CSEN1076: NATURAL LANGUAGE PROCESSING AND INFORMATION RETRIEVAL

LECTURE 3 –VECTOR SPACE MODEL AND IR EVALUATION

MERVAT ABUELKHEIR

VECTOR SPACE MODEL

RECALL: BINARY → COUNT → WEIGHT MATRIX

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
ANTHONY	5.25	3.18	0	0	0	0.35
BRUTUS	1.21	6.1	0	1	0	0
CAESAR	8.59	2.54	0	1.51	0.25	0
CALPURNIA	0	1.54	0	0	0	0
CLEOPATRA	2.85	0	0	0	0	0
MERCY	1.51	0	1.9	0.12	5.25	0.88
WORSER	1.37	0	0.11	4.15	0.25	0

Each document is now represented by **a real-valued vector** of *tf-idf* weights in $\mathbb{R}^{|V|}$

Final ranking of documents for a query → $score(q,d) = \sum_{t \in q \cap d} tf.idf_{t,d}$

DOCUMENTS AS VECTORS

Now we have a $|V|$-dimensional vector space

The representation of a set of documents as vectors in a **common** vector space is known as the **vector space model**

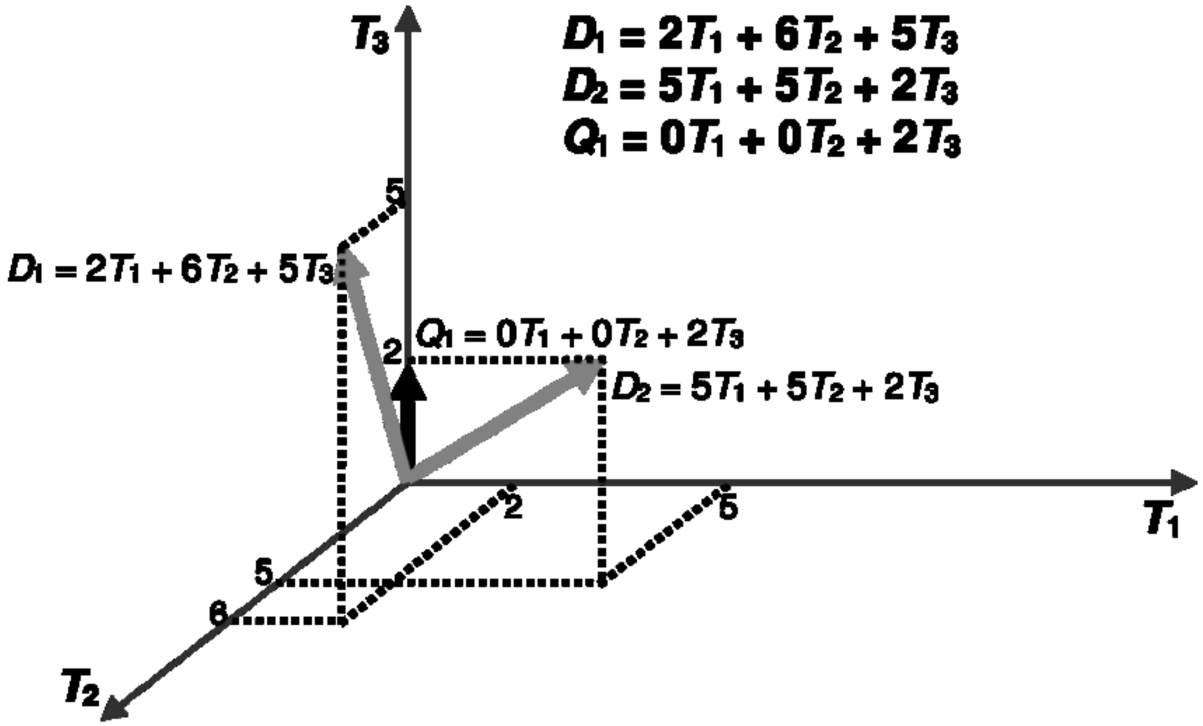
Terms are **axes** (**dimensions**) of the space

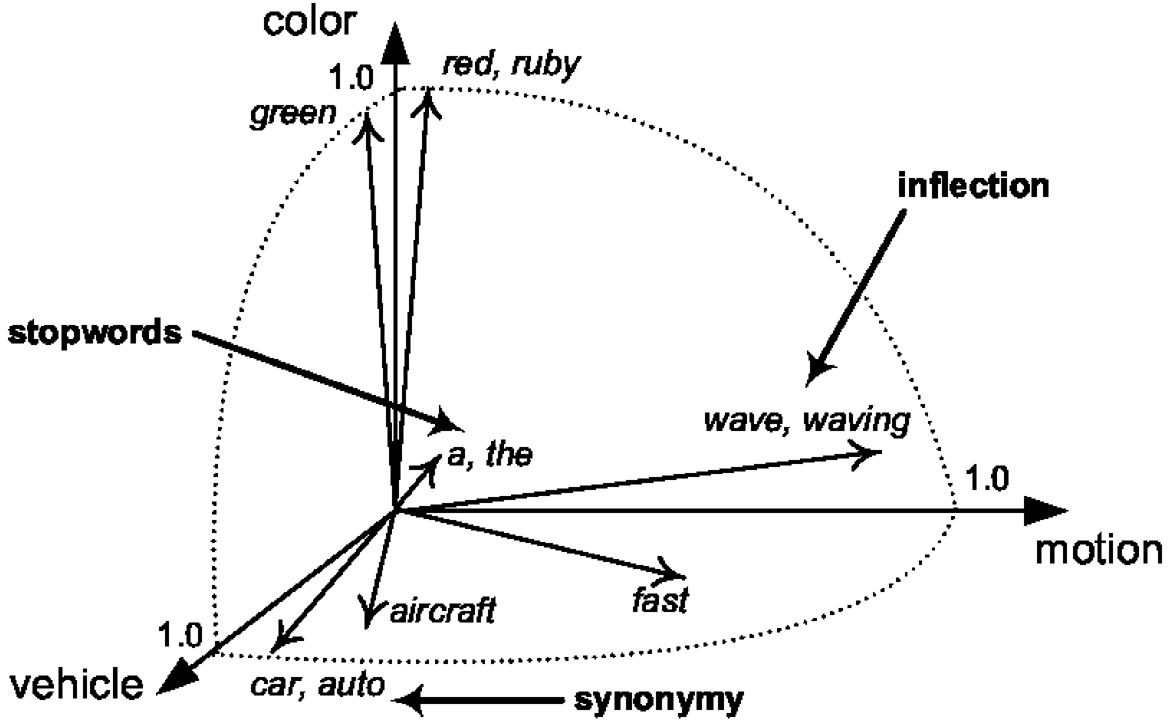
Documents are points or **vectors** in this space

Very high-dimensional: **tens of millions of dimensions** when you apply this to a web search engine

These are very sparse vectors – **most entries are zero**

DOCUMENTS AS VECTORS





Images source: https://www.esearchgate.net

QUERIES AS VECTORS

Key idea 1: Do the same for queries: represent them as vectors in the space

Key idea 2: Rank documents according to their proximity to the query in this space

Proximity = similarity of vectors

Proximity → inverse of distance

This allows us to rank relevant documents higher than non-relevant documents

FORMALIZING VECTOR SPACE PROXIMITY

First attempt: distance between two points

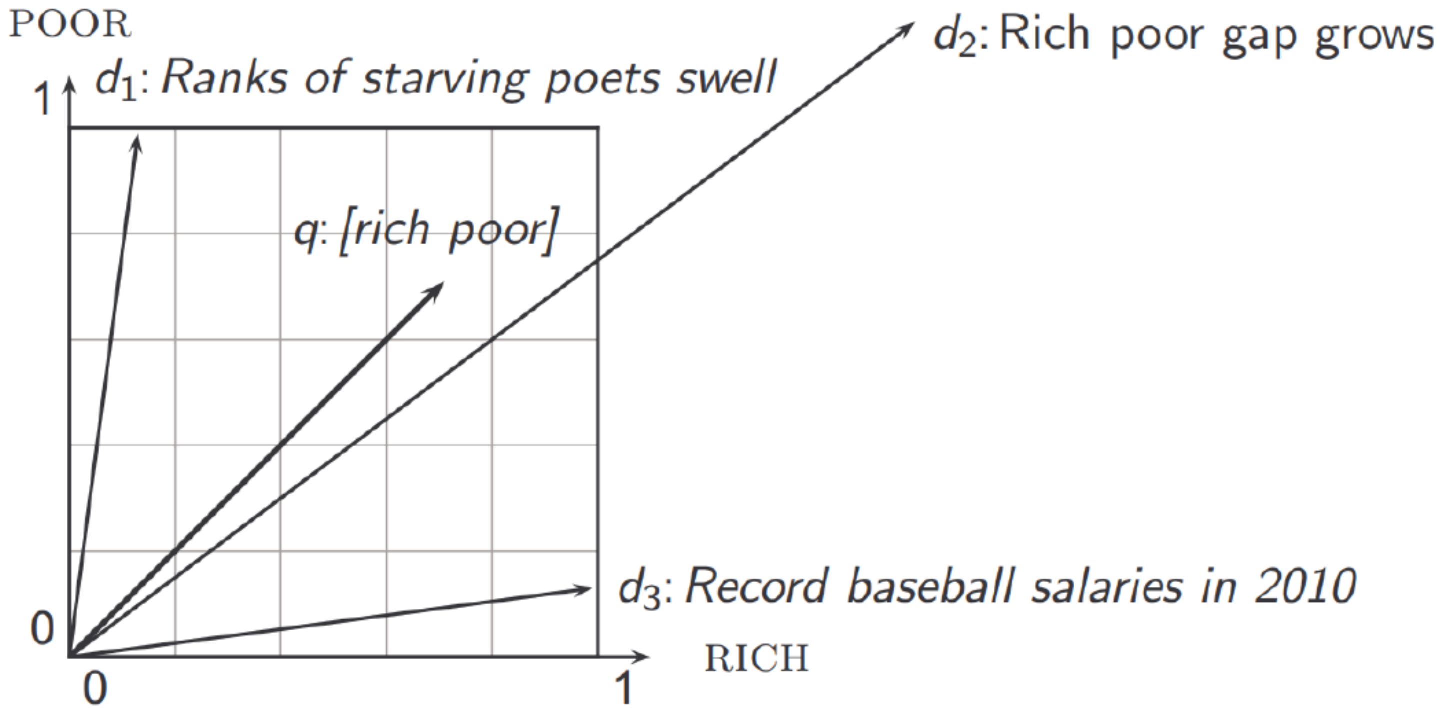
- (= distance between the end points of the two vectors)

Euclidean distance?

Euclidean distance is a bad idea . . .

. . . because Euclidean distance is large for vectors of different lengths

WHY POINT DISTANCE IS A BAD IDEA



The Euclidean distance between $\vec{q}$ and $\overrightarrow{d_2}$ is large even though the distribution of terms in the query $\vec{q}$ and the distribution of terms in the document $\overrightarrow{d_2}$ are very similar

USE **ANGLE** INSTEAD OF POINT **DISTANCE**

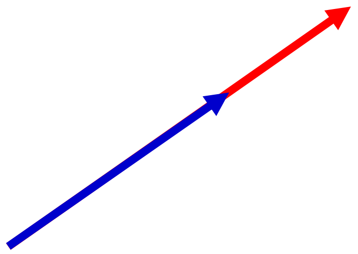
Thought experiment: take a document d and append it to itself. Call this document d'

"Semantically" d and d' have the same content

BUT the Euclidean distance between the two documents can be quite large

The angle between the two documents is 0, corresponding to maximal similarity

Key idea: Rank documents according to angle with query

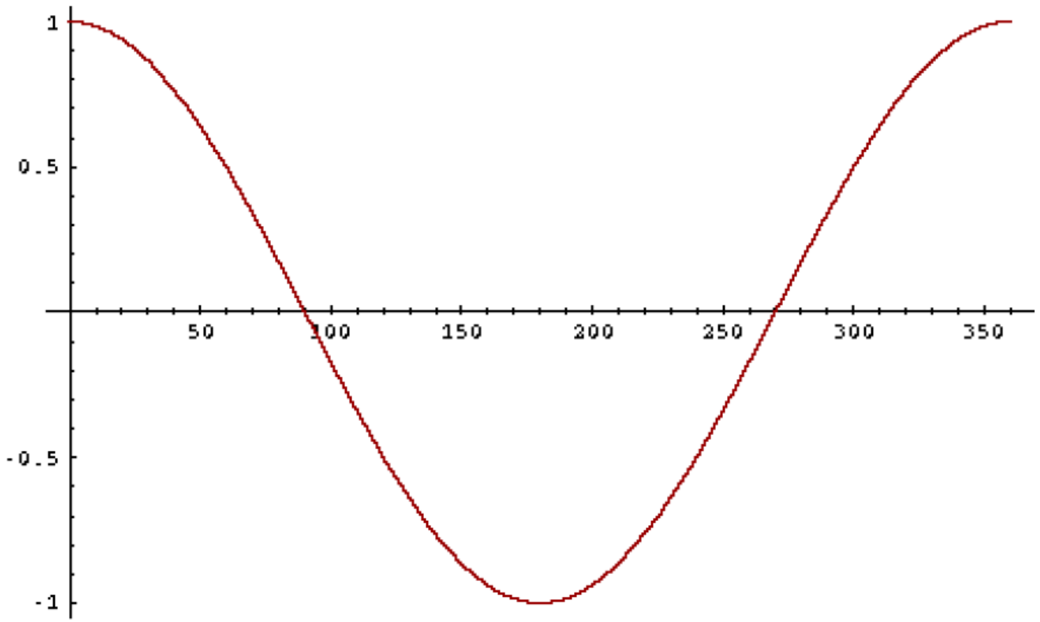


FROM ANGLES TO COSINES

The following two notions are equivalent

- Rank documents in decreasing order of the angle between query and document
- Rank documents in increasing order of $cosine(query, document)$

Cosine is a decreasing function for the interval [0º, 180º]



But how – and why – should we be computing cosines?

HOW – COSINE SIMILARITY BETWEEN QUERY AND DOCUMENT

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

q_i is the *tf-idf* weight of term i in the query

d_i is the *tf-idf* weight of term i in the document

$\cos(\vec{q},\vec{d})$ is the cosine similarity of q and d … or, equivalently, the cosine of the angle between q and d

You can length-normalize vectors before computing the cosine similarity

$\cos(\vec{q},\vec{d}) = \vec{q}\cdot\vec{d} = \sum_{i=1}^{|V|} q_i \times d_i$, and q and d are length-normalized

WHY – LENGTH NORMALIZATION

A vector can be (length-) normalized by dividing each of its components by its length – for this we use the norm:

$$\|\vec{x}\| = \sqrt{\sum_i x_i^2}$$

Dividing a vector by its norm makes it a unit (length) vector (on surface of unit hypersphere)

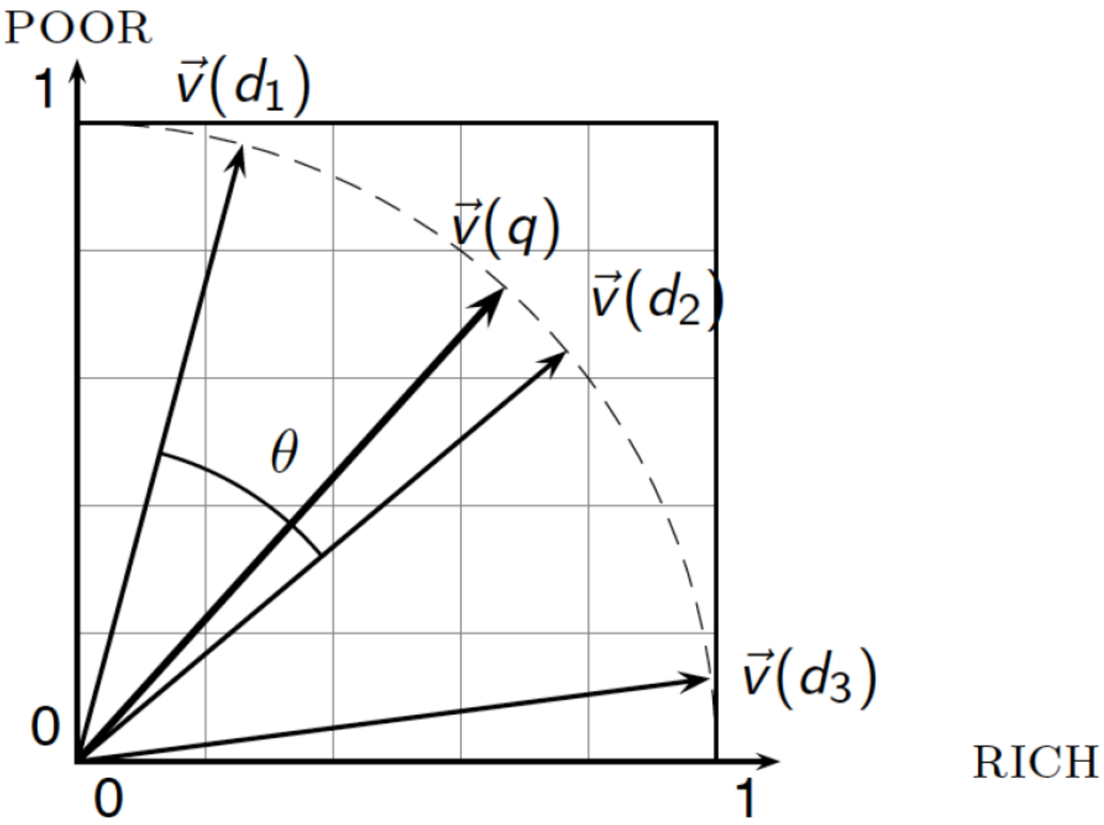
- Also known as cosine normalization

As a result, longer documents and shorter documents have weights of the same order of magnitude

Effect on the two documents d and d' (d appended to itself): they have identical vectors after length-normalization

- **Long and short documents now have comparable weights**

COSINE SIMILARITY ILLUSTRATED



COSINE USED FOR DOCUMENTS SIMILARITY

We can use cosine similarity to discover similar documents

How similar are these novels?

SaS: Sense and Sensibility

PaP: Pride and Prejudice

WH: Wuthering Heights

term	SaS	PaP	WH
affection	115	58	20
jealous	10	7	11
gossip	2	0	6
wuthering	0	0	38

Term frequencies matrix (raw counts)

Note: To simplify this example, we don't do *idf* weighting

3 DOCUMENTS EXAMPLE (CONT.)

Log frequency weighting

term	SaS	PaP	WH
affection	3.06	2.76	2.30
jealous	2.00	1.85	2.04
gossip	1.30	0	1.78
wuthering	0	0	2.58

After length normalization

term	SaS	PaP	WH
affection	0.789	0.832	0.524
jealous	0.515	0.555	0.465
gossip	0.335	0	0.405
wuthering	0	0	0.588

$\cos(SaS, PaP) \approx$

$$0.789 \times 0.832 + 0.515 \times 0.555 + 0.335 \times 0.0 + 0.0 \times 0.0 \approx 0.94$$

$\cos(SaS, WH) \approx 0.79$

$\cos(PaP, WH) \approx 0.69$

Why do we have $\cos(SaS, PaP) > \cos(SaS, WH)$?

COMPUTING THE COSINE SCORE

COSINESCORE(q)

float $Scores[N] = 0$
Initialize $Length[N]$
for each query term t
do calculate $w_{t,q}$ and fetch postings list for t
    for each pair(d, $tf_{t,d}$) in postings list
    do $Scores[d] += wf_{t,d} \times w_{t,q}$
Read the array $Length[d]$
for each d
do $Scores[d] = Scores[d] / Length[d]$
return Top K components of $Scores[]$

tf-idf WEIGHTING HAS MANY VARIANTS

Term frequency		Document frequency		Normalization	
n (natural)	$\text{tf}_{t,d}$	n (no)	1	n (none)	1
l (logarithm)	$1 + \log(\text{tf}_{t,d})$	t (idf)	$\log \frac{N}{\text{df}_t}$	c (cosine)	$\frac{1}{\sqrt{w_1^2 + w_2^2 + \ldots + w_M^2}}$
a (augmented)	$0.5 + \frac{0.5 \times \text{tf}_{t,d}}{\max_t(\text{tf}_{t,d})}$	p (prob idf)	$\max\{0, \log \frac{N - \text{df}_t}{\text{df}_t}\}$	u (pivoted unique)	$1/u$
b (boolean)	$\begin{cases} 1 & \text{if } \text{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$			b (byte size)	$1/\mathit{CharLength}^{\alpha}$, $\alpha < 1$
L (log ave)	$\frac{1 + \log(\text{tf}_{t,d})}{1 + \log(\text{ave}_{t \in d}(\text{tf}_{t,d}))}$				

WEIGHTING MAY DIFFER IN QUERIES VS DOCUMENTS

Many search engines allow for different weightings for queries vs. documents

SMART Notation: denotes the combination in use in an engine, with the notation *ddd.qqq*, using the acronyms from the previous table

A very standard weighting scheme is: *lnc.ltc*

Document: logarithmic *tf* (*l* as first character), no *idf* and cosine normalization

Query: logarithmic *tf* (*l* in leftmost column), *idf* (*t* in second column), cosine normalization

tf-idf EXAMPLE: *lnc.ltc*

Query: "best car insurance". Document: "car insurance auto insurance"

Collection consists of 1,000,000 documents

Term	Query						Document				Prod
	tf-raw	tf-wt	df	idf	wt	n'lize	tf-raw	tf-wt	wt	n'lize	
auto	0	0	5000	2.3	0	0	1	1	1	0.52	0
best	1	1	50000	1.3	1.3	0.34	0	0	0	0	0
car	1	1	10000	2.0	2.0	0.52	1	1	1	0.52	0.27
insurance	1	1	1000	3.0	3.0	0.78	2	1.3	1.3	0.68	0.53

Score = 0+0+0.27+0.53 = 0.8

tf-idf EXAMPLE: *lnc.ltn*

Query: “best car insurance”. Document: “car insurance auto insurance”

Collection consists of 1,000,000 documents

Term	Query						Document				Prod
	tf-raw	tf-wt	df	idf	wt		tf-raw	tf-wt	wt	n’lize	
auto	0	0	5000	2.3	0		1	1	1	0.52	0
best	1	1	50000	1.3	1.3		0	0	0	0	0
car	1	1	10000	2.0	2.0		1	1	1	0.52	1.04
insurance	1	1	1000	3.0	3.0		2	1.3	1.3	0.68	2.04

Score = 0+0+1.04+20.4 = 3.08

SUMMARY – VECTOR SPACE RANKING

Represent the query as a weighted *tf-idf* vector

Represent each document as a weighted *tf-idf* vector

Compute the cosine similarity score for the query vector and each document vector

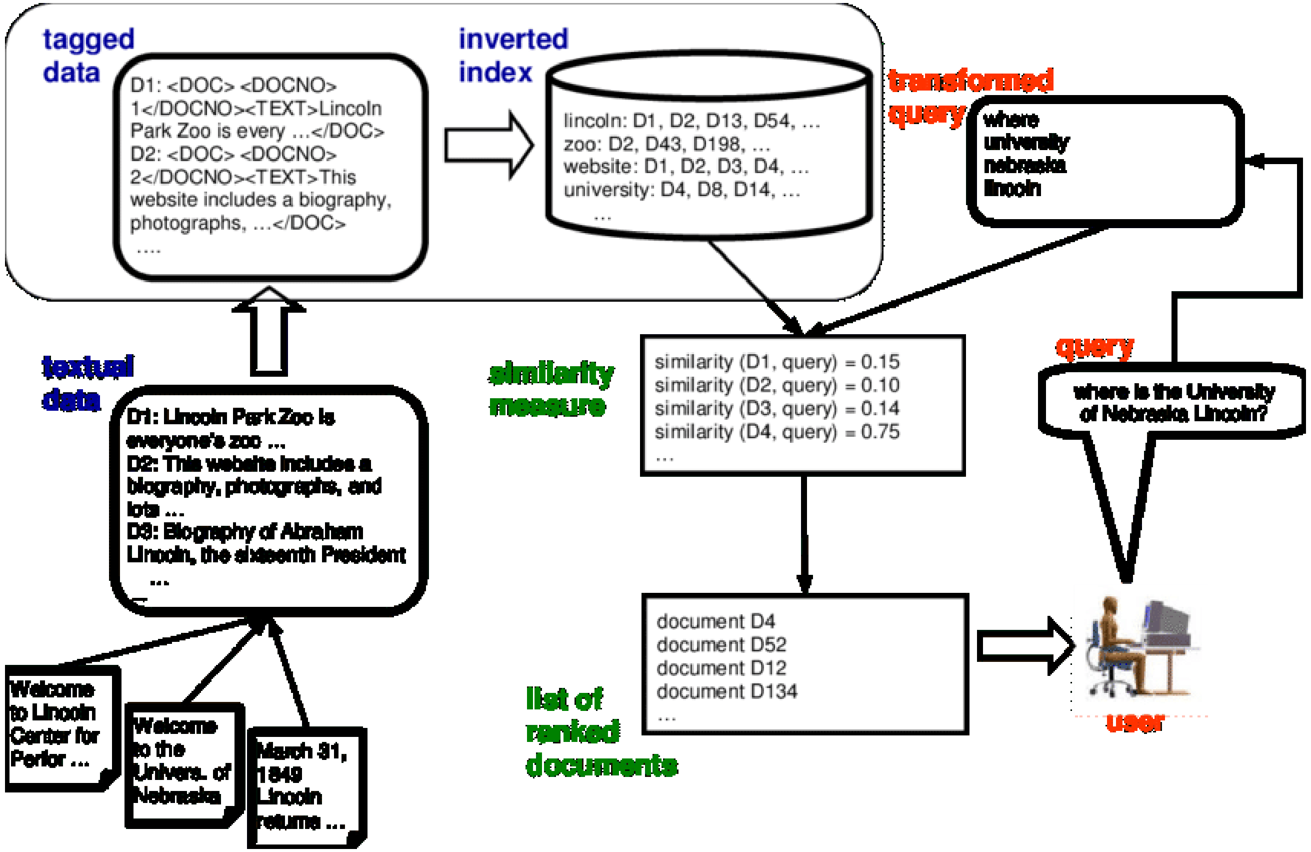
Rank documents with respect to the query by score

Return the top K (e.g., $K = 10$) to the user

Check the cosine similarity tutorial at http://www.miislita.com/information-retrieval-tutorial/cosine-similarity-tutorial.html

EVALUATION OF IR SYSTEMS

BASIC IR ARCHITECTURE



MEASURES FOR A SEARCH ENGINE

How fast does it index?

- Number of documents/hour
- (Average document size)

How fast does it search?

- Latency as a function of index size

Expressiveness of query language

- Ability to express complex information needs
- Speed on complex queries

Uncluttered UI

Is it free?

MEASURES FOR A SEARCH ENGINE

All of the preceding **criteria are measurable**: we can quantify speed/size

- we can make expressiveness precise

The key measure: **user happiness**

- What is this?
- Speed of response/size of index are factors?
- But blindingly fast, useless answers won’t make a user happy

We need a way of **quantifying user happiness** with the results returned

- Elusive to measure, but we can try
- **Relevance of results to user’s information need**

EVALUATING AN IR SYSTEM

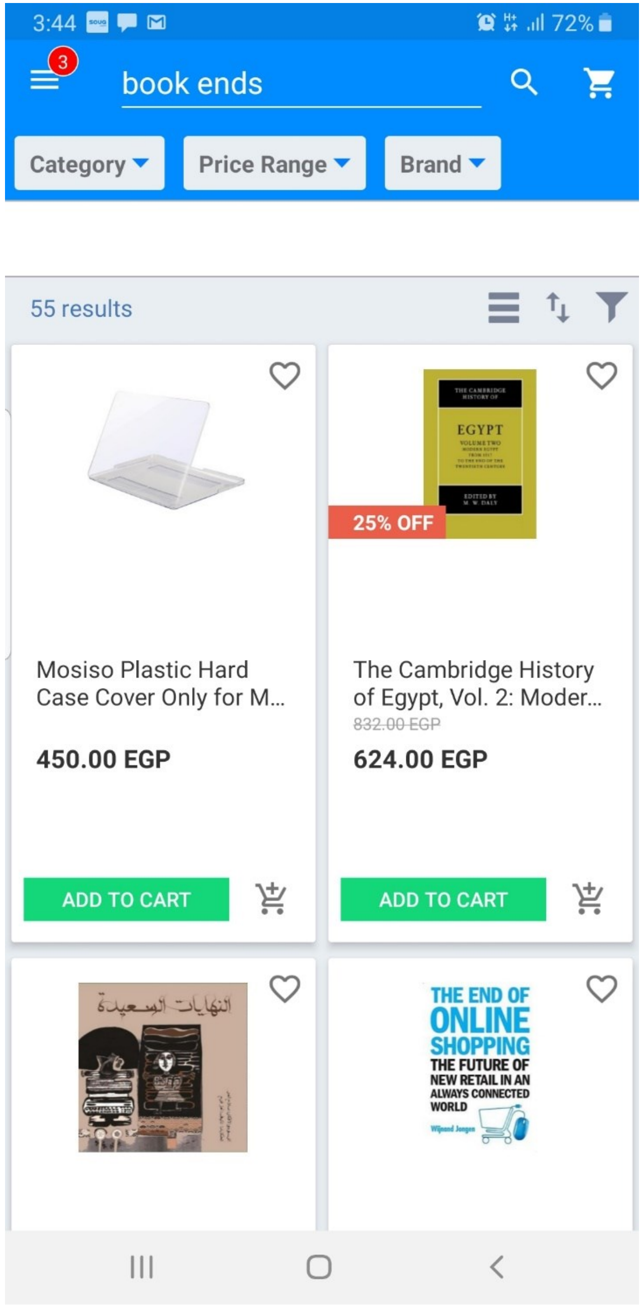
An information need is translated into a query

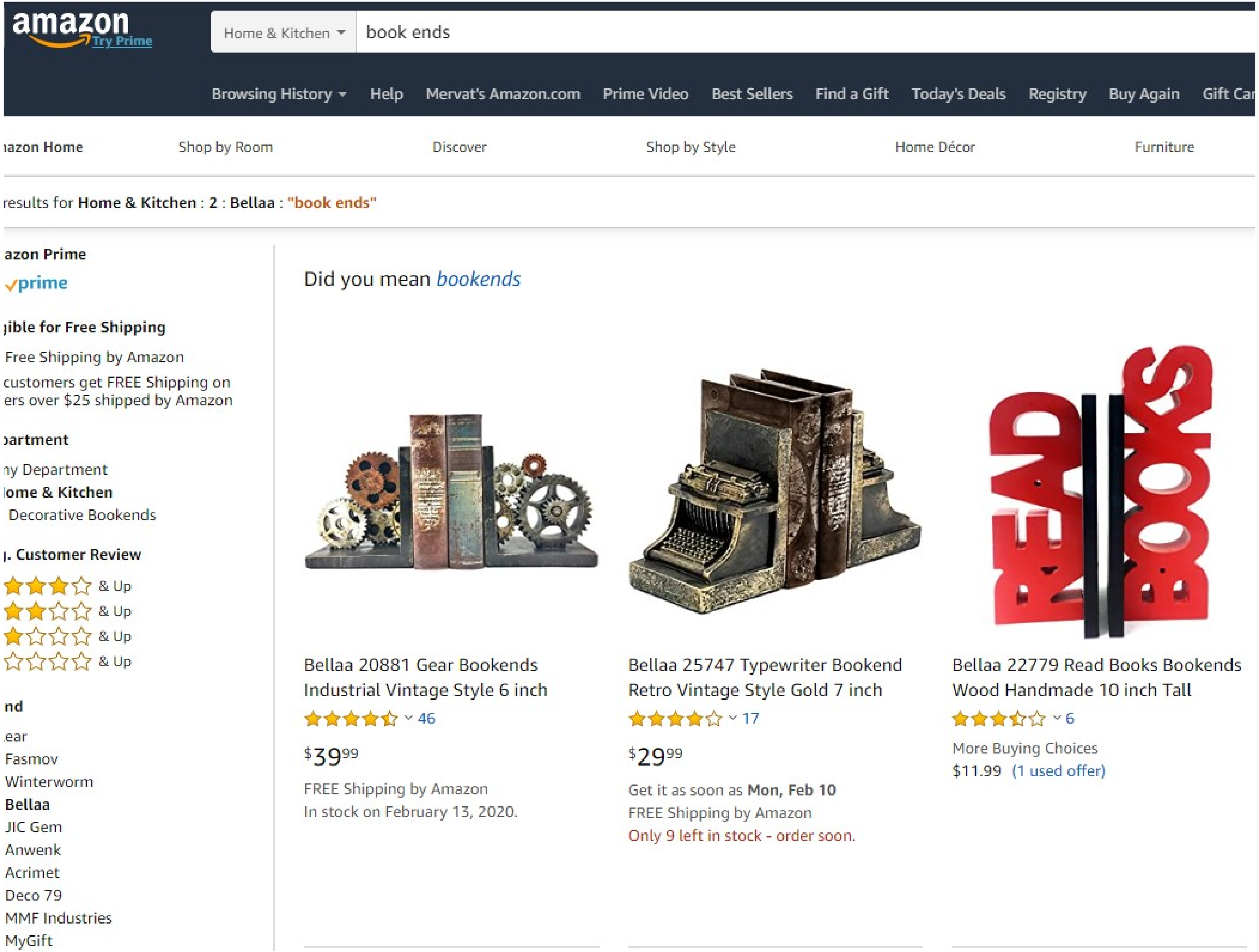
Relevance is assessed **relative to the information need not the query**

e.g., Information need: I'm looking for information on whether drinking green tea is more effective at reducing your risk of heart attacks than coffee

Query: **green tea coffee heart attack effective**

You evaluate whether the document addresses the information need, not whether it has these words


3:44
72%
3
book ends
Category
Price Range
Brand
55 results
25% OFF
Mosiso Plastic Hard Case Cover Only for M...
450.00 EGP
ADD TO CART
The Cambridge History of Egypt, Vol. 2: Moder...
832.00 EGP
624.00 EGP
ADD TO CART
THE END OF ONLINE SHOPPING
THE FUTURE OF NEW RETAIL IN AN ALWAYS CONNECTED WORLD


amazon
Try Prime
Home & Kitchen
book ends
Browsing History
Help
Mervat's Amazon.com
Prime Video
Best Sellers
Find a Gift
Today's Deals
Registry
Buy Again
Shop by Room
Discover
Shop by Style
Home Décor
Furniture
results for Home & Kitchen : 2 : Bellaa : "book ends"
prime
Free Shipping by Amazon
customers get FREE Shipping on
ers over $25 shipped by Amazon
Decorative Bookends
& Up
& Up
& Up
& Up
Fasmov
Winterworm
Bellaa
JIC Gem
Anwenk
Acrimet
Deco 79
MMF Industries
MyGift
Did you mean bookends
Bellaa 20881 Gear Bookends Industrial Vintage Style 6 inch
46
$39.99
FREE Shipping by Amazon
In stock on February 13, 2020.
Bellaa 25747 Typewriter Bookend Retro Vintage Style Gold 7 inch
17
$29.99
Get it as soon as Mon, Feb 10
FREE Shipping by Amazon
Only 9 left in stock - order soon.
Bellaa 22779 Read Books Bookends Wood Handmade 10 inch Tall
6
More Buying Choices
$11.99 (1 used offer)

EVALUATING UNRANKED RESULTS

Collection	Query 1	Query 2	…	Query m
Doc 1	Nonrelevant	Relevant	…	Nonrelevant
Doc 2	Relevant	Relevant	…	Relevant
…	…	…	…	…
Doc n	Nonrelevant	Nonrelevant	…	Relevant

Evaluation of a result set:

- If we have
  - a benchmark document collection
  - a benchmark set of queries
  - An assessment of either *Relevant* or *Nonrelevant* for each query and each document
    - **Usually provided manually by experts** (more on that at the end)
- Then problem is formulated as a **classification problem**
- **Accuracy** is a commonly used evaluation measure in machine learning classification work

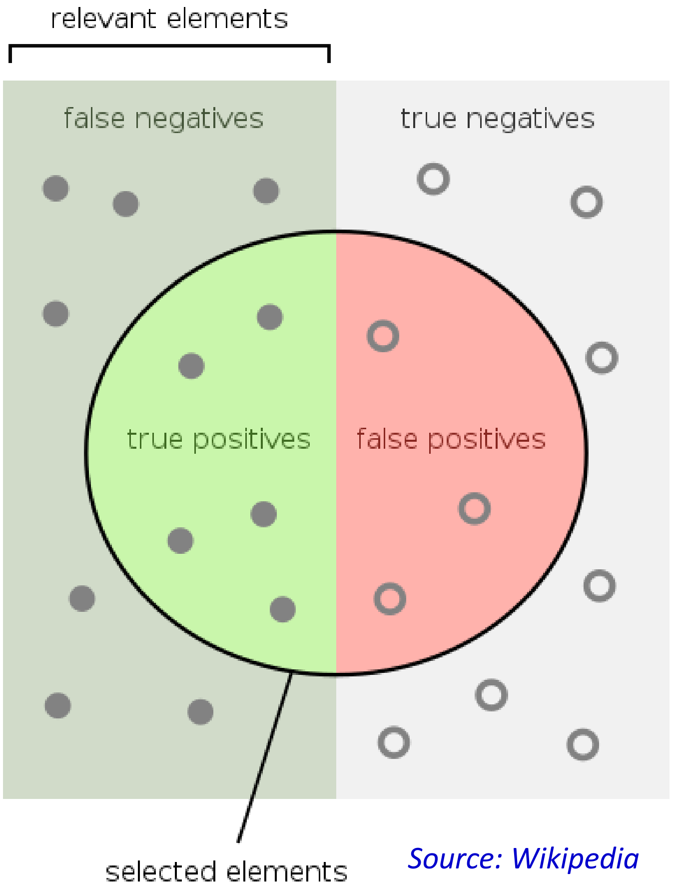
*Why don't we use the **accuracy** metric in IR?*

ACCURACY

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN}$$

		Search Results		
Documents		Retrieved	Not retrieved	Total
	Relevant	**TP**	**FN**	P
	Nonrelevant	**FP**	**TN**	N
	Total	$\hat{P}$	$\hat{N}$	**P + N**

Confusion Matrix



Source: Wikipedia

WHY NOT JUST USE ACCURACY?

$$Accuracy = \frac{TP + \boldsymbol{TN}}{TP + FN + \mathrm{FP} + \mathrm{TN}}$$

Of 100 documents in system, if you have 95 nonrelevant documents and only 5 relevant documents, what does a 97% IR accuracy reflect?

Accuracy is no good measure for unbalanced classification problems

How to build a 99.9999% accurate search engine on a low budget?

- People doing information retrieval *want to find something* and have a certain tolerance for junk

Better measures are **Precision/Recall/F-measure**

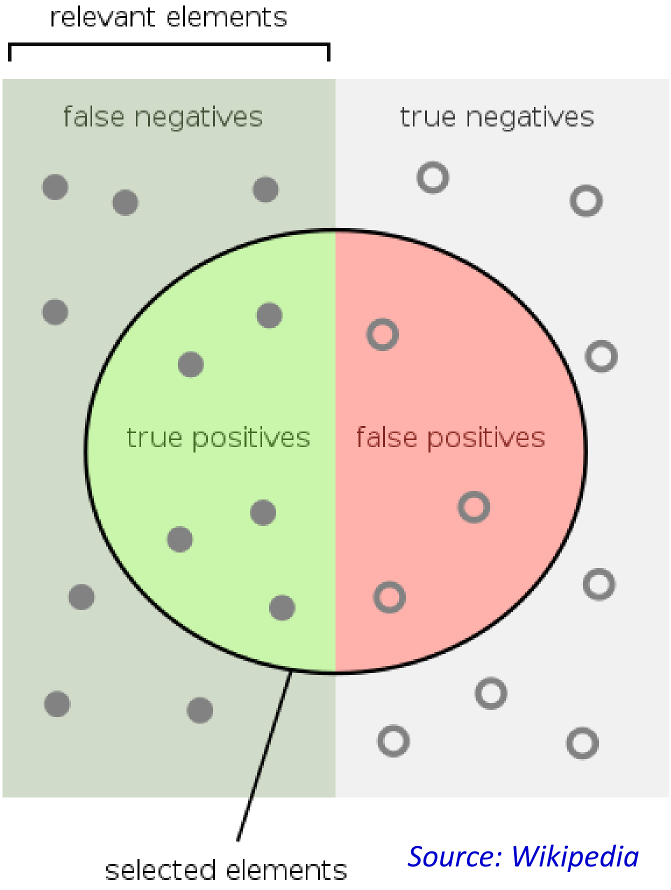
PRECISION RECALL

$Precision = \dfrac{\#\,relevant\ items\ in\ results}{Total\ \#\,items\ in\ results}$	$Recall\ (Sensitivity) = \dfrac{\#\,relevant\ items\ in\ results}{Total\ \#\,relevant\ items\ in\ collection}$
$Precision = \dfrac{TP}{TP+FP}$	$Recall = \dfrac{TP}{TP+FN}$

	Retrieved	Not retrieved	Total
Relevant	**TP**	**FN**	P
Nonrelevant	**FP**	**TN**	N
Total	$\hat{P}$	$\hat{N}$	**P + N**

Recall

Precision



Source: Wikipedia

HOW TO FILL IN THE CONFUSION MATRIX

Collection	Query 1 – Ground Truth	Query 1 – Your IR	Match?
Doc 1	Nonrelevant	Not retrieved	Increase ***TN*** by 1
Doc 2	Relevant	Not retrieved	Increase ***FN*** by 1
Doc 3	Relevant	Retrieved	Increase ***TP*** by 1
Doc 4	Nonrelevant	Retrieved	Increase ***FP*** by 1
Doc 5	Relevant	Retrieved	Increase ***TP*** by 1
Doc 6	Nonrelevant	Not retrieved	Increase ***TN*** by 1
Doc 7	Nonrelevant	Retrieved	Increase ***FP*** by 1
Doc 8	Nonrelevant	Not retrieved	Increase ***TN*** by 1

	Retrieved	*Not retrieved*	*Total*
Relevant	***2***	***1***	*3*
Nonrelevant	***2***	***3***	*5*
Total	*4*	*4*	***8***

$$Accuracy \approx 62\%$$

$$Precision = 50\% \qquad Recall = 60\%$$

PRECISION/RECALL

You can get high recall (but low precision) by retrieving all docs for all queries!

Recall is a non-decreasing function of the number of docs retrieved

In a good system, **precision decreases as either the number of docs retrieved or recall increases**

- This is not a theorem, but a result with strong empirical confirmation

DIFFICULTIES IN USING PRECISION/RECALL

Should average over large document collection/query ensembles

Need human relevance assessments

- People aren’t reliable assessors

Assessments have to be binary

- Nuanced assessments?

How to measure tradeoff?

PRECISION RECALL F-MEASURE

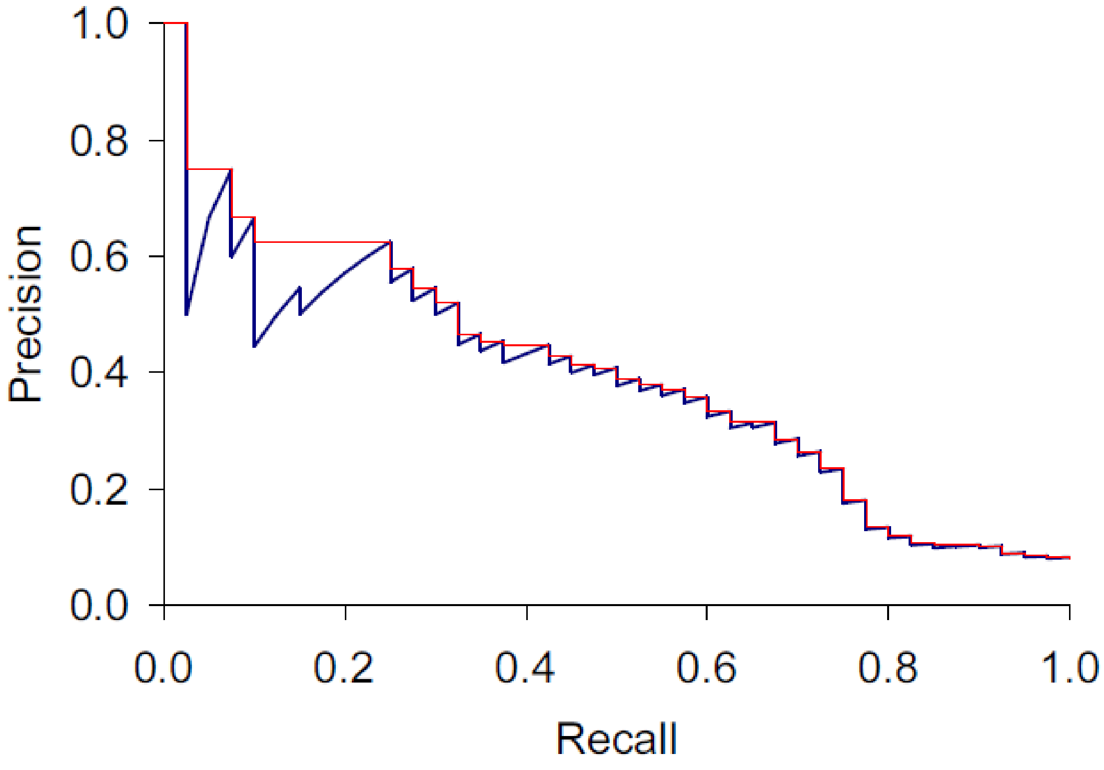
$Precision = \frac{\#\ relevant\ items\ in\ results}{Total\ \#\ items\ in\ results}$	$Recall\ (Sensitivity) = \frac{\#\ relevant\ items\ in\ results}{Total\ \#\ relevant\ items\ in\ collection}$	F_1 → harmonic mean of precision and recall (assess precision/recall tradeoff)
$Precision = \frac{TP}{TP + FP}$	$Recall = \frac{TP}{TP + FN}$	$F_1 = 2 \times \frac{Recall \times Precision}{Recall + Precision}$

Precision, recall, and F-measure are **set-based measures**

- Need extensions to work with ranked results

EVALUATING RANKED RESULTS

- The system can return any number of results
- By taking various numbers of the top returned documents (levels of recall), the evaluator can produce a **precision-recall curve**
- The interpolated precision p_{interp} at a certain recall level r is defined as the highest precision found for any recall level $r' \geq r$



PRECISION@K

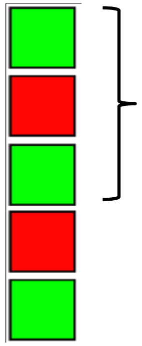
Set a rank threshold K

Compute % relevant docs in top K

Ignore documents ranked lower than K

Example:

- Precision@3 is 2/3



$K = 3$

PRECISION@K

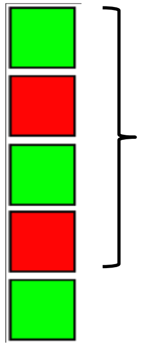
Set a rank threshold K

Compute % relevant docs in top K

Ignore documents ranked lower than K

Example:

- Precision@3 is 2/3
- Precision@4 is 2/4



$K = 4$

PRECISION@K

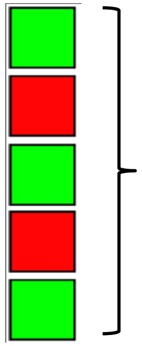
Set a rank threshold K

Compute % relevant docs in top K

Ignore documents ranked lower than K

Example:

- Precision@3 is 2/3
- Precision@4 is 2/4
- Precision@5 is 3/5



$K = 5$

- Appropriate for most of web search: all people want are good matches on the first one or two results pages
- But: averages badly and has an arbitrary parameter of k

In similar fashion we have Recall@K

BUT recall that *recall* is a function of relevant documents in collection

RECALL/PRECISION @ TOP(N)

		Precision	Recall
1	Relevant		
2	Not Relevant		
3	Not Relevant		
4	Relevant		
5	Relevant		
6	Not Relevant		
7	Relevant		
8	Not Relevant		
9	Not Relevant		
10	Not Relevant		

Assume 10 relevant docs in collection

Can you compute recall and precision?

RECALL/PRECISION @ TOP(N)

		Precision	Recall
1	Relevant	1	0.1
2	Not Relevant	0.5	0.1
3	Not Relevant	0.33	0.1
4	Relevant	0.5	0.2
5	Relevant	0.6	0.3
6	Not Relevant	0.5	0.3
7	Relevant	0.57	0.4
8	Not Relevant	0.5	0.4
9	Not Relevant	0.44	0.4
10	Not Relevant	0.4	0.4

Assume 10 relevant docs in collection

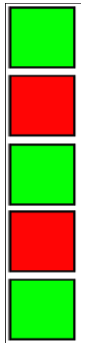
AVERAGE PRECISION (AP)

Consider rank position of each ***relevant*** doc

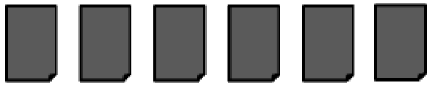
- $K_1, K_2, \ldots K_r$

Compute Precision@K for each $K_1, K_2, \ldots K_r$

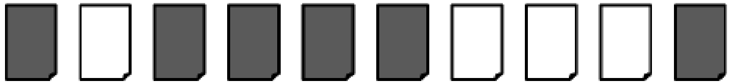
Average precision = average of $P@K$

Example:  has Average Precision of $\frac{1}{3} \times \left(\frac{1}{1} + \frac{2}{3} + \frac{3}{5}\right) \approx 0.76$

AVERAGE PRECISION (AP)

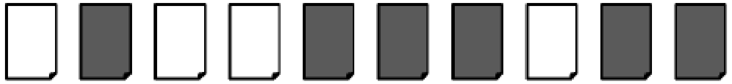
 = the relevant documents

Ranking #1



Recall	0.17	0.17	0.33	0.5	0.67	0.83	0.83	0.83	0.83	1.0
Precision	1.0	0.5	0.67	0.75	0.8	0.83	0.71	0.63	0.56	0.6

Ranking #2

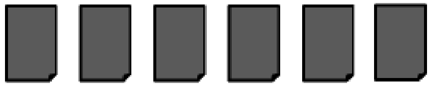


Recall	0.0	0.17	0.17	0.17	0.33	0.5	0.67	0.67	0.83	1.0
Precision	0.0	0.5	0.33	0.25	0.4	0.5	0.57	0.5	0.56	0.6

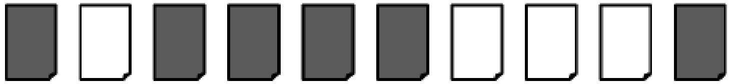
$$\text{Ranking \#1: } (1.0 + 0.67 + 0.75 + 0.8 + 0.83 + 0.6)/6 = 0.78$$

$$\text{Ranking \#2: } (0.5 + 0.4 + 0.5 + 0.57 + 0.56 + 0.6)/6 = 0.52$$

MEAN AVERAGE PRECISION (MAP)

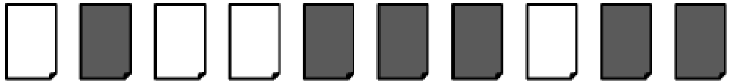
 = the relevant documents

Ranking #1



Recall	0.17	0.17	0.33	0.5	0.67	0.83	0.83	0.83	0.83	1.0
Precision	1.0	0.5	0.67	0.75	0.8	0.83	0.71	0.63	0.56	0.6

Ranking #2



Recall	0.0	0.17	0.17	0.17	0.33	0.5	0.67	0.67	0.83	1.0
Precision	0.0	0.5	0.33	0.25	0.4	0.5	0.57	0.5	0.56	0.6

MAP is Average Precision across multiple queries/rankings

For queries q_j and ranked results R_{jk} at relevant documents d_k:

$$MAP(Q) = \frac{1}{|Q|}\sum_{j=1}^{|Q|}\frac{1}{m_j}\sum_{k=1}^{m_j} Precision(R_{jk})$$

$$MAP(Q) = \frac{0.78 + 0.52}{2} = 0.65$$

$$\text{Ranking \#1: } (1.0 + 0.67 + 0.75 + 0.8 + 0.83 + 0.6)/6 = 0.78$$

$$\text{Ranking \#2: } (0.5 + 0.4 + 0.5 + 0.57 + 0.56 + 0.6)/6 = 0.52$$

MEAN AVERAGE PRECISION (MAP)

If a relevant document never gets retrieved, we assume the precision corresponding to that relevant doc to be zero

MAP is a macro-averaging metric: **each query counts equally**

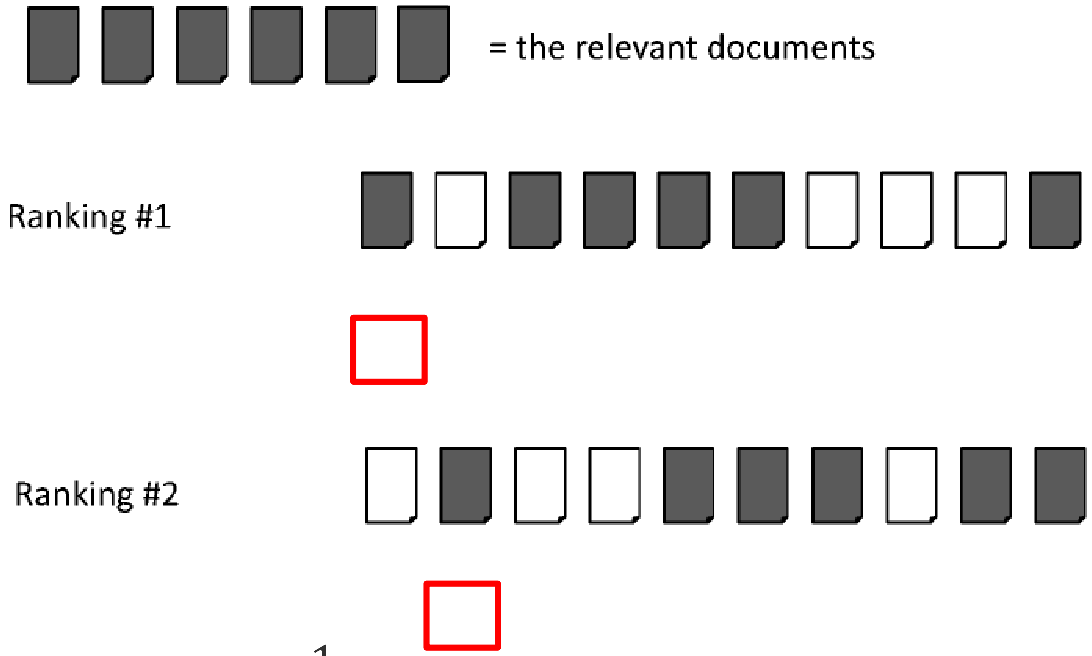
Now perhaps most commonly used measure in research papers

Good for web search?

MAP assumes user is interested in finding many relevant documents for each query

MAP requires many relevance judgments in text collection

MEAN RECIPROCAL RANK (MRR)



$$Ranking\ \#1 \ = \frac{1}{1} = 1$$

$$Ranking\ \#2 \ = \frac{1}{2} = 0.5$$

MRR is Average of reciprocal ranks across multiple queries/rankings

For queries q_j and highest ranked result R_j at relevant documents d_k:

$$MRR(Q) = \frac{1}{|Q|}\sum_{j=1}^{|Q|}\frac{1}{R_j}$$

$$MRR(Q) \ = \frac{1 + 0.5}{2} = 0.75$$

MEAN RECIPROCAL RANK (MRR)

MRR is associated with a user model where the user only wishes to see one relevant document

Assuming that the user will look down the ranking until a relevant document is found, and that document is at rank n, then the precision of the set they view is $\frac{1}{n}$

MRR is equivalent to MAP when each query has precisely one relevant document

MRR is an appropriate measure for known item search (navigational search), where the user is trying to find a document that they either have seen before or know to exist

CREATING TEST COLLECTIONS FOR IR EVALUATION

TEST COLLECTIONS

TABLE 4.3 Common Test Corpora

Collection	*NDocs*	*NQrys*	*Size (MB)*	*Term/Doc*	*Q-D RelAss*
ADI	82	35			
AIT	2109	14	2	400	>10,000
CACM	3204	64	2	24.5	
CISI	1460	112	2	46.5	
Cranfield	1400	225	2	53.1	
LISA	5872	35	3		
Medline	1033	30	1		
NPL	11,429	93	3		
OSHMED	34,8566	106	400	250	16,140
Reuters	21,578	672	28	131	
TREC	740,000	200	2000	89-3543	» 100,000

TEST COLLECTIONS

Still need
- Test queries
- Relevance assessments

➢ **Test queries**
- Must be germane to docs available
- Best designed by domain experts
- Random query terms generally not a good idea

➢ **Relevance assessments**
- Human judges, time-consuming
- Are human panels perfect?

KAPPA MEASURE FOR INTER-JUDGE (DIS)AGREEMENT

Kappa measure

- Agreement measure among judges
- Designed for **categorical judgments**
- **Corrects for chance agreement**

$$Kappa\ Statistic\ = \frac{P(A) - P(E)}{1 - P(E)}$$

$P(A)$ – proportion of time judges agree

$P(E)$ – Probability the two judges agreed by chance

$Kappa\ statistic\ =$ 0 for chance agreement, 1 for total agreement

Number of docs	Judge 1	Judge 2
300	Relevant	Relevant
70	Nonrelevant	Nonrelevant
20	Relevant	Nonrelevant
10	Nonrelevant	Relevant

KAPPA MEASURE FOR INTER-JUDGE (DIS)AGREEMENT

$$P(A) = \frac{370}{400} = 0.925$$

$$P(nonrelevant) = \frac{80}{400} \times \frac{90}{400} = 0.045$$

$$P(relevant) = \frac{320}{400} \times \frac{310}{400} = 0.62$$

$$P(E) = 0.045 + 0.62 = 0.665$$

$$Kappa = \frac{0.925 - 0.665}{1 - 0.665} = 0.776$$

		Judge 2		
Judge 1		*Relevant*	*Nonrelevant*	*Total*
	Relevant	**300**	**20**	320
	Nonrelevant	**10**	**70**	80
	Total	310	90	**400**

Kappa > 0.8 = good agreement

$0.67 < Kappa < 0.8$ → "tentative conclusions"

For >2 judges: average pairwise $kappas$

CRITIQUE OF PURE RELEVANCE

Relevance vs Marginal Relevance

- A document can be redundant even if it is highly relevant
- Duplicates
- The same information from different sources
- Marginal relevance is a better measure of utility for the user

Marginal Relevance maximizes both **relevance** and **novelty**

- evaluates relevance → similarity score between the query and the “item under consideration” to be added
- evaluates novelty → similarity score between the “item under consideration” and “items had so far”
- weighted combination of both the scores to get final results

MISCELLANEOUS IR TOPICS

Bigger Data Collection beats Clever Systems

Query Expansion

Crawlers

Estimating index size

NEXT TIME

Part-of-Speech Tagging

REFERENCES

This lecture is heavily relying on the following courses:

- CS 276 / LING 286: Information Retrieval and Web Search, Stanford University
- Natural Language Processing Lecture Slides from the Stanford Coursera course by Dan Jurafsky and Christopher Manning
- Information Retrieval slides by Simone Teufel, Cambridge University