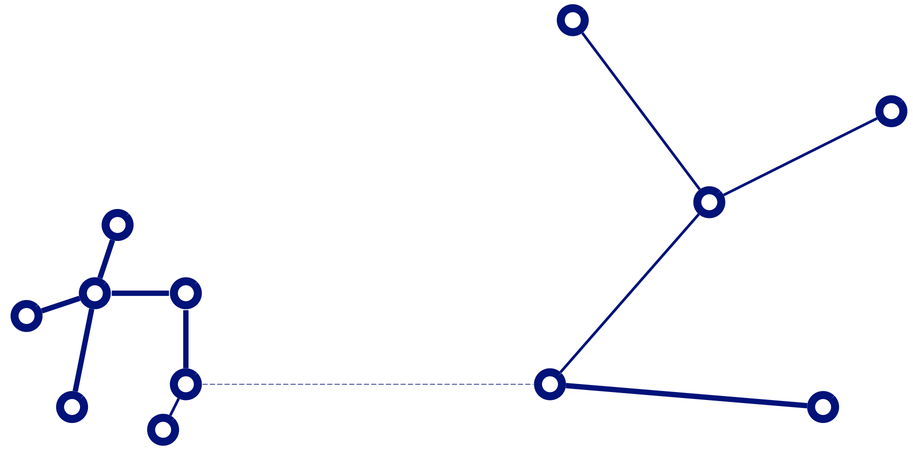
Cluster generation - ‘sound’ methods

- Approach#3: Graph theoretic [Zahn]:
  - build MST;
  - delete edges longer than 3* std of the local average



Cluster generation - ‘sound’ methods

- Result:



- why ‘3’?
- variations
- Complexity?

Cluster generation - ‘iterative’ methods

General outline:

- Choose ‘seeds’ (how?)
- assign each vector to its closest seed (possibly adjusting cluster centroid)
- possibly, re-assign some vectors to improve clusters

Fast and practical, but ‘unpredictable’

Cluster generation

one way to estimate # of clusters *k*: the 'cover coefficient' [Can+] ~ SVD

LSI - Detailed outline

- LSI
  - problem definition
  - main idea
  - experiments

Information Filtering + LSI

- [Foltz+,'92] Goal:
  - users specify interests (= keywords)
  - system alerts them, on suitable news-documents
- Major contribution:
  LSI = Latent Semantic Indexing
  - latent ('hidden') concepts

Information Filtering + LSI

Main idea

- map each document into some **'concepts'**
- map each term into some **'concepts'**

'Concept':~ a set of terms, with weights,
e.g. DBMS_concept:
"data" (0.8),
"system" (0.5),
"retrieval" (0.6)

Information Filtering + LSI

Pictorially: term-document matrix (BEFORE)

	'data'	'system'	'retrieval'	'lung'	'ear'
TR1	1	1	1		
TR2	1	1	1		
TR3				1	1
TR4				1	1

Information Filtering + LSI

Pictorially: **concept-document** matrix and...

	'DBMS-concept'	'medical-concept'
TR1	1	
TR2	1	
TR3		1
TR4		1

Information Filtering + LSI

... and **concept-term** matrix

	'DBMS-concept'	'medical-concept'
data	1	
system	1	
retrieval	1	
lung		1
ear		1

Information Filtering + LSI

Q: How to search, e.g., for ‘system’?

Information Filtering + LSI

A: find the corresponding concept(s); and the corresponding documents

	'DBMS-concept'	'medical-concept'
data	1	
system	1	
retrieval	1	
lung		1
ear		1

	'DBMS-concept'	'medical-concept'
TR1	1	
TR2	1	
TR3		1
TR4		1

Information Filtering + LSI

A: find the corresponding concept(s); and the corresponding documents

	'DBMS-concept'	'medical-concept'
data	1	
system	1	
retrieval	1	
lung		1
ear		1

	'DBMS-concept'	'medical-concept'
TR1	1	
TR2	1	
TR3		1
TR4		1

Information Filtering + LSI

Thus it works like an (automatically constructed) thesaurus:

we may retrieve documents that DON’T have the term ‘system’, but they contain almost everything else (‘data’, ‘retrieval’)

LSI - Discussion - Conclusions

- Great idea,
  - to derive ‘concepts’ from documents
  - to build a ‘statistical thesaurus’ automatically
  - to reduce dimensionality (down to few “concepts”)
- How exactly SVD works? (Details, next)