I00041

Information Retrieval

Lecture 4: Vector Space Model

Credits to Chris Manning for Stanford lecture slides

Taxonomy of IR models

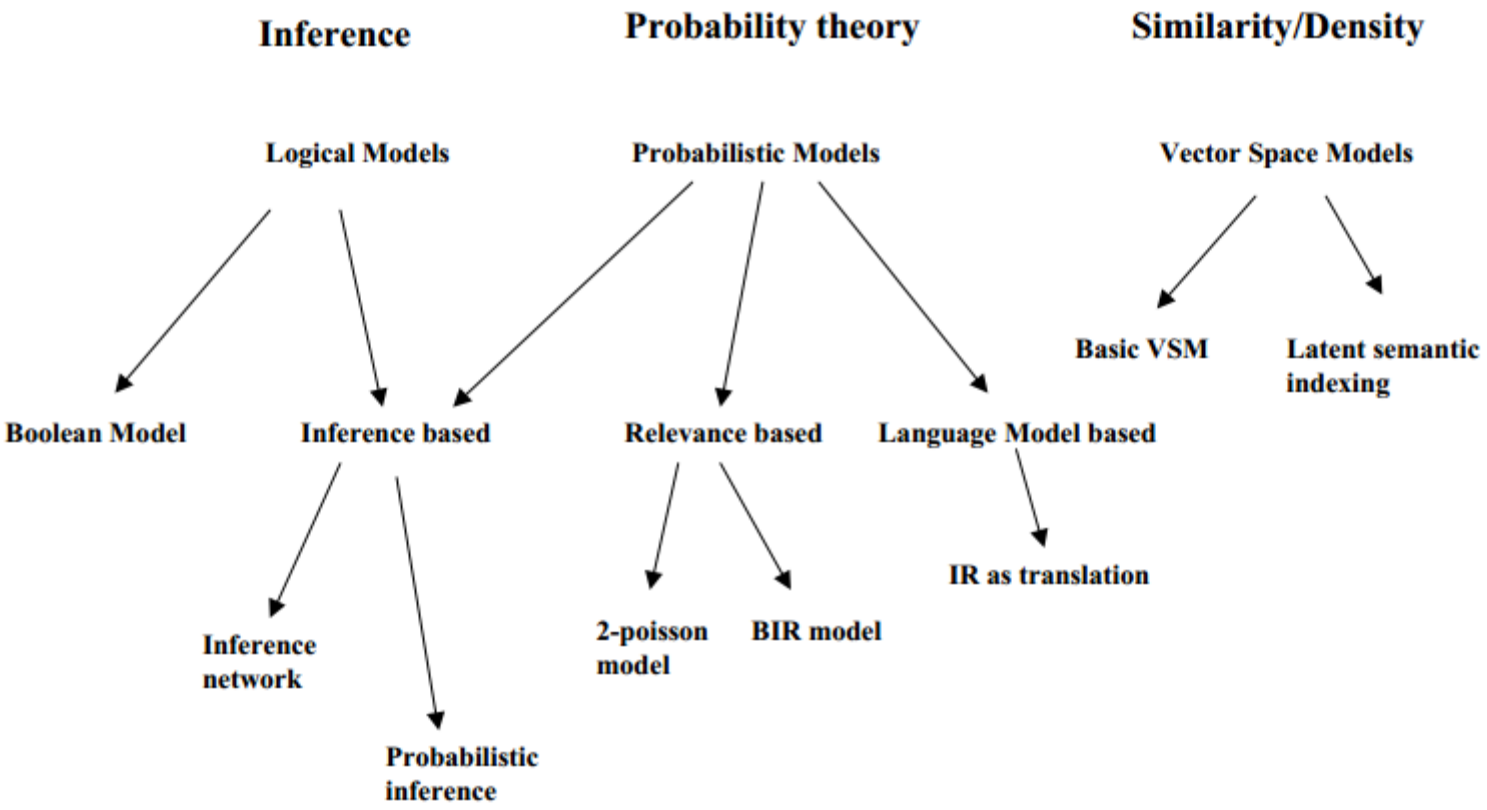


Figure 2.3. Taxonomy of IR models

Ranked retrieval

- Rather than
  - a set of documents satisfying a query expression,

  in ranked retrieval,
  - the system returns an ordering of the (top) documents in the collection for a query
- Free text queries: Rather than
  - a query language of operators and expressions,
  - the user's query is just one or more words in a human language
- Ranked retrieval is normally been associated with free text queries and vice versa

Ranked retrieval

- When a system produces a ranked result set, large result sets are not an issue
  - We just show the top k (≈ 10) results
  - We don't overwhelm the user

- Premise: the ranking algorithm works➔ score is correlated with relevance

- *Also called relevance ranking*

Formalization of Ranked Retrieval

- For each query Q_i and document D_j, compute a score RSV(Q_i,D_j)
  - RSV: **Retrieval Status Value**
  - RSV is theory neutral (could be distance based, probability based etc.)
  - What are desirable properties of RSV(Q,D)?
- At least ordinal scale (should support >, <,=)
- Value should increase with relevance relation between the query Q_i and document D_j
- Rank documents by RSV
- Any suggestions?

#1 attempt:
Set-based approach

Query and Document are each a set of terms

Jaccard coefficient

- A commonly used measure of overlap of two sets *A* and *B*

  jaccard *(A,B)* = $|A \cap B| / |A \cup B|$ $(|A \cup B| \neq 0)$

  jaccard (0,0) = 1

  - jaccard *(A,A)* = 1
  - jaccard *(A,B)* = 0 if $A \cap B = 0$
  - jaccard (*A,B*) = jaccard (*B,A*)

Commonality

length normalization

- *A* and *B* don't have to be the same size.
- Always assigns a number between 0 and 1.
- RSV(Q,D)=jaccard(terms(Q),terms(D))

- *15 March, day of the assasination of Julius Caesar.
- Ides: mid month (full moon)
- Movie

Jaccard coefficient: Scoring example

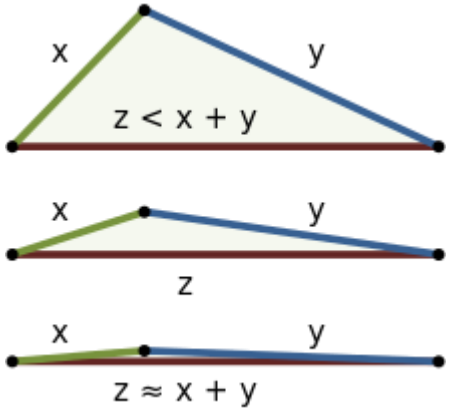
- What is the query-document match score that the Jaccard coefficient computes for each of the two documents below?
- Query: Ides of March* →{ide, march}
- Document 1: Caesar died in March → {caesar, dy, march}
- Document 2: the long march → {long, march}

→jaccard ({ide, march}, {caesar, dy, march}) = 1 / 4

→jaccard ({ide, march}, {long, march}) = 1 / 3

- Bonus for overlapping terms, penalty for length differences. Do we want this?

Characteristics of Jaccard



- Presence only. Why is this an advantage?
- Assumes binary term document matrix.
- Jaccard distance:
  - J distance = 1- J sim
  - Satisfies triangle equality, so can be used as a metric

- Related similarity coefficient:
  - Dice coefficient: $Dice(A,B) = 2x|A \cap B| / (|A| + |B|) = 2J/(J+1)$
  - What is the range of this coefficient?
  - What is difference with Jaccard coefficient?

Counterexample for Distance metric Dice

	Caesar	Brutus
D1	1	
D2	1	1
D3		1

	Intersection	Union	sumlengths	Jaccard	Dice	1-J	1-D
X=D1,D2	1	2	3	0,5	0,666667	0,5	0,333333
Y=D2,D3	1	2	3	0,5	0,666667	0,5	0,333333
Z=D1,D3	0	2	2	0	0	1	1

Y+Z-X	1	1
X+Z-Y	1	1
X+Y-Z	0	-0,33333

Refresher what is a ‘metric’

- In mathematics a metric *d* is defined as a function on a set (e.g. weights, vectors)
  - If $d(x,y) = 0 \Rightarrow x=y$ (identity)
  - $d(x,y) = d(y,x)$ (symmetry)
  - $d(x,z) =< d(x,y) + d(y,z)$ (triangle inequality)
  - $d(z,y) => 0$ (non negativity)

Refresher scales

Types of data on the basis of measurement

Scale	True Zero	Equal Intervals	Order	Category	Example
Nominal	No	No	No	Yes	Marital Status, Sex, Gender, Ethnicity
Ordinal	No	No	Yes	Yes	Student Letter Grade, NFL Team Rankings
Interval	No	Yes	Yes	Yes	Temperature in Fahrenheit, SAT Scores, IQ, Year
Ratio	Yes	Yes	Yes	Yes	Age, Height, Weight

- => A metric is expressed in ratio scale

Consequences for Mean reciprocal rank

- Evaluation of ‘known item retrieval’

- Rank Position of the ‘known item’
- Reciprocal value (nice property 0-1)
- Compute the mean across a series of queries

- Not interval scale (only ordinal)
- So we cannot compute the mean!
- So for IR ranking evaluation, we need measures on at least interval scale.
- For RSV: at least ordinal but normalized needs ratio

Jaccard in practice (1)

- Q: Ides of March
- D1: **George Clooney month: The Ides of March**

  The Ides of March is a 2011 American political drama film directed by George Clooney from a screenplay written by Clooney, along with Grant Heslov and Beau Willimon. The film is an adaptation of Willimon's 2008 play Farragut North.

  Next week: The Descendants
- D2: **George Clooney month: Up in the Air**

  Up in the Air is a 2009 American comedy-drama film directed by Jason Reitman and co-written by Reitman and Sheldon Turner. It is a film adaptation of the 2001 novel of the same name, written by Walter Kirn.

  Next week: The Ides of March

- ***Compute Jaccard(Q,D1) and Jaccard(Q,D2)***
- *We need to give credits for multiple occurrences (Tf)*

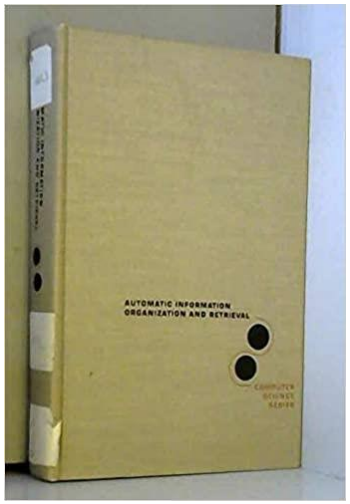
Jaccard in practice (2)

- Q: New data protection directive
- D1: New data
- D2: Data protection
- D3: Law on protection of personal data
- What about Jaccard coefficients?
- Google document frequency:
  - New: 5.5 billion
  - Data: 1.6 billion
  - Protection: 428 million
  - Directive: 22 million
- *We need to penalize highly frequent terms.*

Issues with Jaccard for scoring

- We need a more sophisticated way of normalizing for length
- It doesn't consider *term frequency* (how many times a term occurs in a document)

- (Not too) rare terms in a collection are more informative than (too) frequent terms. Jaccard doesn't consider this information



Automatic Information organization and retrieval, 1968

https://sigir.org/resources/museum/

Gerard Salton
1927-1995
Cornell University

#2 attempt
Term-weighting approach (Vector Space Model)

Measures term importance in documents

Boolean model: Binary term-document incidence matrix

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

Each document is represented by a binary vector that assigns a value from {0,1} to each term from V.

Moving to Term-document count matrices

- Consider the number of occurrences of a term in a document:
  - Each document is a count vector that assigns a count to each term from V: a **column** below

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

Documents: works of Shakespeare

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

This matrix can be stored as an inverted file, with posting lists

Bag of words model

- Vector representation doesn't consider the ordering of words in a document
  - *John is quicker than Mary*
    and
    Mary is quicker than John
    have the same vectors

- This is called the bag of words model.
  - Mathematics: a multiset of words
- In a sense, this is a step back: The positional index was able to distinguish these two documents.
  - We will look at "recovering" positional information later in this course.
  - For now: bag of words model

{ 73:Antony, 157:Brutus, 227:Caesar, 10:Calpurnia }

Term frequency tf

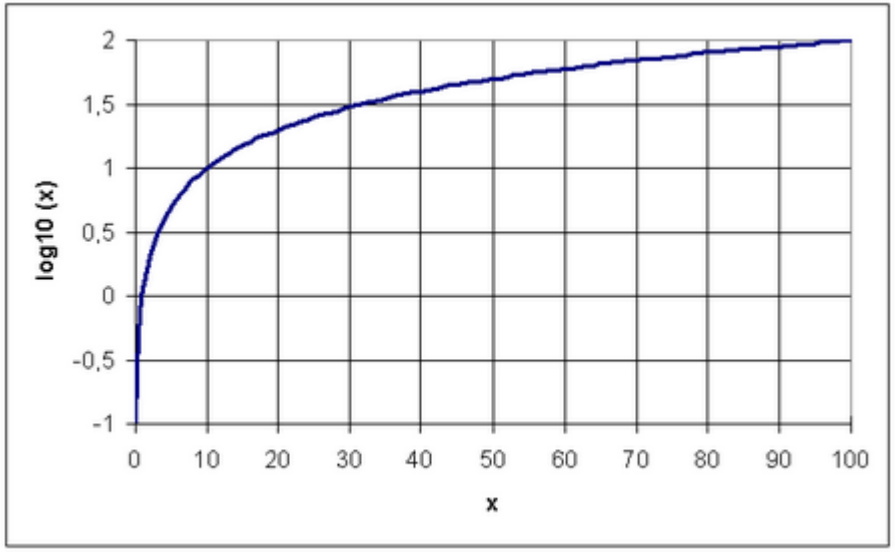
- The term frequency $tf_{t,d}$ of term *t* in document *d* is defined as the number of times that *t* occurs in *d*.

- We want to use tf when computing query-document match scores. But how?
- Raw term frequency is not what we want:
  - A document with 10 occurrences of the term is more relevant than a document with 1 occurrence of the term.
  - But not 10 times more relevant.
- Relevance does not increase proportionally with term frequency.

NB: frequency = count in IR

Relevancy and growth

- Differences are more relevant for small values than for large values:
  - the difference between ages 3 and 4 is larger than for 83 and 84
- The larger the values, the less important the differences are.
- This behavior corresponds to the logarithm
- → logarithmic normalization
  x → log (x)



Log-frequency weighting

- The log frequency weight of term t in d is

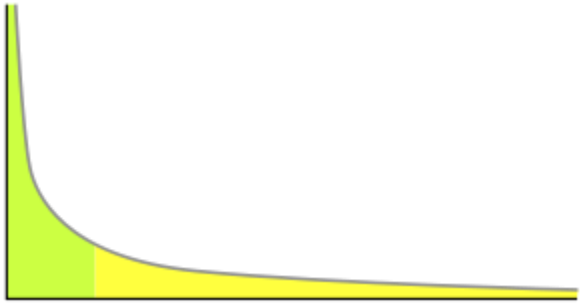
$$w_{t,d} = \begin{cases} 1 + {}^{2}\log(\mathrm{tf}_{t,d}) & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$$

- $w_{t,d}$ is interpreted as the evidence of d being relevant for query t
- Score for a document-query pair: sum the evidence

$$RSV = score(q,d) = \sum_{t \in q} w_{t,d} = \sum_{t \in q \cap d} \left(1 + \log\left(\mathrm{tf}_{t,d}\right)\right)$$

- The score is 0 if none of the query terms is present in the document.

Document frequency



- Rare terms are more informative than frequent terms
  - Recall stop words
  - Consider a term in the query that is rare in the collection (e.g., *arachnocentric*)

- A document containing this term is very likely to be relevant to the query *arachnocentric*
  - → We want a high weight for rare terms like *arachnocentric*.

Document frequency, continued

- (Not too) frequent terms are less informative than (not too) rare terms
  - Consider a query term that is frequent in the collection (e.g., *high, increase, line*)
- A document containing such a term is more likely to be relevant than a document that doesn't
  - But it's not a sure indicator of relevance.
- We will use document frequency (df) to capture this.

idf weight

- df_t is the document frequency of *t*: the number of documents that contain *t*
  - df_t is an inverse measure of the informativeness of *t* about the document identity (topical content, aboutness)
  - $df_t \leq N$
- Then df_t / *N* may be seen as the likelihood of *t*'s occurrence in any document

- We define the idf (inverse document frequency) of *t* by

$$idf_t = {}^2\log\,(N/df_t)$$

idf example, $N = 2^{25} = 33{,}554{,}432$

term	df_t	Idf_t
calpurnia	$2^0 = 1$	$^2\log(2^{25}/2^0) = {}^2\log(2^{25}) = 25$
animal	$2^5 = 32$	$^2\log(2^{25}/2^5) = {}^2\log(2^{20}) = 20$
sunday	$2^{10} = 1{,}024$	$^2\log(2^{25}/2^{10}) = {}^2\log(2^{15}) = 15$
fly	$2^{15} = 32{,}768$	$^2\log(2^{25}/2^{15}) = {}^2\log(2^{10}) = 10$
under	$2^{20} = 1{,}048{,}576$	$^2\log(2^{25}/2^{20}) = {}^2\log(2^5) = 5$
the	$2^{25} = 33{,}554{,}432$	$^2\log(2^{25}/2^{25}) = {}^2\log(2^0) = 0$

$$\text{idf}_t = {}^2\log\,(N/\text{df}_t)$$

There is one idf value for each term t in a collection.

Effect of idf on ranking

- What effect does idf have on ranking for one-term queries?
  - E.g. iPhone

- idf has no effect on ranking one term queries
  - idf affects the ranking of documents for queries with at least two terms. ***Why?***
  - For the query capricious person, idf weighting makes occurrences of capricious count for much more in the final document ranking than occurrences of person.

tf-idf weighting

- The tf-idf weight of a term is the product of its tf weight and its idf weight.

$$tf\text{-}idf\ (t,d) = \underbrace{(1+\log tf_{t,d})}_{Internal\ weight} \times \underbrace{\log\left(N / df_t\right)}_{External\ weight}$$

- Best known weighting scheme in information retrieval
  - "-" in tf-idf is a hyphen, not a minus sign! Alternative names: tf.idf, tf x idf
  - Increases with the number of occurrences within document
  - Increases with the rarity of the term in the collection

Binary → count → weight matrix

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	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	1,95

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

Final ranking of documents for a query

- The total evidence of a document *d* being relevant for a query *q* then amounts to the sum of all partial evidence

$$Score(q,d) = \sum_{t \in q \cap d} tf\text{-}idf(t,d)$$

- There are many variants
  - E.g whether query terms are also weigthted.

addressing length normalization: Documents as vectors

- So we have a |V|-dimensional vector space
- Terms are axes 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.

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
- Rank more relevant documents higher than less relevant documents

Query as a vector

- The query is also interpreted as a vector

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	q
Antony	5,25	3,18	0	0	0	0,35	0
Brutus	1,21	6,1	0	1	0	0	1
Caesar	8,59	2,54	0	1,51	0,25	0	1
Calpurnia	0	1,54	0	0	0	0	0
Cleopatra	2,85	0	0	0	0	0	0
mercy	1,51	0	1,9	0,12	5,25	0,88	0
worser	1,37	0	0,11	4,15	0,25	1,95	0

Supply

Demand

- How similar are demand and supply?

Formalizing vector space proximity

- First cut: distance between two points
  - (= distance between the end points of the two vectors)
- Euclidean distance?

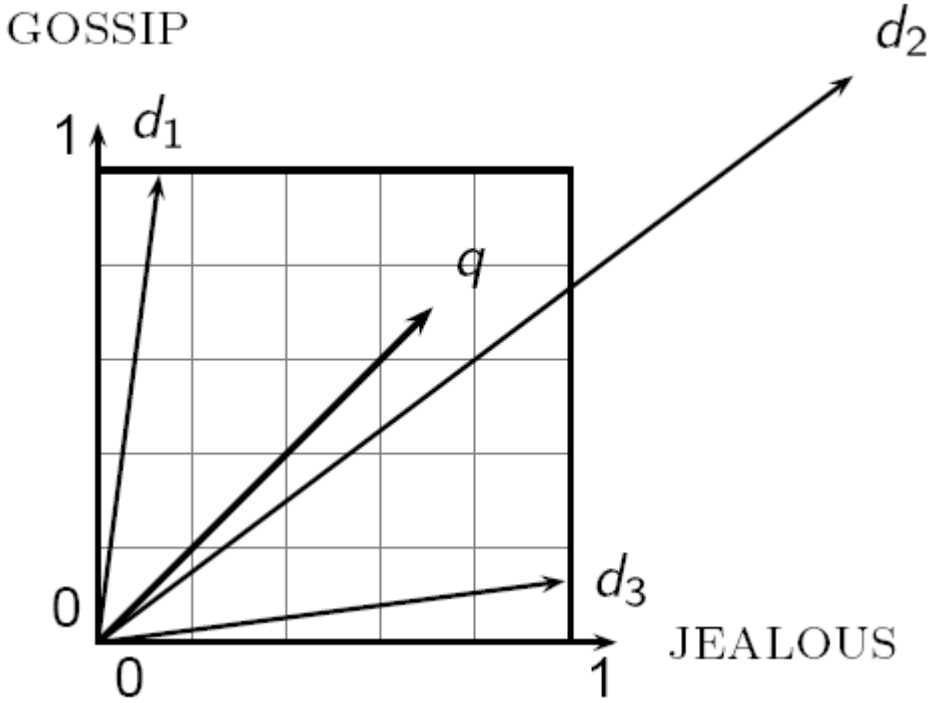
- Euclidean distance is a bad idea . . .

Why distance is a bad idea

The Euclidean distance between q and d_2 is large

even though the distribution of terms in the query q and the distribution of terms in the document d_2 are very similar.

The distance with d_1 and d_3 is similar, but these documents are very dissimilar!



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

Use angle instead of distance

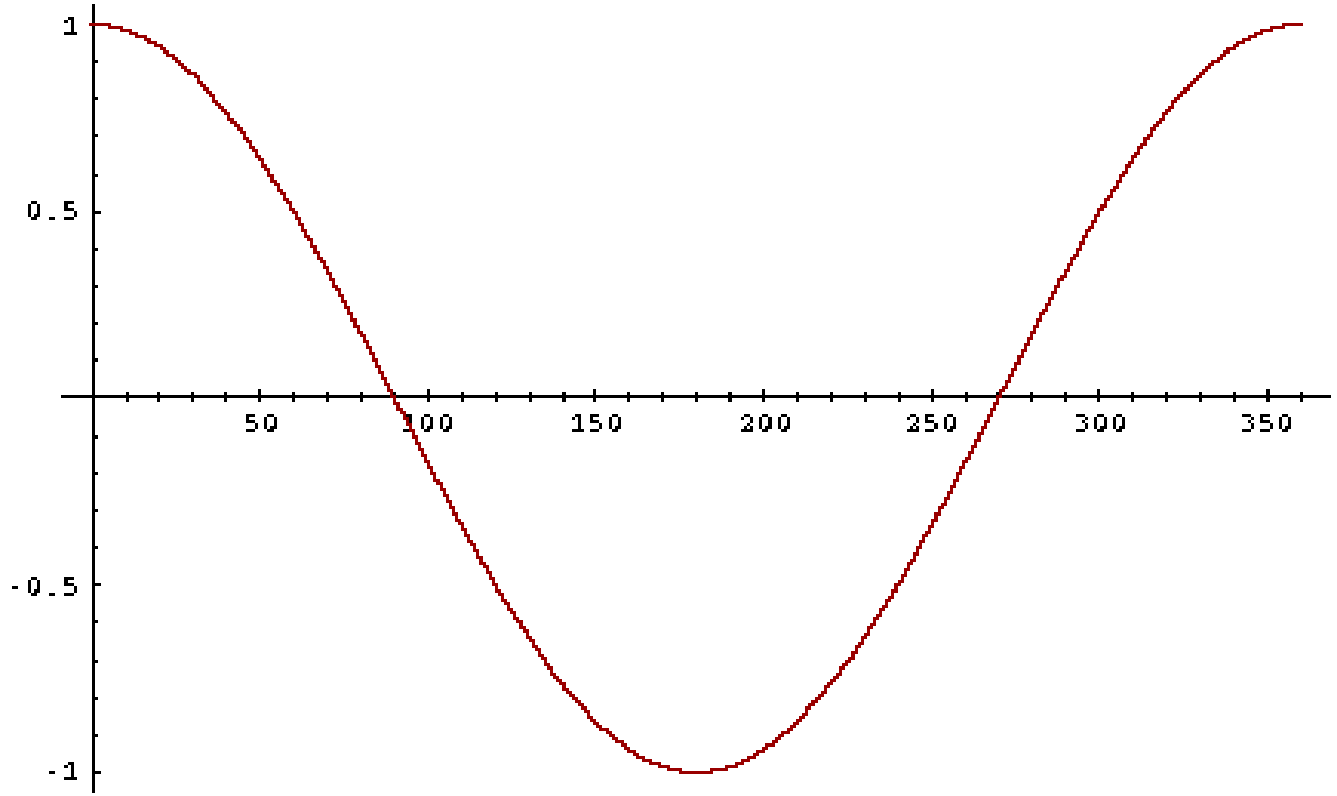
- Thought experiment: take a document *d* and append it to itself. Call this document *d*′.
- "Semantically" d and d′ have the same content
- The Euclidean distance between the two documents can be quite large
- The angle between the two documents is 0, corresponding to maximal similarity.

- 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 monotonically decreasing function for the interval [0^o, 180^o]

From angles to cosines



- But how should we be computing cosines?

Length normalization

- A vector can be (length-) normalized by dividing each of its components by its length – for this we use the L_2 norm:

$$\|\vec{x}\|_2 = \sqrt{\sum_i x_i^2}$$

- Dividing a vector by its L_2 norm makes it a unit (length) vector (on surface of unit hypersphere)
- Effect on the two documents d and d′ (d appended to itself) from earlier slide: they have identical vectors after length-normalization.
  - Long and short documents now have comparable weights

cosine(query,document)

Dot product

Unit vectors

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

q_i is the weight of term i in the query
d_i is the weight of term i in the document

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

Cosine for length-normalized vectors

- For length-normalized vectors, cosine similarity is simply the dot product (or scalar product):

$$\cos(\vec{q},\vec{d}) = \vec{q} \bullet \vec{d} = \sum_{i=1}^{|V|} q_i d_i$$

for q, d length-normalized.

Cosine similarity amongst 3 documents

How similar are the novels?

SaS: *Sense and Sensibility*

PaP: *Pride and Prejudice*

WH: *Wuthering Heights*

Note: To simplify this example
- we consider 4 terms only
- we don't do idf weighting.

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

Term frequencies (counts)

sim(SaS,PaP) ?
sim(SaS,WH) ?
sim(PaP,WH) ?

3 documents example contd.

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

SaS: *Sense and Sensibility*

PaP: *Pride and Prejudice*

WH: *Wuthering Heights*

$\text{sim(SaS,PaP)} = 0.789 \times 0.832 + 0.515 \times 0.555 + 0.335 \times 0 + 0 \times 0$

$= 0.94$

$\text{sim(SaS,WH)} = 0.79$

$\text{sim(PaP,WH)} = 0.69$

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), no normalization …

tf-idf weighting has many variants

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

- 'lnc.ntc' is a typical term weighting example for vector space systems
- Check the source code of your (python) package….

tf-idf example: **lnc.ntc**

Document: *car insurance auto insurance*
Query: *best car insurance*

Term	Query						Document				Prod
	tf-raw	tf-wt	df	idf	wt	n'lize	tf-raw	tf-wt	wt	n'lize	
auto	0		5000				1				
best	1		50000				0				
car	1		10000				1				
insurance	1		1000				2				

Doc length = $\sqrt{1^2 + 0^2 + 1^2 + 1.3^2} \approx 1.92$

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

Short and long documents

- In adapting the Okapi algorithms to deal with long documents, Stephen Robertson articulated the "**scope vs. verbosity" hypothesis:**
- *We may postulate at least two reasons why documents might vary in length. Some documents may simply cover more material than others; an extreme version would have a long document consisting of a number of unrelated short documents concatenated* ***together (the 'scope hypothesis").*** *An opposite view would have long documents like short documents but longer; in other words, a long document covers a similar scope to a short document, but simply uses more words* ***(the "verbosity hypothesis").***

Case study SIGIR paper 1996
Pivoted document length normalization

- Example of a 'data driven' experimental IR study
  - Data analysis inspires new model….
- PhD Thesis work of Amith Singhal, then at Cornell, has been "head of Google's core ranking team" (Wikipedia)

- Starting point: TREC 1-3 study showed that the SMART system underperformed compared with Okapi and Inquery systems (more about TREC in L02)

BM25

Analysis

- Doc length normalization
  - Removes score bias for long documents (Hyp2)
    - Hyp1: long = more of the same
    - Hyp2: long = covering more topics. More unique terms

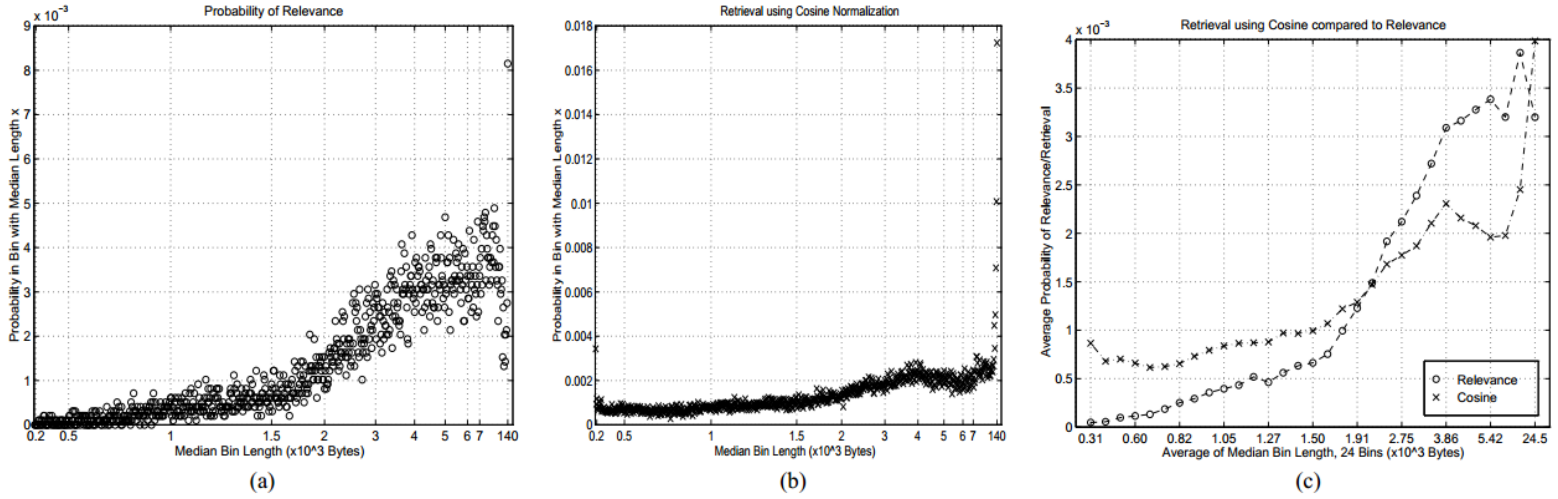
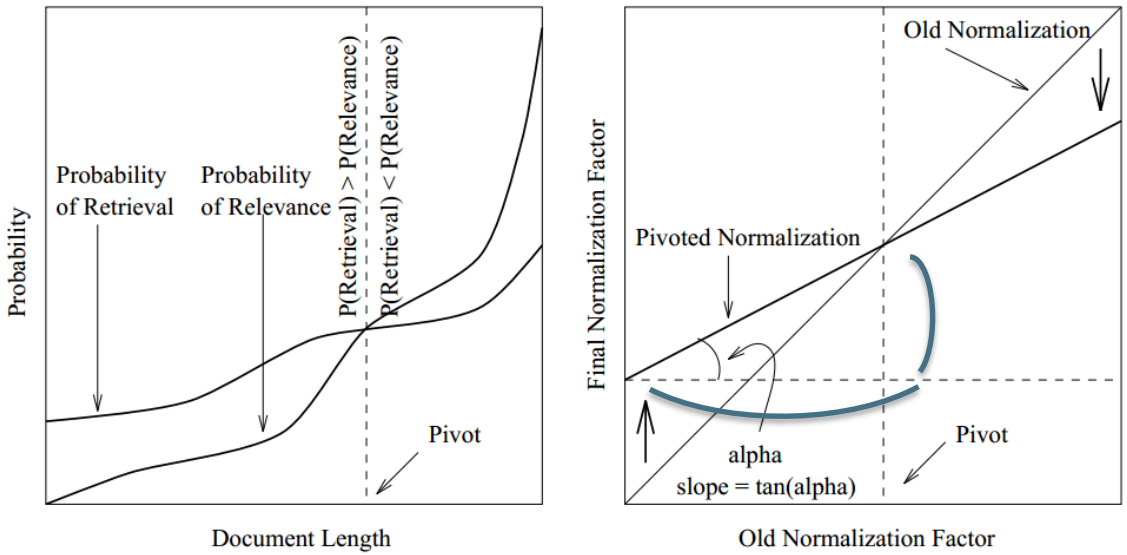


Figure 1: Probability that a relevant/retrieved document is from a bin, plotted against the median bin length. The analysis for the relevant documents is shown in (a), (b) shows the analysis for documents retrieved using cosine normalization, and (c) compares the smooth plots for (a) and (b).

Log log plots: for TREC1-3 datasets: mix of newswire and long legal docs



We need to correct by penalizing short and crediting long docs

Figure 2: **Pivoted Normalization.** The normalization factor for documents for which $P(retrieval) > P(relevance)$ is increased, whereas the normalization factor for documents for which $P(retrieval) < P(relevance)$ is decreased.

- P(ret)=RSV=sim(Q,D) x 1/ Normfactor ('u' instead of cosine length 'c')
  - Penalty means increasing normfactor, credit by decreasing normfactor

$$\frac{tf \cdot idf \ \ weight}{(1.0 - \ slope) \times pivot + \ slope \times old \ \ normalization}$$

E.g. pivot = 2000, slope = 1/3

$$\frac{tf.idf\ weight}{\frac{2}{3}2000 + \frac{1}{3}|x|}$$

For |x| = pivot, normfactor does not change

Pivoted length normalization in practice

Pivoting addresses 'scope hypothesis'

- Using |D| as length normalization factor leads to over representation of very long documents
  - Cause: $|D| \sim \sqrt{\#unique\ terms}$ (term weights are close to one)
  - Singhal et al, *Pivoted document length normalization, SIGIR 1996*
- Solution: use the number of unique terms in a document as normalization factor for Tf
  - Rationale: inproduct is a summation over the vocabulary of (unique) terms
  - Lnu.ltu

Average Tf ('verbosity hypothesis')

$$\text{RSV}(q, d_k) = \sum_{i=1}^{V_q} \frac{(1 + \log tf_{q,i}) \cdot \log \frac{N}{df_i}}{(1.0 - s)p + s \cdot V_q} \times \frac{\left(\frac{1+\log(tf_{k,i})}{1+\log(\sum_{i=1}^{V_d} tf_i/L)}\right) \cdot 1.0}{(1.0 - s)p + s \cdot V_d} \quad (14)$$

where V_q and V_d are the number of unique terms in the query and the document respectively, p and s are pivot and slope and L is the indexing vocabulary size.

Take away from this paper

- IR models make various assumptions
  - E.g. document do not differ significantly in length
- IR models are often refined after being tested on new data
- Can lead to rather complex models

- Combination of theory and data analysis

- Pragmatic approximations often made to enable run-time efficiency

Comparing IR models/systems

- Effectiveness
- Efficiency
- Explainability / Usability
- Parsimony (Occam's Razor)
  - *A simpler model, e.g. a model with fewer parameters but with same expl. power, is to be preferred*
  - *=>resource use, efficiency*

Comparing models

	Effectiveness	Efficiency	Explain/Use	Parsimony
Boolean	No ranking	++	++/-	+
Vector Space (lnc.ntc)	Ranking	++	fair	fair
Lnu.ltu	Ranking ++	++	fair	- (more parameters)

End of lecture