CS4442/9542b: Artificial Intelligence II
Prof. Olga Veksler

Lecture 11

NLP: Information Retrieval

Many slides from: L. Kosseim (Concordia), Jamie Callan (CMU), *Christopher Manning (Stanford),* L. Venkata Subramaniam, Phillip Resnik

Outline

- Introduction to Information Retrieval (IR)
- Ad hoc information retrieval
  - Boolean Model
  - Vector Space Model
    - Cosine similarity measure
    - Choosing term weights
  - Performance evaluation methods
  - Improving IR system
    - Query expansion
    - Relevance feedback

Information Retrieval Intro

- **Then:** most digital information is stored in databases
  - Structured data storage
  - Supports efficient information extraction with queries
  - mostly used by corporations/governments
- **Now:** most digital information is stored in unstructured text form (reports, email, web pages, discussion boards, blogs, etc)
  - Estimates: 70%, 90% ?? All depends how you measure.
  - Unstructured data, not in traditional databases
  - Used by companies/organizations/people
  - How do you extract information from unstructured text data?

The Problem

- When people see text, they understand its meaning (by and large)
- When computers see text, they get only character strings (and perhaps HTML tags)
- We'd like computer agents to see meanings and be able to intelligently process text
- These desires have led to many proposals for structured, semantically marked up formats
- But often human beings still resolutely make use of text in human languages
- This problem isn't likely to just go away

Information Retrieval

- IR deals with retrieving information from unstructured document repositories
- Traditionally
  - Text documents repositories
- More recently
  - Speech
  - Images
  - music
  - Video

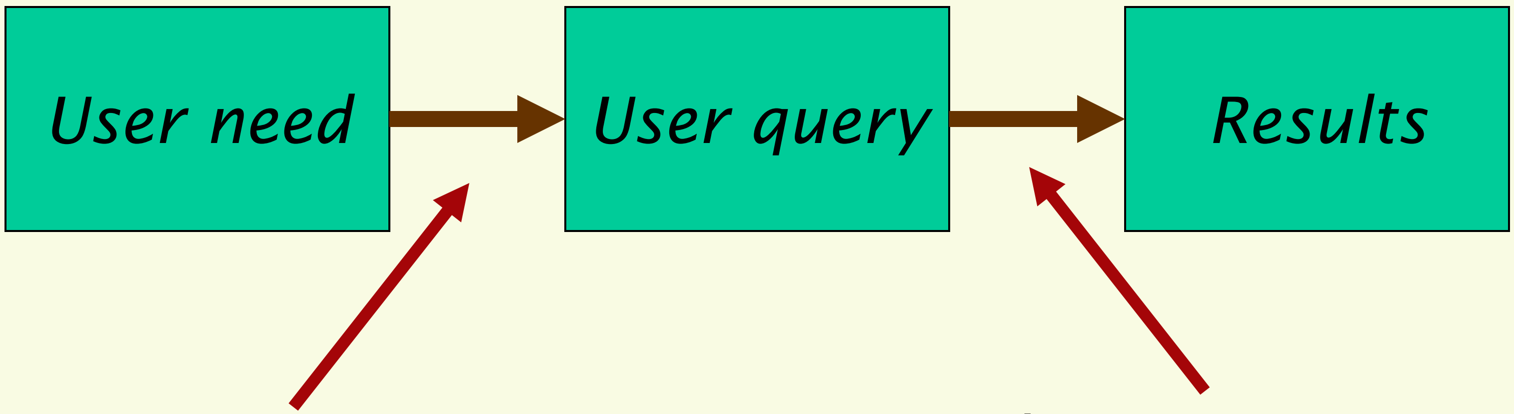
Translating User Needs: Databases

User need → *User query* → *Results*

For databases, a lot of people know how to do this correctly, using SQL or a GUI tool

The answers coming out here will then be precisely what the user wanted

Translating User Needs: Text Documents



*For meanings in text,
no IR-style query
gives one exactly
what one wants;
it only hints at it*

*The answers
coming out may
be roughly what
was wanted, or
can be refined*

Sometimes!

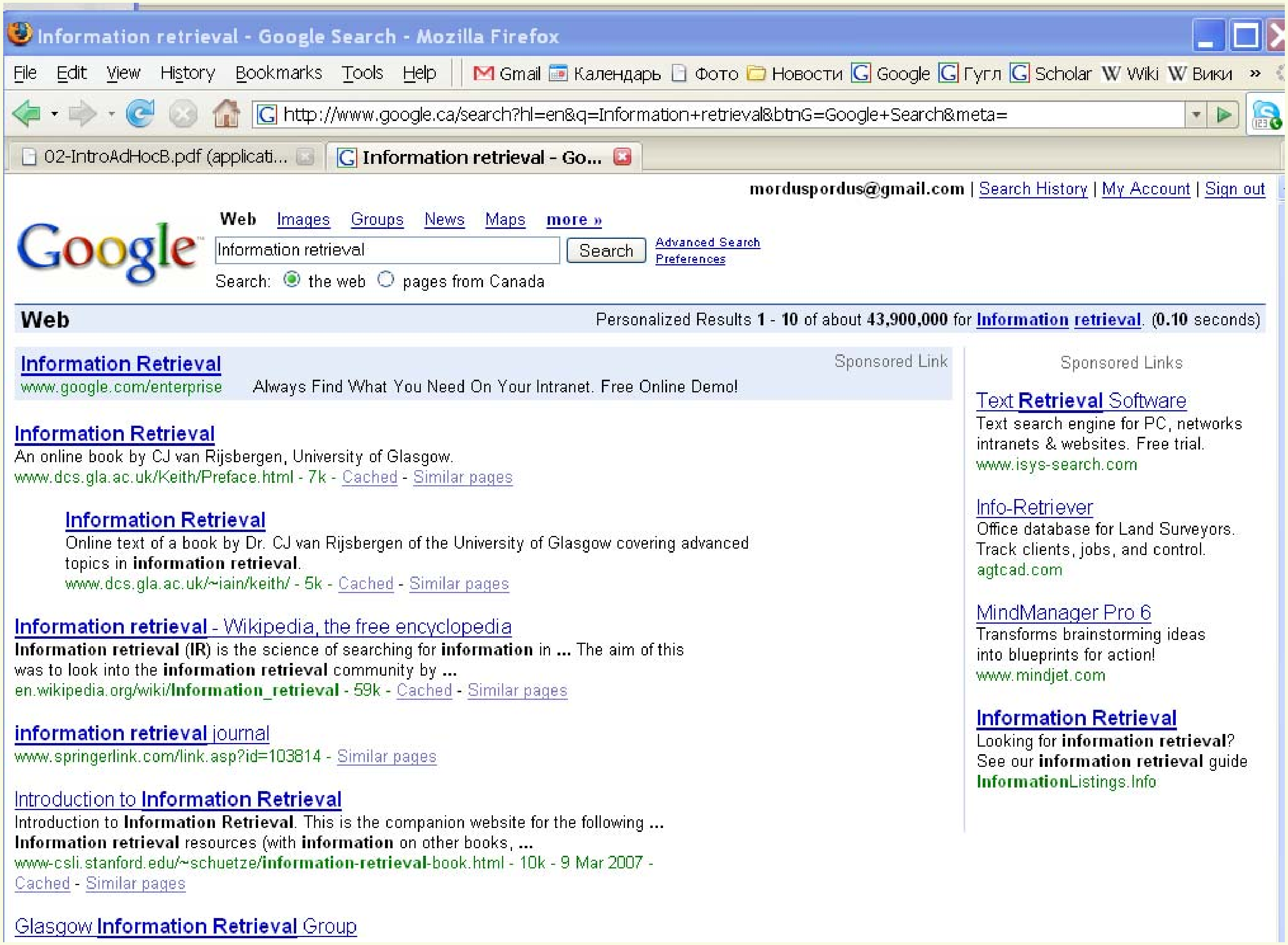
Major Types of Information Retrieval

- ad-hoc retrieval
  - user creates an “ad hoc” query which is usually not reused or saved
  - system returns a list of (hopefully) relevant documents
  - sometimes also called “archival” retrieval
  - no training data is available
  - **topic of the lecture**
- classification / categorization
  - training data is available
  - documents are classified in a pre-determined set of categories
  - Ex: Reuters (corporate news (CORP-NEWS), crude oil (CRUDE), acquisitions (ACQ), …)
  - any of machine learning techniques can be used
- filtering / routing
  - special cases of categorization
  - 2 categories: relevant and not-relevant
  - filtering:
    - absolute assessment (d1 is relevant but d2 is not)
  - routing:
    - relative ranking of documents (like in ad-hoc) (d1 is more relevant than d2)

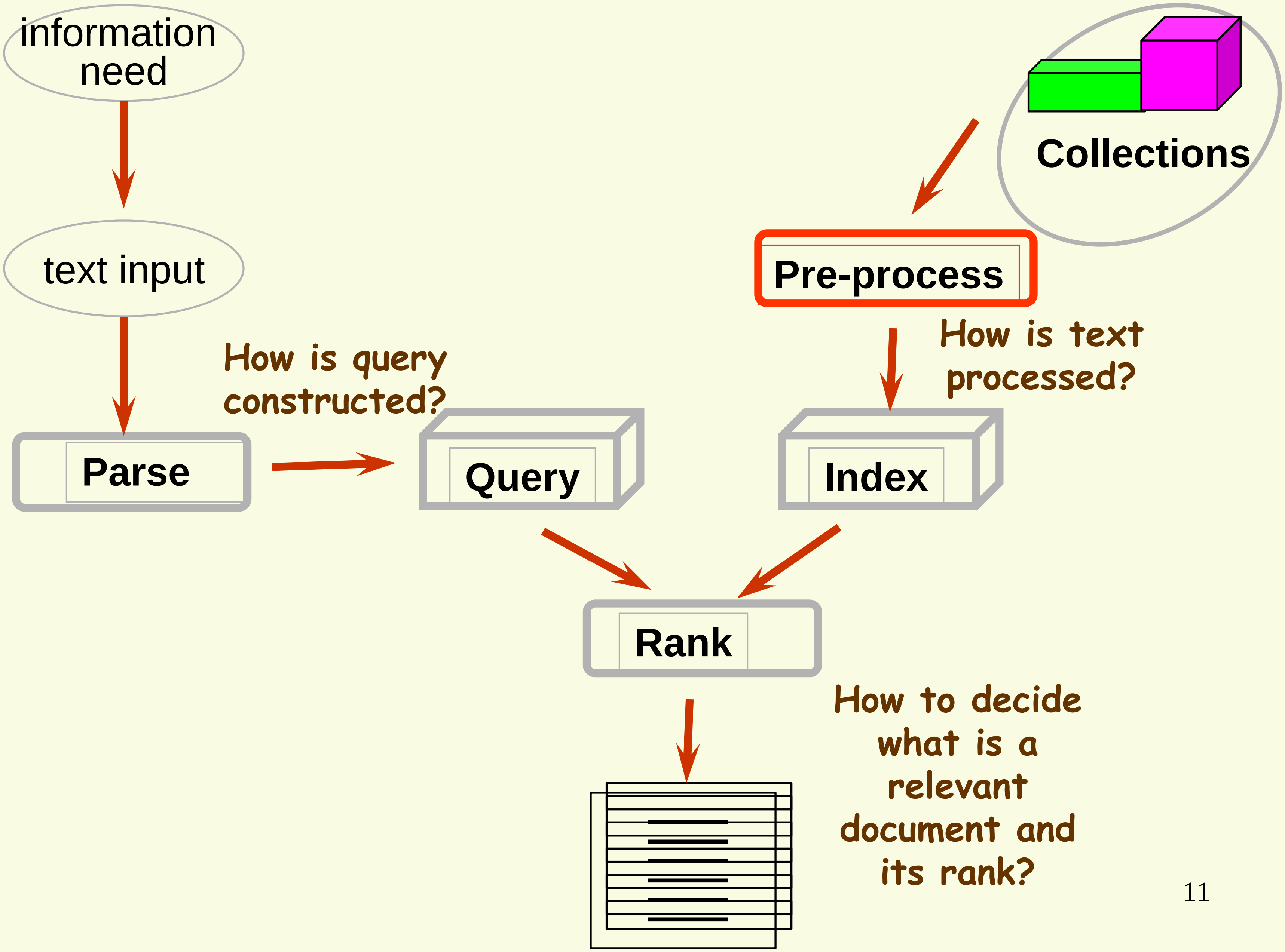
Different Types of Ad-Hoc Retrieval

- Web search
  - Massive collection (10^8-10^9) of documents
  - Query log analysis reveals population-based patterns
  - Typically high precision (most retrieved documents are relevant), low recall (not all relevant documents are retrieved)
- Commercial information providers (e.g. West, LexisNexis)
  - Large Collection (10^6-10^8) of documents
  - often high recall is essential (e.g. legal or patent search)
- Enterprise search (e.g. UWO, IBM)
  - Medium-sized to large collection (10^4-10^6) of documents
  - Opportunity to exploit domain knowledge
- Personal search (e.g. your PC)
  - Small collection (10^3-10^4) of documents
  - Good opportunity to learn a user model, do personalization

Example of ad-hoc IR



Information Retrieval Process



Relevance

- In what ways can a document be relevant to a query?
  - Answer precise question precisely
  - Partially answer question
  - Suggest a source for more information
  - Give background information
  - Remind the user of other knowledge
  - Others ...

Two Major Issues

- Indexing
  - How do we represent a collection of documents to support fast search?
- Retrieval methods
  - How do we match a user query to indexed documents?

Indexing

- Most IR systems use **inverted index** to represent collection of texts
- Inverted Index = a data structure that lists for each word all documents in the collection that contain that word

assassination	$\{d_1, d_4, d_{95}, d_5, d_{90}\ldots\}$
murder	$\{d_3, d_7, d_{95}\ldots\}$
Kennedy	$\{d_{24}, d_7, d_{44}\ldots\}$
conspiracy	$\{d_3, d_{55}, d_{90}, d_{98}\ldots\}$

- Inverted Index is also called inverted file and postings file
- Inverted index is usually implemented as a dictionary which allows fast lookups based on word
  - B-trees, hash tables, etc are used to implement a dictionary

Indexing

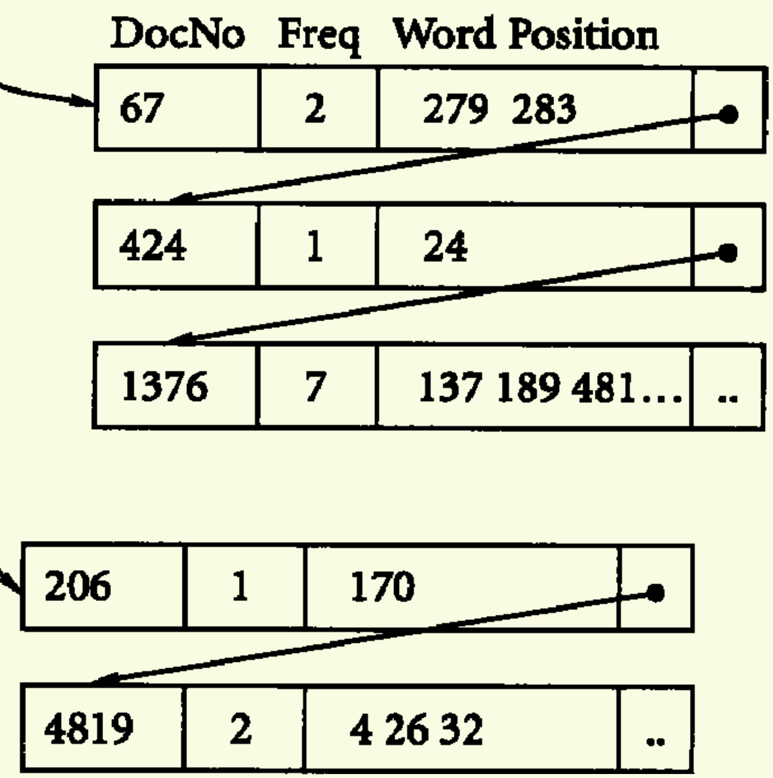
- More sophisticated version of inverted index also contains position information, say byte offset from the beginning of the document
  - Can search for phrases efficiently
  - Example: need to find “car insurance”
    - “car” occurs in documents (d_1, offset 5), (d_7, offset 10), (d_9, offset 35)
    - “insurance” occurs in documents (d_2, offset 3), (d_7, offset 11), (d_8, offset 7)
    - “car insurance” occurs in document d_7
  - Still rather primitive: “car insurance” ≠ “insurance for car”
  - Possible solution: can find frequent phrases (simply frequently occurring bigrams, trigrams, etc.) and index those too, in addition to words:

car insurance	$\{d_1, d_4, d_{95}, d_5, d_{90} \ldots\}$
insurance for car	$\{d_5, d_7, d_{95}, d_{90} \ldots\}$

- So we index words and word phrases
- I will often say “term” to refer to these indexed entities
  - However, sometimes I will just say “word”, because it’s simpler.

Inverted Index Example

term	DocCnt	FreqCnt	Head
ABANDON	28	51	•
ABIL	32	37	•
ABSENC	135	185	...
ABSTRACT	7	10	...



- For each term:
  - **DocCnt:** in how many documents the word occurs
  - **FreqCnt:** the total number of times the word occurs in all documents
- For each document
  - **Freq**: how many times word occurs in this document
  - **WordPosition**: offset where these occurrences are found in the document

Choosing Terms To Index

1. Controlled Vocabulary Indexing
   - A human expert selects a set of terms to index
   - This is done for libraries, web directories, etc
   - Pros
     - Usually “controlled” terms are unambiguous
   - Cons:
     - Expensive, need manual work
     - Controlled vocabularies can’t represent arbitrary detail
2. Free Text Indexing
   - Automatically select “good” terms to index
   - Some search engines do this
3. Full Text Indexing
   - Most search engines do this
   - Cons:
     - Many words are ambiguous
   - Pros:
     - Can represent arbitrary detail
     - Inexpensive and easy

Full Text Indexing

Are these terms useful?

Term	Tf	Term	Tf	Term	tf
the	78	up	8	pictures	6
to	35	for	7	red	6
i	31	have	7	digital	5
and	29	image	7	eye	5
a	19	like	7	not	5
camera	17	mode	7	on	5
is	17	much	7	or	5
in	12	software	7	shutter	5
with	11	very	7	sony	5
be	9	can	6	than	5
but	9	images	6	that	5
it	9	movies	6	after	4
of	9	my	6	also	4
this	9	no	6	: :	:

Can you tell what this document is about?

Full Text Indexing Design Issues

- To stem or not to stem
  - Stemming: *laughing, laughs, laugh* and *laughed* are all stemmed to *laugh*
  - Problem: semantically different words like *gallery* and *gall* may both be truncated to *gall* making the stems unintelligible to
- Exclude/Include Stop words
  - Stop words make up about 50% of the text, excluding them makes representation more space efficient
  - But impossible to search for documents for phrases containing stop words
    - “to be or not to be”, “take over”
    - Most queries are unaffected, but could be very annoying sometimes

Full Text Indexing: after Stemming and Stop Word Removal

Term	Tf	Term	Tf	Term	tf
camera	18	sony	5	lag	3
image	13	after	4	last	3
like	8	any	4	lcd	3
mode	8	auto	4	mavica	3
up	8	battery	4	record	3
buy	7	flash	4	reduce	3
movie	7	problem	4	size	3
picture	7	zoom	4	15	2
software	6	include	3	2mp	2
red	6	2100	3	8x10	2
digital	5	button	3	98	2
eye	5	down	3	automatic	2
look	5	feature	3	bag	2
shutter	5	focus	3	best	2

Problems with Index Terms

- May not retrieve relevant documents that include synonymous terms.
  - “restaurant” vs. “café”
  - “PRC” vs. “China”
- May retrieve irrelevant documents that include ambiguous terms.
  - “bat” (baseball vs. mammal)
  - “Apple” (company vs. fruit)
  - “bit” (unit of data vs. act of eating)

Retrieval models

- 3 basic models:
  - boolean model
    - the oldest one, similar to what is used in database queries
  - vector-space model
    - most popular in IR
  - probabilistic model
    - more powerful than those above
    - tries to model the probability that the document is generated by the given query
    - but we will not study this one

- Different approaches vary on:
  - how they represent the query & the documents
  - how they calculate the relevance between the query and the documents

Boolean Model

- user gives a set of terms (keywords) that are likely to appear in relevant documents
  - *Ex: JFK Kennedy conspiracy assassanation*
- Connects the terms in the query with Boolean operators (AND, OR, NOT)

```
AND(Kennedy, conspiracy, assassanation)
```

- Can expand query using synonyms

```
AND (OR (Kennedy, JFK),
        (OR (conspiracy, plot),
           (OR (assassanation, assassinated,
                assassinate, murder, murdered, kill, killed)
           )
          )
     )
```

Example

- Which of these documents will be returned for the following query :

  computer `AND` (*information* `OR` *document*) `AND` *retrieval*

document collection:

d_1: {*computer* √ , *software*, *information* √, *language*} ×

d_2: {*computer* √, *document* √, *retrieval* √, *library*} √

d_3: {*computer* √, *information* √, *filtering*, *retrieval* √} √

Implementation With Set Operators

- Assume that:
  - the inverted index contains:

    t1-list: {d1,d2,d3,d4}   t2-list: {d1,d2}   t3-list: {d1,d2,d3}   t4-list: {d1}
  - The query Q = (t1 AND t2) OR (t3 AND (NOT t4))
- We perform set operations:
  - to satisfy (t1 **AND** t2), we **intersect** the t1 and t2 lists
    - {d1,d2,d3,d4} ∩ {d1,d2} = {d1,d2}
  - to satisfy (t3 AND (**NOT** t4)), we **subtract** the t4 list from the t3 list
    - {d1,d2,d3} - {d1} = {d2,d3}
  - to satisfy (t1 AND t2) **OR** (t3 AND (NOT t4)), we take the **union** of the two sets of documents obtained for the parts.
    - {d1,d2} ∪ {d2,d3} = {d1,d2,d3}

Analysis of the Boolean Model

- advantages
  - simple retrieval model
  - queries are expressed with Boolean operators (semantics is clearly defined)
  - Results are easy to explain
  - usually computationally efficient
- disadvantages
  - retrieval strategy is a binary decision (relevant or not)
  - difficult to *rank* documents in order of relevance
  - non-expert users have difficulty to express their need as Boolean expressions. Studies show that people create quires that are either
    - **too strict**: few relevant documents are found
    - **too loose**: too many documents (most of them irrelevant) are found
    - Therefore most boolean searches on the web either return no documents or a huge set of documents

Vector-Space Model

- Documents and queries can be represented by a “term vector”
  - Each dimension corresponds to a term in the vocabulary
- Similarity between a document and a query is determined by a distance in vector space
- First system is “SMART” system
  - Developed by G. Salton at Cornell 1960-1999
  - Still used widely today

Gerard Salton

Term-Document Matrix

- the collection of documents is represented by a matrix of weights called a term-by-document matrix

	d_1	d_2	d_3	d_4	d_5	…
$term_1$	w_{11}	w_{12}	w_{13}	w_{14}	w_{15}	
$term_2$	w_{21}	w_{22}	w_{23}	w_{24}	w_{25}	
$term_3$	w_{31}	w_{32}	w_{33}	w_{34}	w_{35}	
…						
$Term_N$	w_{n1}	w_{n2}	w_{n3}	w_{n4}	w_{n5}	

- 1 column = representation of one document
- 1 row = representation of 1 term across all documents
- cell w_{ij} = weight of term i in document j
  - simplest weight w_{ij} is the number of times term i occurred in document j
- note: the matrix is sparse (most weights are 0)

Bags of Words

- This is also called **bags of words** representation
  - The document is the “Bag”
  - The “bag” contains word tokens
  - A particular word may occur more than once in the bag
  - “Stop” words are usually ignored
    - “the”,”a”,”to”,…
  - Word order is completely ignored

“I see what I eat “ = “I eat what I see”

Document 1

The quick brown fox jumped over the lazy dog’s back.

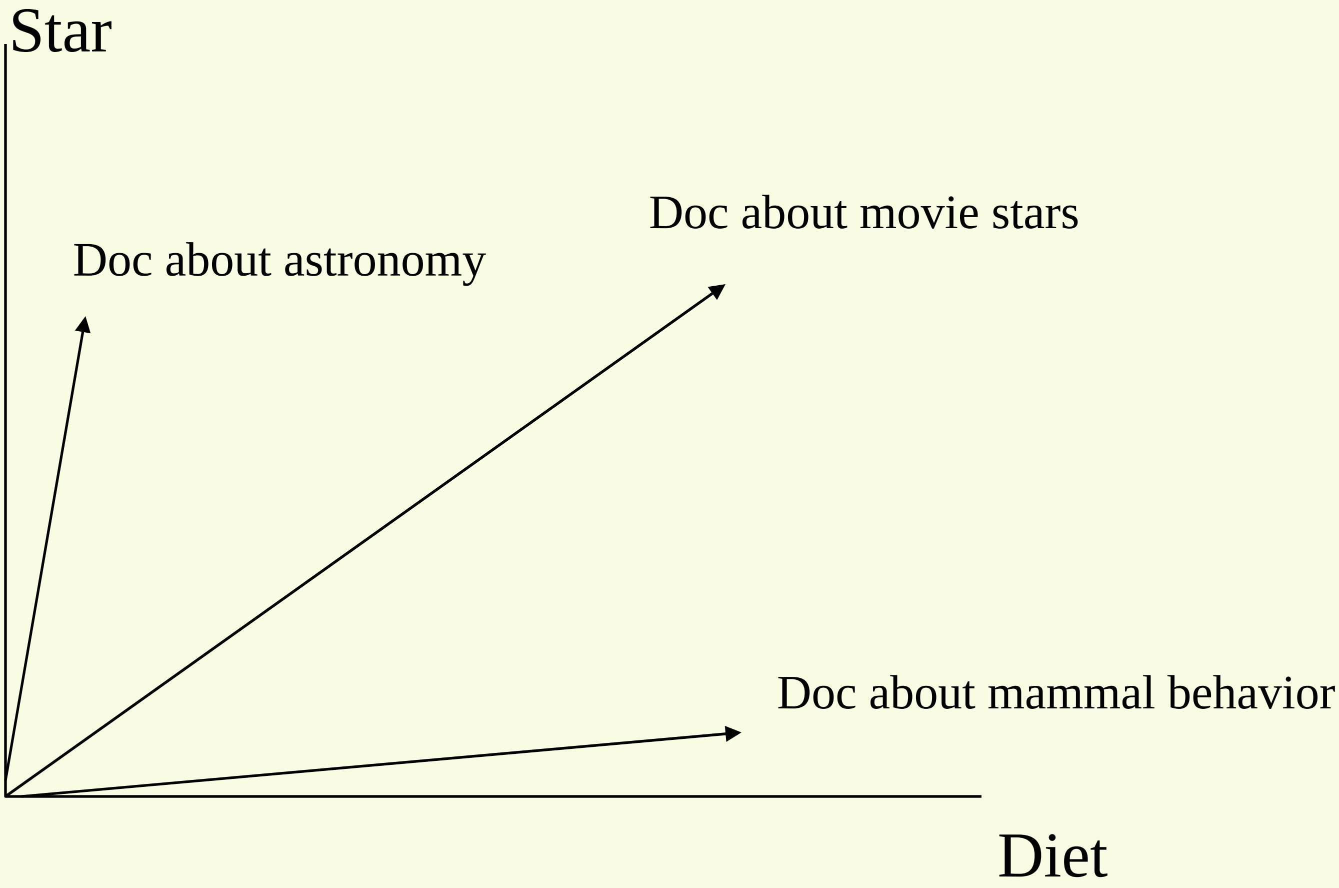
Document 2

Now is the time for all good men to come to the aid of their party.

Indexed Term	Document 1	Document 2
aid	0	1
all	0	1
back	1	0
brown	1	0
come	0	1
dog	1	0
fox	1	0
good	0	1
jump	1	0
lazy	1	0
men	0	1
now	0	1
over	1	0
party	0	1
quick	1	0
their	0	1
time	0	1

Stop words: for, is, of, ‘s, the, to

Documents as Vectors



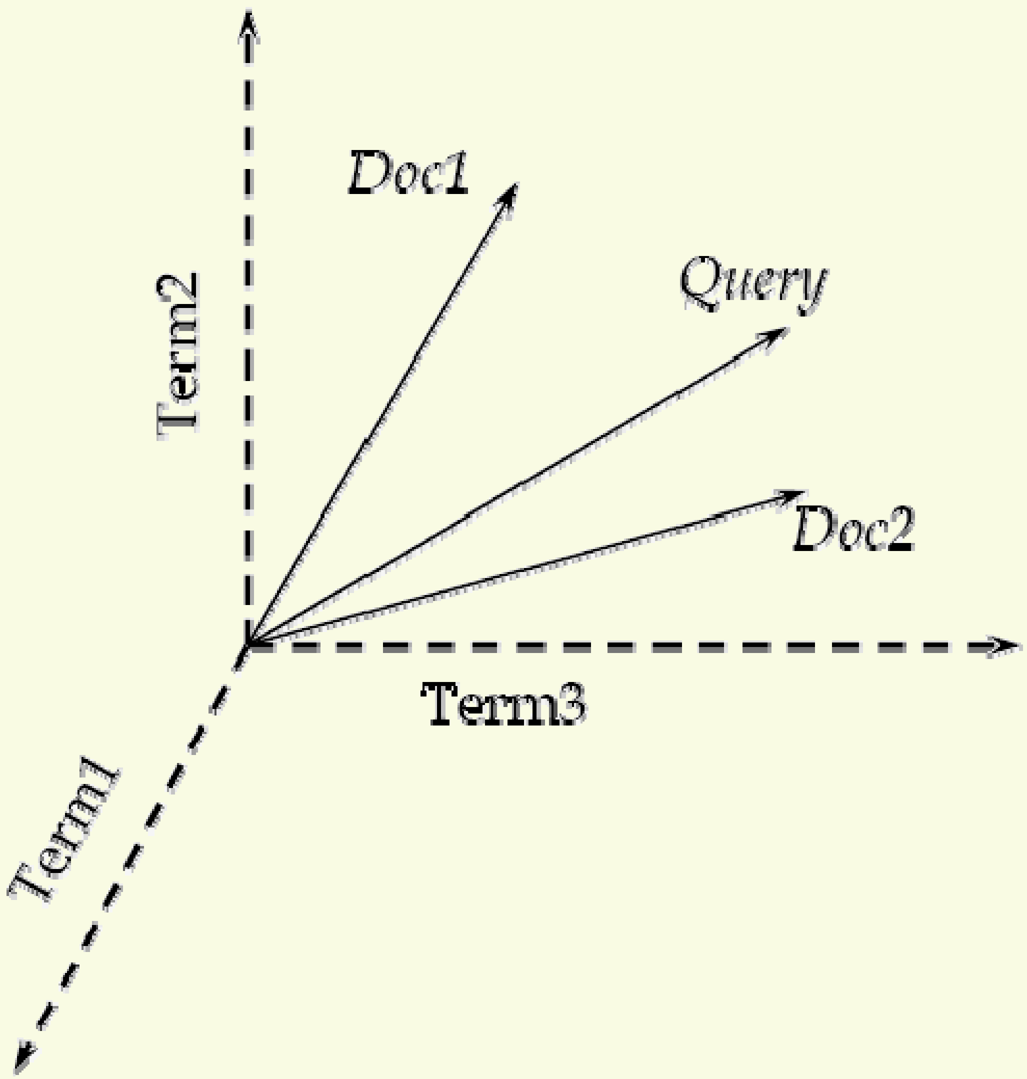
Query Representation

- A query can also be represented as a vector, like a document

$$q = (0,0,0,1,0,\ldots\ldots,1,\ldots\ldots 0,1)$$

- Size of vector corresponding to query ***q*** is also the number of terms

Vector Space Similarity



Similarity is inversely related to the angle between the vectors.

Doc2 **is the most similar to the *Query*.**

Rank the documents by their similarity to the *Query*.

Example

- The collection:
  - d_1 = {introduction knowledge in speech and language processing ambiguity models and algorithms language thought and understanding the state of the art and the near-term future some brief history summary}
  - d_2 = {hmms and speech recognition speech recognition architecture overview of the hidden markov models the viterbi algorithm revisited advanced methods in decoding acoustic processing of speech computing acoustic probabilities training a speech recognizer waveform generation for speech synthesis human speech recognition summary}
  - d_3 = {language and complexity the chomsky hierarchy how to tell if a language isn't regular the pumping lemma are English and other languages regular languages ? is natural language context-free complexity and human processing summary}
- The query:

  Q = {speech language processing}

Example Continued

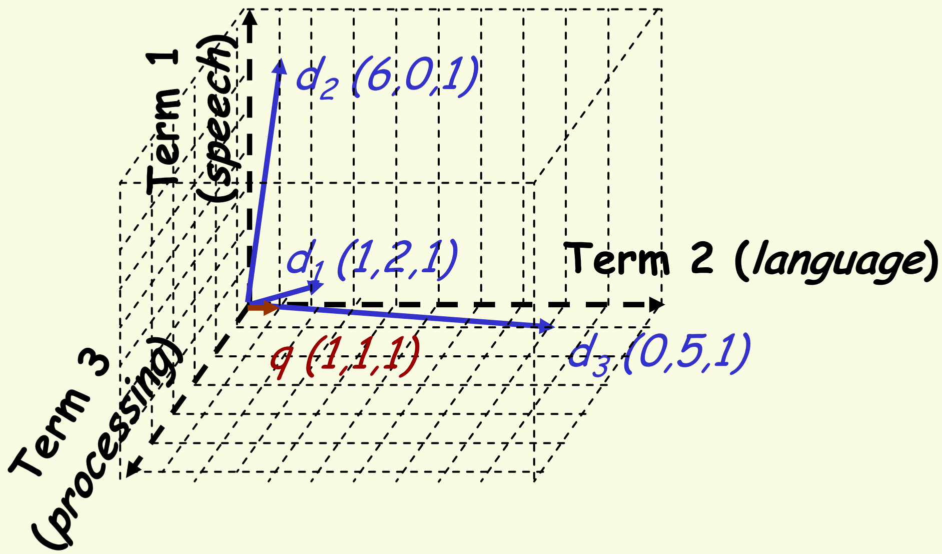
- The collection:
  - d_1 = {introduction knowledge in speech and language processing ambiguity models and algorithms language thought and understanding the state of the art and the near-term future some brief history summary}
  - d_2 = {hmms and speech recognition speech recognition architecture overview of the hidden markov models the viterbi algorithm revisited advanced methods in decoding acoustic processing of speech computing acoustic probabilities training a speech recognizer waveform generation for speech synthesis human speech recognition summary}
  - d_3 = {language and complexity the chomsky hierarchy how to tell if a language isn't regular the pumping lemma are English and other language regular language ? is natural language context-free complexity and human processing summary}
- The query:

  Q = {speech language processing}

Example Continued

- using raw term frequencies for weights

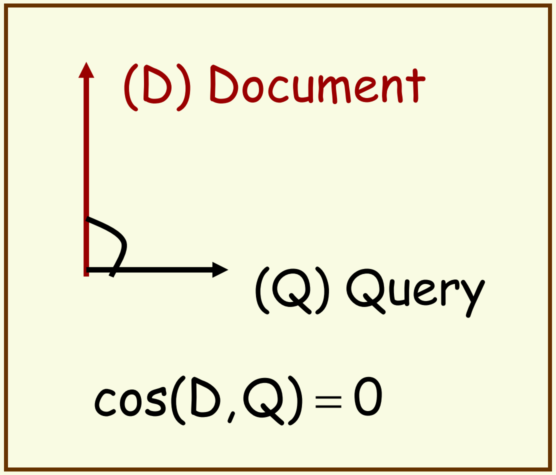
	d_1	d_2	d_3	Q
introduction	…	…	…	…
knowledge	…	…	…	…
…	…	…	…	…
speech	1	6	0	1
language	2	0	5	1
processing	1	1	1	1
…	…	…	…	…

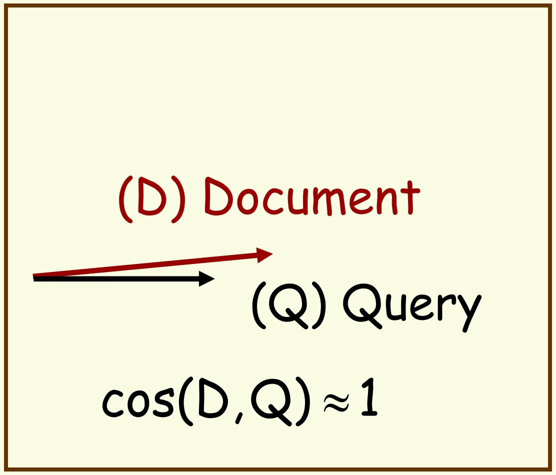


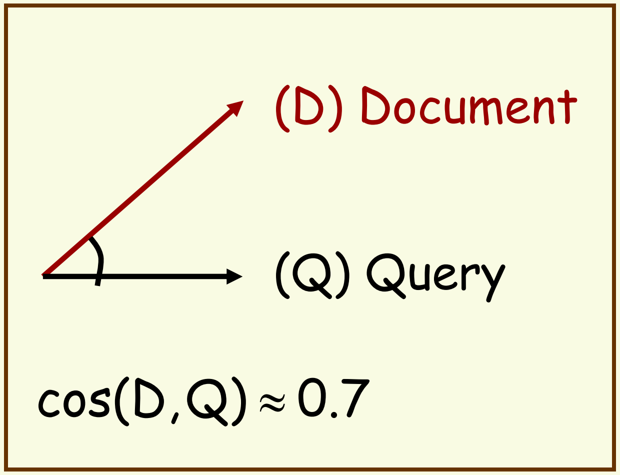
- vectors for the documents and the query can be seen as a point in a multi-dimensional space
  - where each dimension is a term

The Cosine Measure

- similarity between the document and query (or two documents) is measured by the cosine of the angle (in N-dimensions) between the 2 vectors
  - if two vectors are identical, they will have a cosine of 1
  - if two vectors are orthogonal (i.e. share no common term), they will have a cosine of 0







- Only the direction is relevant, not the magnitude:
  - any query q is as close to document [1, 2, 1] as to document [2, 4, 2]

The Cosine Measure Continued

- The cosine of 2 vectors (in N dimensions)

inner product

$$cos(d,q) = \frac{d \cdot q}{\|d\| \, \|q\|} = \frac{\sum_{i=1}^{N} d_i \, q_i}{\sqrt{\sum_{i=1}^{N} d_i^2} \, \sqrt{\sum_{i=1}^{N} q_i^2}}$$

lengths of the vectors

- also known as the *normalized inner product*

Example Again

	d_1	d_2	d_3	Q
introduction	1	0	0	0
knowledge	1	0	0	0
…				
speech	1	6	0	1
language	2	0	5	1
processing	1	1	1	1
…				

Q = {speech language processing}

query (1,1,1)

d_1 (1,2,1)
d_2 (6,0,1)
d_3 (0,5,1)

$$sim(d_1,Q) = \frac{(1\text{x}1) + (2\text{x}1) + (1\text{x}1)}{\sqrt{(1^2+2^2+1^2)} \times \sqrt{(1^2+1^2+1^2)}} = \frac{1+2+1}{\sqrt{6} \times \sqrt{3}} = 0.943$$

$$sim(d_2,Q) = \frac{(6\text{x}1) + (0\text{x}1) + (1\text{x}1)}{\sqrt{(6^2+0^2+1^2)} \times \sqrt{(1^2+1^2+1^2)}} = \frac{6+0+1}{\sqrt{37} \times \sqrt{3}} = 0.664$$

$$sim(d_3,Q) = \frac{(0\text{x}1) + (5\text{x}1) + (1\text{x}1)}{\sqrt{(0^2+5^2+1^2)} \times \sqrt{(1^2+1^2+1^2)}} = \frac{0+5+1}{\sqrt{26} \times \sqrt{3}} = 0.680$$

The Cosine Measure Continued

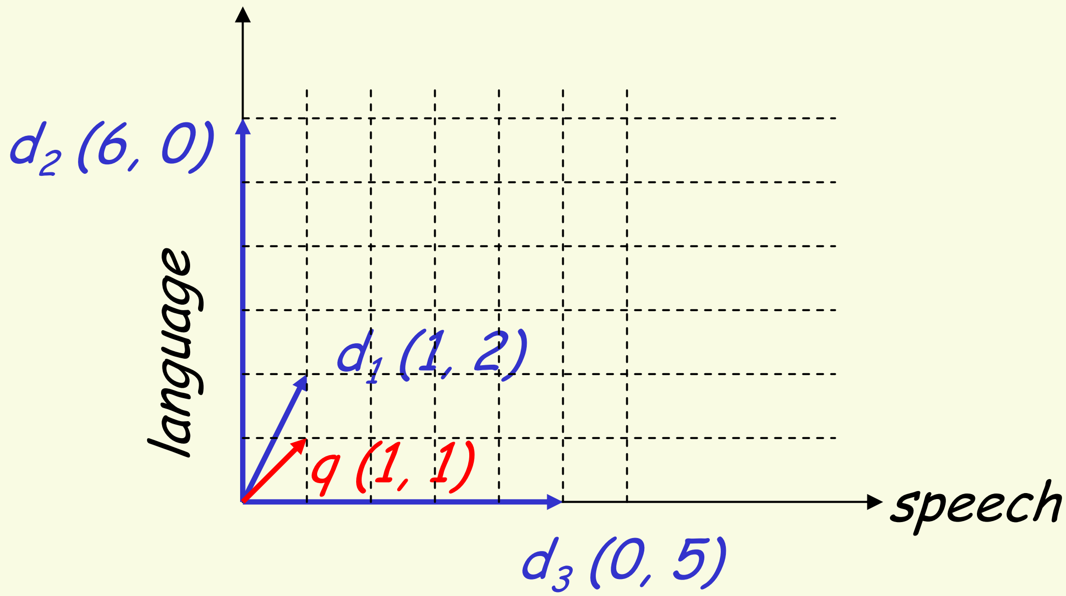
- For efficiency, can normalize raw term frequencies to convert all vectors to length 1
- If ***q*** and ***d*** are normalized, then

$$cos(d,q) = \frac{d \cdot q}{\|d\|\,\|q\|} = d \cdot q$$

Example

Query = "speech language"

original representation:

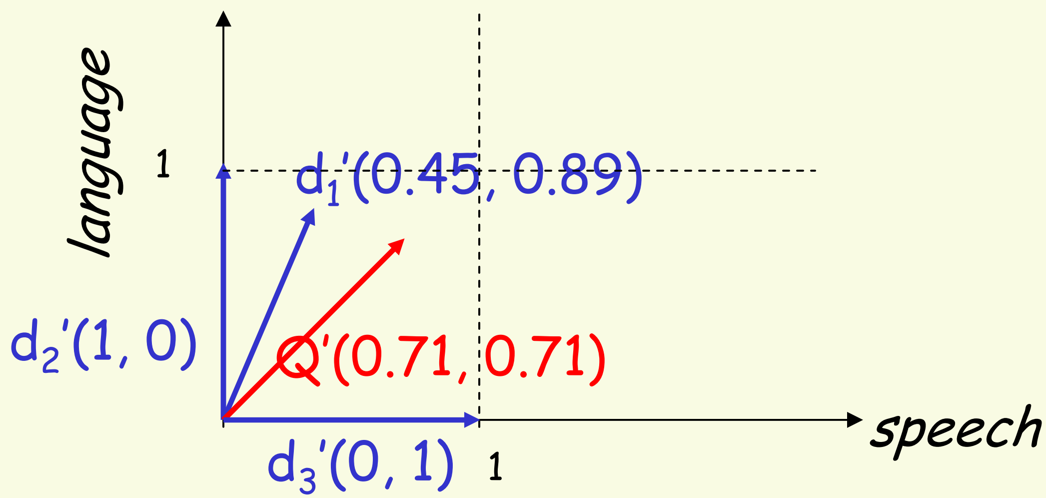


Normalization: reduces vectors to the same length to compute angle

Normalized vectors

Query = “speech language”

representation after normalization:



Q(1,1) $L=\sqrt{1^2+1^2}=1.41$ ***--> normalized Q’ (0.71, 0.71)***

d_1(1,2) $L=\sqrt{1^2+2^2}=2.24$ ***--> normalized d_1’ (0.45, 0.89)***

d_2(6,0) $L=\sqrt{6^2+0^2}=6$ ***--> normalized d_2’ (1, 0)***

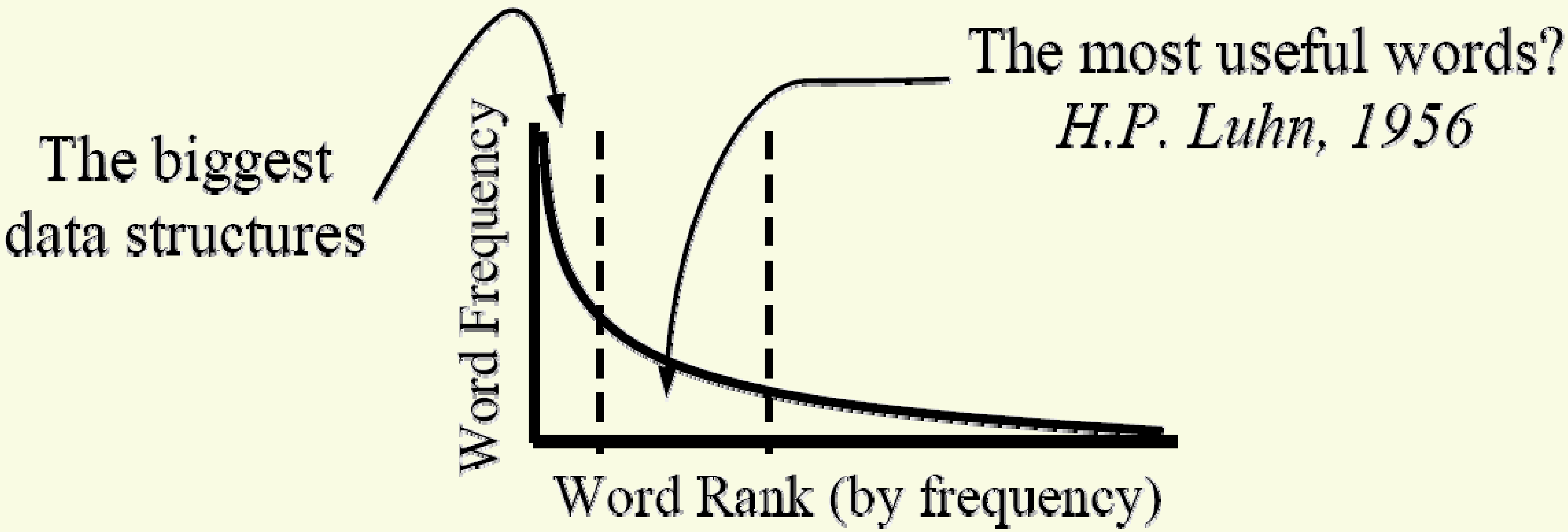
d_3(0,5) $L=\sqrt{0^2+5^2}=5$ ***--> normalized d_3’ (0, 1)***

Term Weights

- The weight w_{ij} reflects the importance of the term T_i in document D_j.
- So far we have used term counts as term weights
  - Normalized them
- Can also use binary weights
  - 0 of term T_i does not occur in document D_j and 1 otherwise
- Vector space model can support real-valued term weights
  - Which might be useful
- But it gives no guidance about what the term weights should be
  - Ad-hoc solutions (use whatever you want for term weights)
  - Use expected distribution of terms
  - Borrow ideas from other retrieval models

Term Weights

- We know something about word distributions: Zipf's law: a few words are frequent, most words are rare



Term Weights

- The weight w_{ij} reflects the importance of the term T_i in document D_j.

- Intuitions:
  1. If a term is frequent in a document, it is probably important in that document: *star*, *play*,…
  2. But if a term that appears in many **other** documents it is not important: e.g., *going*, *come*, …

Assigning Weights to terms

- Want to weight terms highly if they are
  - Frequent in relevant documents…BUT
  - Infrequent in the collection as a whole
- For any term, **tf** (term frequency) is stored in the inverted index
- The higher is **tf** in a document, the better it is describing what the document is about
  - But only if this term is not frequent across all documents!

Inverse Document Frequency

- IDF provides high values for rare words and low values for common words
- Let ***M*** be the number of documents in the collection and ***df*** be the number of documents containing the term
- ***idf*** is often calculated as:

$$idf = \log\left(\frac{M}{df}\right)$$

- Logarithmic "damping", since if a word which is twice more frequent is not necessarily twice more important
- For a collection of 10,000 documents:

$$\log\left(\frac{10000}{10000}\right) = 0 \qquad \log\left(\frac{10000}{5000}\right) = 0.301$$

$$\log\left(\frac{10000}{20}\right) = 2.698 \qquad \log\left(\frac{10000}{1}\right) = 4$$

Term Weights: tf x idf

- Term frequency (**tf**)
  - the frequency count of a term in a document
- Inverse document frequency (**idf**)
  - The amount of information contained in the statement "Document X contains the term T_i".
- We want to combine **tf** and **idf** for term weighting
- Simplest way:
  - Assign **tf** x **idf** weight to each term in each document

tf x idf

$$w_{ik} = tf_{ik} \times log(M / df_k)$$

$$C \text{ is the collection of documents}$$

$$T_k = term\ k$$

$$tf_{ik} = frequency\ of\ term\ T_k\ in\ document\ D_i$$

$$idf_k = log\left(\frac{M}{df_k}\right)\ inverse\ document\ frequency\ of\ term\ T_k\ in\ C$$

$$M = total\ number\ of\ documents\ in\ the\ collection\ C$$

$$df_k = the\ number\ of\ documents\ in\ C\ that\ contain\ T_k$$

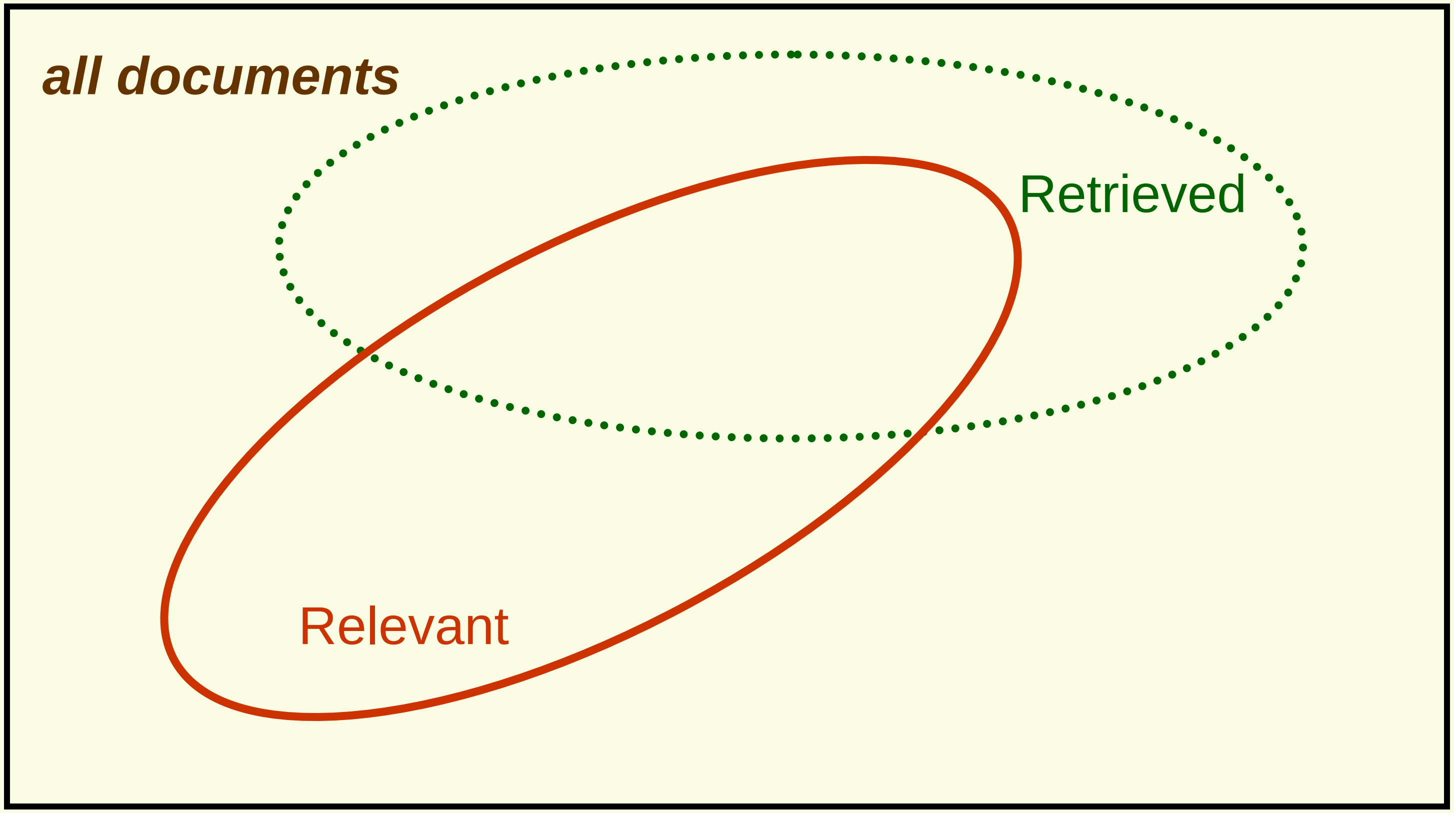
Analysis of the Vector Space Model

- advantages:
  - Simple and effective
  - term-weighting scheme improves retrieval performance
  - partial matching allows for retrieval of documents that approximate the query
  - cosine ranking allows for sorting the results
- disadvantages
  - no real theoretical basis for the assumption of a term space
  - Assumed independence between terms is not really true
- Note: In WWW search engines the weights may be calculated differently
  - use heuristics on where a term occurs in the document (ex, title)
  - notion of *hub* and *authority*
  - …

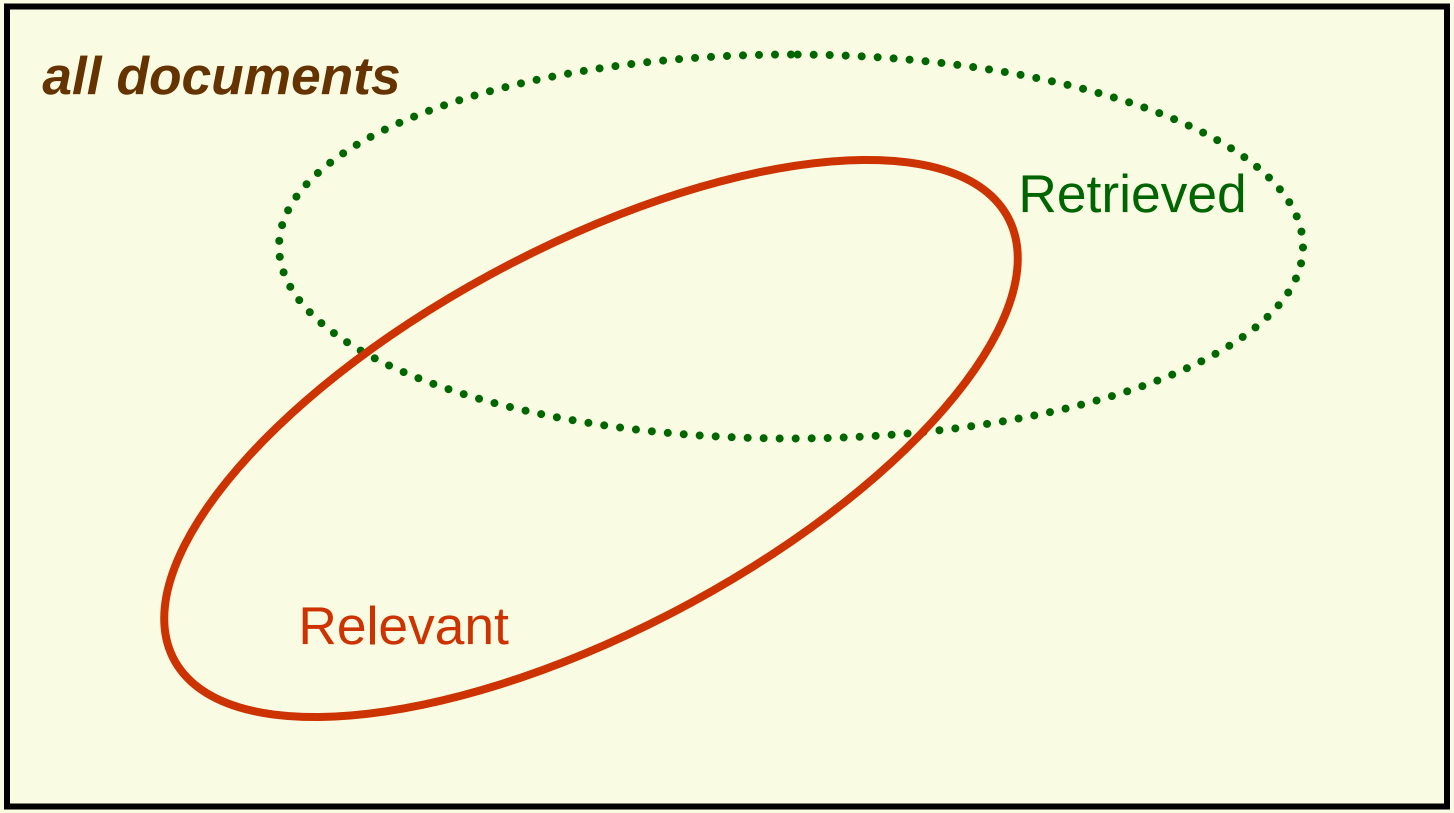
Evaluation

- Suppose you have several retrieval methods. Which one works the best?
  - For us, “best” = effectiveness
  - Other possible measures: ease of use, efficiency, nice interface, etc.
- To evaluate, we need
  - A set of documents
  - A set of queries
  - A set of relevance query/document judgments
- To compare two (or more) methods
  - Each method is used to retrieve documents relevant for queries
  - Results are compared using some measures
  - Common measures are based on **precision** and **recall**

Relevant vs. Retrieved



Precision vs. Recall



$$Precision = \frac{number\ \ of\ relevant\ documents\ retrieved}{number\ of\ documents\ retrieved}$$

$$Recall = \frac{number\ of\ relevant\ documents\ retrieved}{number\ of\ relevant\ documents\ in\ collection}$$

Evaluation: Example of P&R

- Relevant: d_3 d_5 d_9 d_{25} d_{39} d_{44} d_{56} d_{71} d_{123} d_{389}
- system1: d_{123} d_{84} d_{56}
  - Precision : ??
  - Recall : ??
- system2: d_{123} d_{84} d_{56} d_6 d_8 d_9
  - Precision : ??
  - Recall : ??

Evaluation: Example of P&R

- Relevant: d_3 d_5 d_9 d_{25} d_{39} d_{44} d_{56} d_{71} d_{123} d_{389}

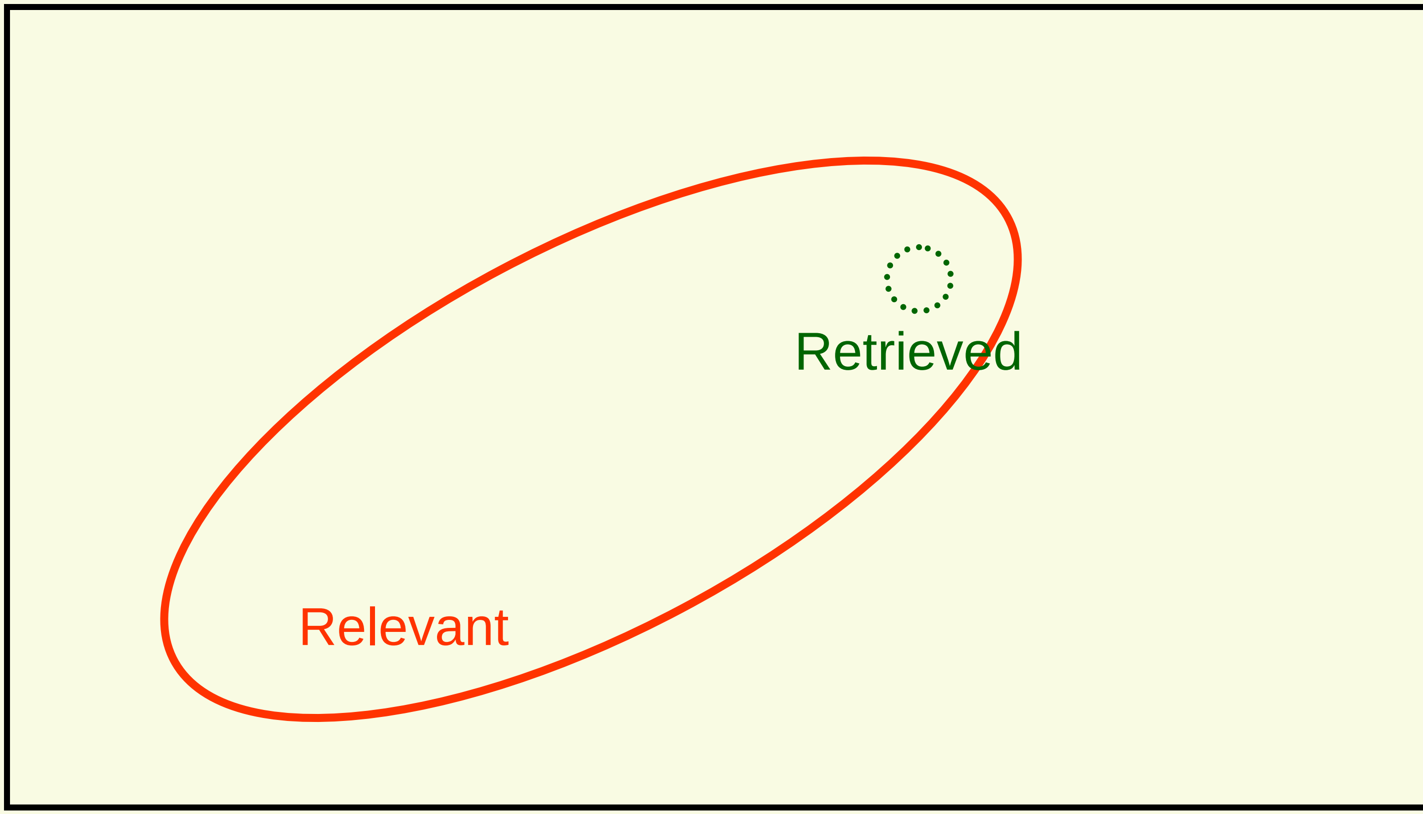
- system1: d_{123}√ d_{84} × d_{56}√
  - Precision: 66% (2/3)
  - Recall: 20% (2/10)
- system2: d_{123}√ d_{84}× d_{56}√ d_6× d_8× d_9√
  - Precision: 50% (3/6)
  - Recall: 30% (3/10)

Why Precision and Recall?

- Get as much good stuff (high recall) while at the same time getting as little junk as possible (high precision)

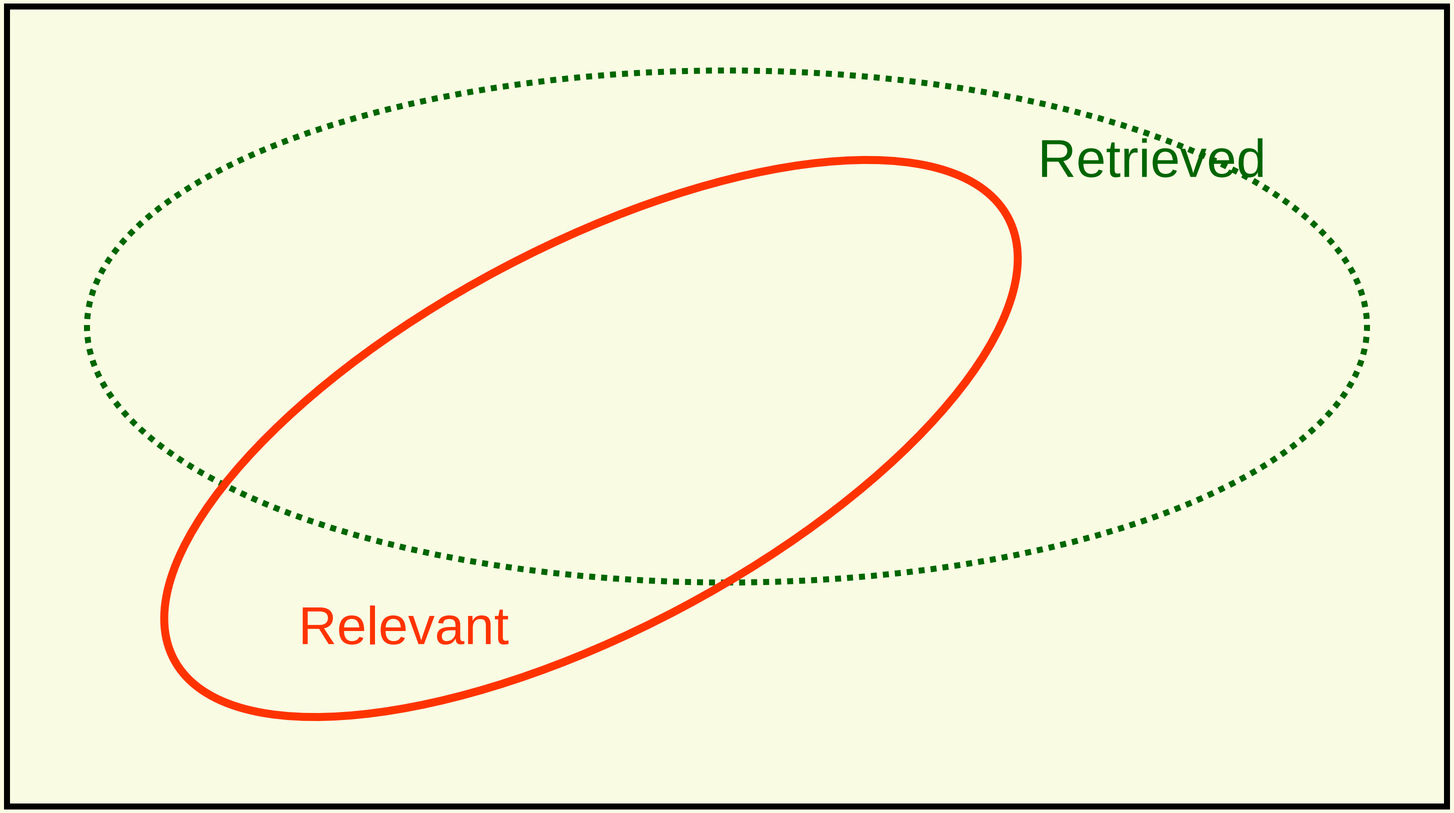
Retrieved vs. Relevant Documents

very high precision, very low recall



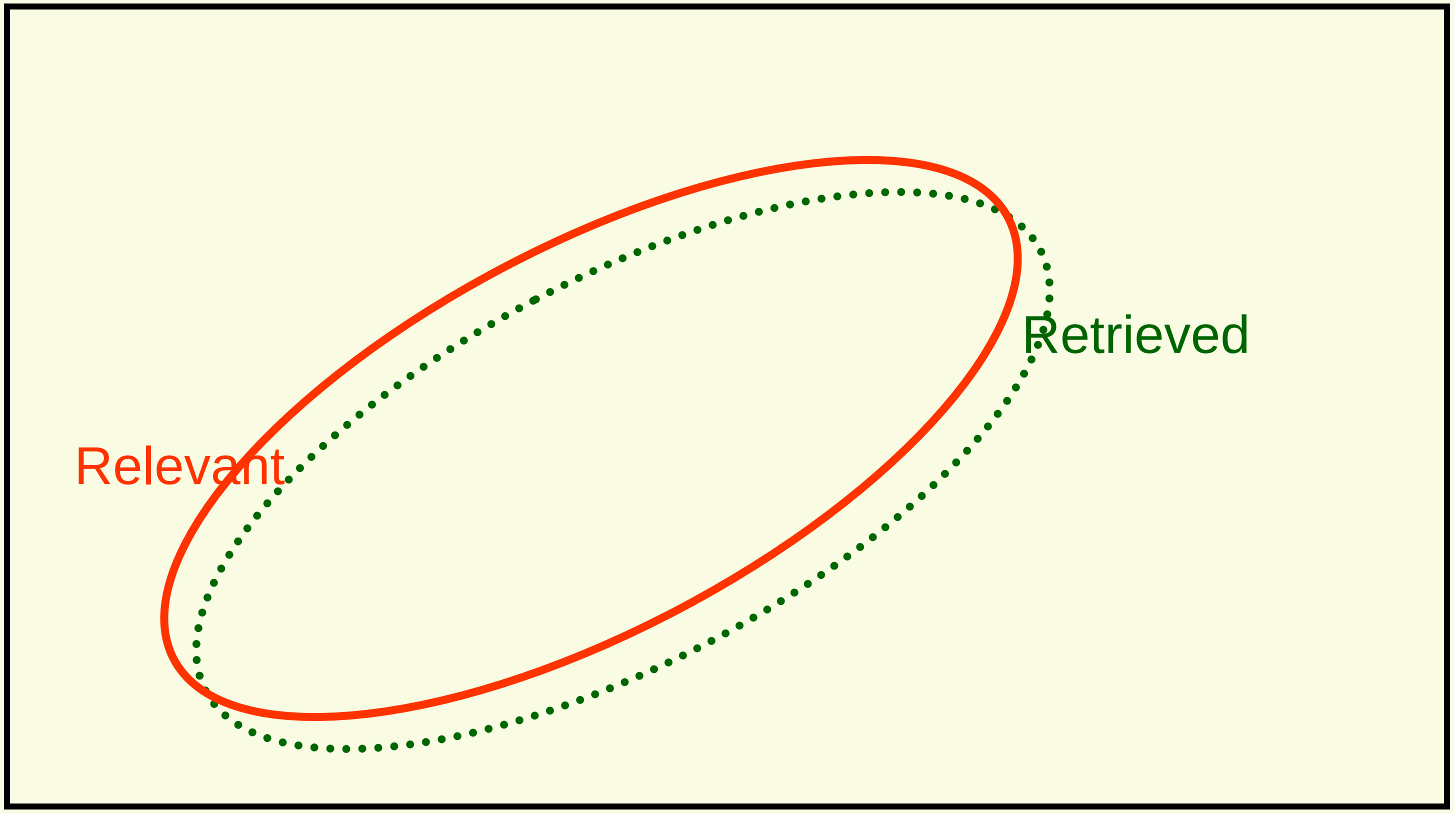
Retrieved vs. Relevant Documents

high recall, but low precision



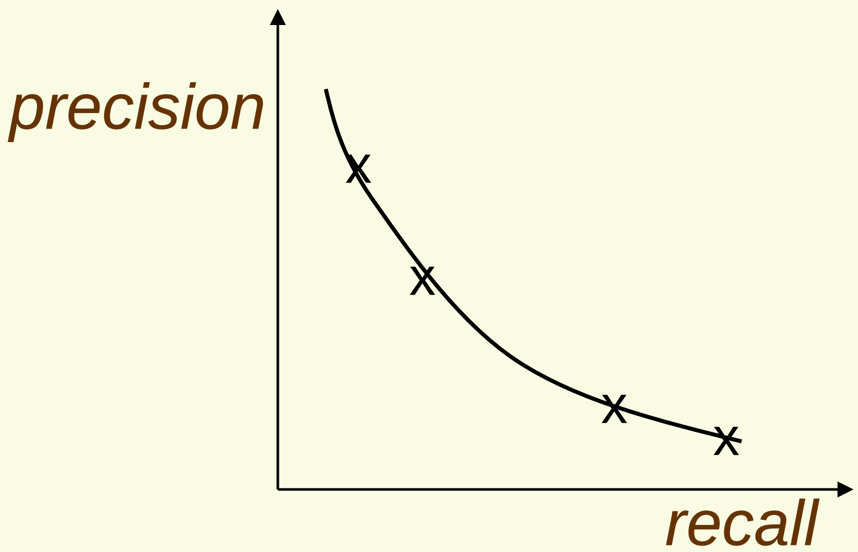
Retrieved vs. Relevant Documents

high precision, high recall (at last!)



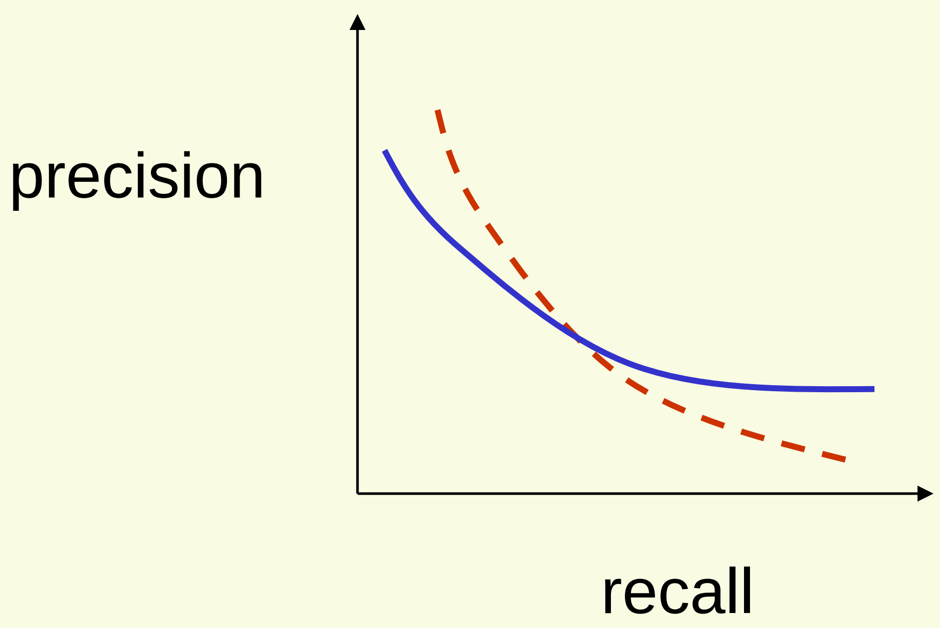
Precision/Recall Curves

- There is a tradeoff between Precision and Recall
  - Easy to get either high precision or high recall, but not both
- So measure Precision at different levels of Recall
- Note: this is an AVERAGE over MANY queries



Precision/Recall Curves

- Difficult to determine which of these two hypothetical results is better:
  - Is blue method performing better than the red one?



Importance of Ranking

- IR systems typically output a ranked list of documents
- Should take "relevance" into account when measuring performance
- The three systems have same precision/recall rates, but the method in the first column is better since it ranks the relevant documents higher

system 1	*system 2*	*system 3*
d1 √	d10 ×	d6 ×
d2 √	d9 ×	d1 √
d3 √	d8 ×	d2 √
d4 √	d7 ×	d10 ×
d5 √	d6 ×	d9 ×
d6 ×	d1 √	d3 √
d7 ×	d2 √	d5 √
d8 ×	d3 √	d4 √
d9 ×	d4 √	d7 ×
d10 ×	d5 √	d8 ×

Cutoff

- Look at precision of the top 5 (or 10, … etc) ranked documents

	system 1	system 2	system 3
	d1 √	d10 ×	d6 ×
	d2 √	d9 ×	d1 √
	d3 √	d8 ×	d2 √
	d4 √	d7 ×	d10 ×
	d5 √	d6 ×	d9 ×
	d6 ×	d1 √	d3 √
	d7 ×	d2 √	d5 √
	d8 ×	d3 √	d4 √
	d9 ×	d4 √	d7 ×
	d10 ×	d5 √	d8 ×
precision at 5	1.0	0.0	0.4
precision at 10	0.5	0.5	0.5

- How to decide on the “cut off” threshold?
  - Threshold 5 is informative in this example, threshold 10 is not informative

Uninterpolated Average Precision

- Instead of using a single "cut off", average precision at many "cut off" points
  - Usually at points where a relevant document is found

for system 3

- At cutoff **d1**: 2 retrieved, 1 relevant, precision ½
- At cutoff **d2**: 3 retrieved, 2 relevant, precision 2/3
- ………………
- At cutoff **d4**: 8 retrievd, 5 relevant, precision 5/8
- Average precision 0.5726

	system 1	*system 2*	*system 3*	
	d1 √	d10 ×	d6 ×	
	d2 √	d9 ×	d1 √	**1/2**
	d3 √	d8 ×	d2 √	**2/3**
	d4 √	d7 ×	d10 ×	
	d5 √	d6 ×	d9 ×	
	d6 ×	d1 √	d3 √	**3/6**
	d7 ×	d2 √	d5 √	**4/7**
	d8 ×	d3 √	d4 √	**5/8**
	d9 ×	d4 √	d7 ×	
	d10 ×	d5 √	d8 ×	
precision at 5	1.0	0.0	0.4	
precision at 10	0.5	0.5	0.5	
aver. precision	1.0	0.3544	0.5726	

F-Measure

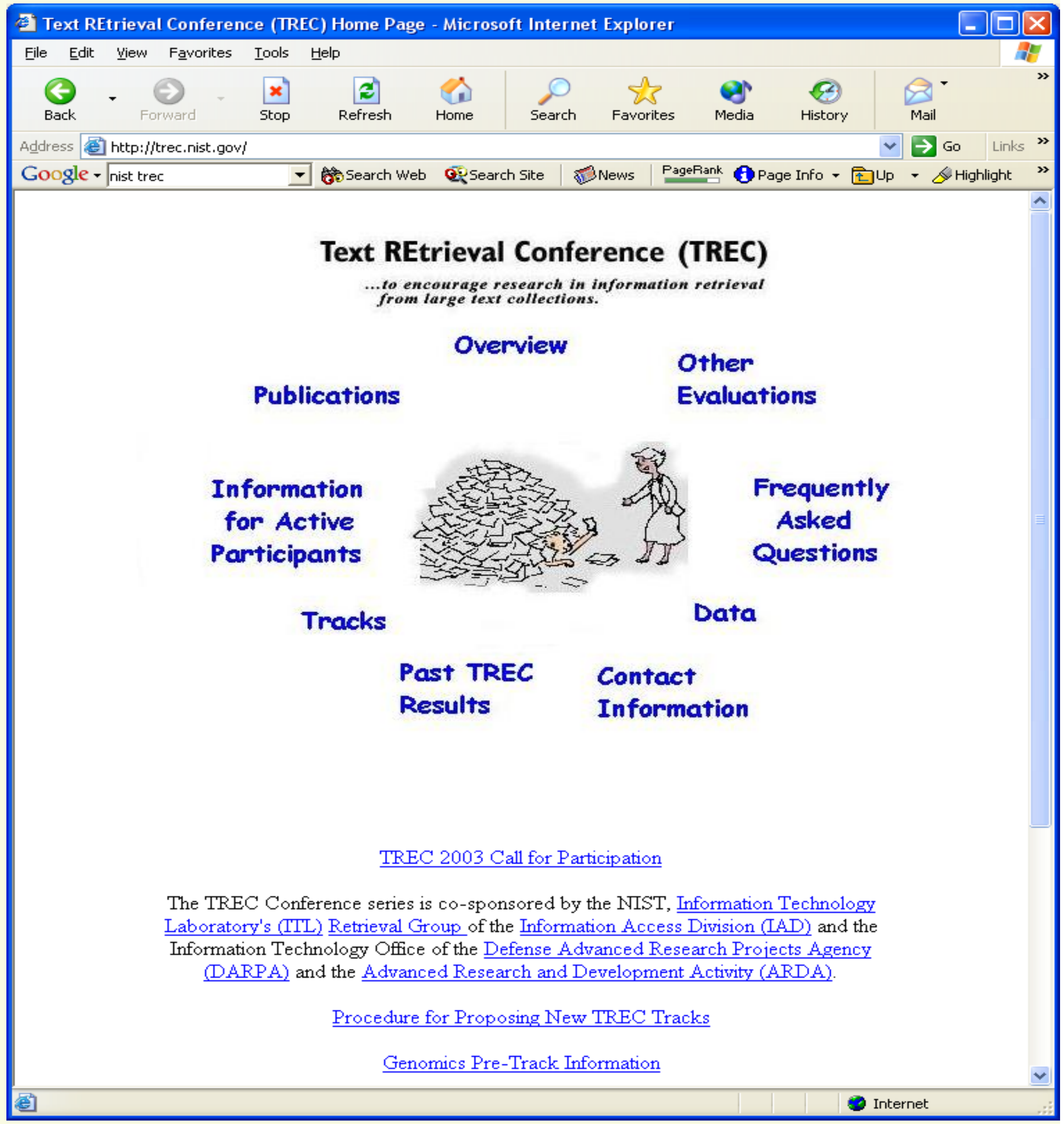
- Sometime only one pair of precision and recall is available
  - e.g., filtering task
- F-Measure

$$F = \frac{1}{\alpha \frac{1}{P} + (1-\alpha)\frac{1}{R}}$$

  - α > 0.5: precision is more important
  - α < 0.5: recall is more important
  - Usually α = 0.5

Evaluation: TREC

- Text Retrieval Conference/competition
- Collection: about 3 Gigabytes > 1 million documents
  - Newswire & text news (AP, WSJ,…)
- Queries + relevance judgements
  - Queries devised and judged by annotators
- Participants
  - Various research and commercial group
- Tracks
  - Cross-lingual, filtering, genome, video, web, QA, etc.



IR System Improvements

- Most Queries are short
  - Web queries tend to be 2-3 keywords long
- The two big problems with short queries are:
  - Synonymy: poor recall results from missing documents that contain synonyms of search terms, but not the terms themselves
  - Polysemy/Homonymy: Poor precision results from search terms that have multiple meanings leading to the retrieval of non-relevant documents

Query Expansion

- Find a way to expand a user's query to automatically include relevant terms (that they should have included themselves), in an effort to improve recall
  - Use a dictionary/thesaurus
  - Use relevance feedback

Query Expansion

- Example:
  - query: *seller of email solutions for cell phones*
  - document: […] *Giszmotron is a leading vendor of electronic messaging services for cellular devices* […]

- But effect of polysemy on IR:
  - *cell* --> *a prison room* or *a unit* ?

  --> returning irrelevant documents

  --> decrease precision

- Effects of synonymy and hyponymy on IR

  --> missing relevant documents

  --> decrease recall

- Solution: let's expand the user query with related terms
  - often using a thesaurus to find related terms (synonyms, hyponyms)
  - new terms will have lower weights in the query
  - ex: expanded query: *seller vendor phones device …*
  - need to do WSD

Relevance Feedback

- Ask the user to identify a few documents which appear to be related to their information need
- Extract terms from those documents and add them to the original query
- Run the new query and present those results to the user
- Iterate (ask the user to identify relevant documents…extract terms… add them to the query…)
  - Typically converges quickly

Blind Feedback

- Assume that first few documents returned are most relevant rather than having users identify them
- Proceed as for relevance feedback
- Tends to improve recall at the expense of precision

Additional IR Issues

- In addition to improved relevance, can improve overall information retrieval with some other factors:
  - Eliminate duplicate documents
  - Provide good context
- For the web:
  - Eliminate multiple documents from one site
  - Clearly identify paid links

IR within NLP

- IR needs to process the large volumes of online text
- And (traditionally), NLP methods were not *robust* enough to work on thousands of real world texts.
- so IR:
  - not based on NLP tools (ex. syntactic/semantic analysis)
  - uses (mostly) simple (shallow) techniques
  - based mostly on word frequencies
- in IR, meaning of documents:
  - is the composition of meaning of individual words
  - ordering & constituency of words play are not taken into account
  - *bag of word* approach

    I see what I eat.
    I eat what I see.
    same meaning

Summary

- Information Retrieval is the process of returning documents from unstructured data collection to meet a user’s information need based on a query
- Typical methods are BOW (bag of words) which rely on keyword indexing with little semantic processing
- Results can be improved by adding semantic information (such as thesauri) and by filtering and other post-hoc analysis.