Data-analysis and Retrieval
Scoring and Ranking

Hans Philippi
(partially based on the slides from the Stanford course on IR)

May 1, 2024

Scoring and ranking: overview

- Ranked retrieval
- Scoring documents:
  - Today: how well do documents match our query?
  - Later: are some matching documents preferable to others?
- *Term frequency* and *inverse document frequency*
- Collection statistics and weighting
- Vector space model

Scoring and ranking

- Thus far, we discussed boolean queries
  - appropriate for expert users
  - appropriate for applications
- Inappropriate for naive users
  - most users are not familiar with logic
  - number of results may be a problem

Scoring and ranking

- Thus far, we discussed boolean queries
  - appropriate for expert users
  - appropriate for applications
- Inappropriate for naive users
  - most users are not familiar with logic
  - number of results may be a problem

Query model

- *Bag of words* model: free text query, no connectives
- Apply ranking and determine top-k
- Positional information and phrase queries can be added

Ranking: scoring function

- Query q is a (small) set of terms
- Document d is a (large) set of terms
- We need a scoring function s: $< q, d > \rightarrow v$, with $v \in [0, 1]$ or $v \in \mathbb{R}^+$
- This score expresses the quality of the match between q and d
- This score enables us to calculate a matching top-k

Scoring function: term-frequency matrix

Observation 1: frequency does matter

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello
anthony	157	73	0	0	0
brutus	4	157	0	1	0
caesar	232	227	0	2	1
calpurnia	0	10	0	0	0
cleopatra	57	0	0	0	0
mercy	2	0	3	5	5
worser	2	0	1	1	1

What would be your top-2 for the next queries:

- q_1 = anthony
- q_2 = caesar brutus

Scoring function: term frequency

- A document can be seen as a term-frequency vector
- The more frequent a query term t occurs in document d, the higher the score should be
- The term frequency $tf_{t,d}$ should contribute to the score function, but simply using the raw frequency might be overdone

Scoring function: log-frequency weighting

Weighting based on log-frequency

- $w_{t,d} = 0, \quad \text{if } tf_{t,d} = 0$
- $w_{t,d} = 1 + log(tf_{t,d}), \quad \text{if } tf_{t,d} > 0$

$0 \rightarrow 0$
$1 \rightarrow 1$
$2 \rightarrow 1.3$
$10 \rightarrow 2$
$1000 \rightarrow 4$ etc

First proposal for score function:

$$score(q, d) = \sum_{t \in q \cap d} (1 + log(tf_{t,d}))$$

Scoring function: selectivity of terms

Consider the query $q = \textit{muziek voor contrafagot}$

Scoring function: selectivity of terms

Consider the query q = *muziek voor contrafagot*

- number of results for *muziek*: 60 million
- number of results for *voor*: 860 million
- number of results for *contrafagot*: 55000

Proposal: rare terms (= low number of results) should be given a high weight in the score function

Scoring function: inverse document frequency

- We define df_t, the *document frequency* of term t, as the number of documents in our collection that contain t
- Suppose N is the total number of documents
- We define idf_t, the *inverse document frequency* of term t, as

$$idf_t = log(N/df_t)$$

- The *idf* is a measure for the rareness of a term

Inverse document frequency: example

$$idf_t = log(N/df_t)$$

$$N = 1,000,000$$

term	df	idf
calpurnia	1	?
animal	100	
sunday	1000	
fly	10,000	
under	100,000	
the	1,000,000	

Inverse document frequency: example

$$idf_t = log(N/df_t)$$

$$N = 1,000,000$$

term	df	idf
calpurnia	1	6
animal	100	4
sunday	1000	3
fly	10,000	2
under	100,000	1
the	1,000,000	0

Principle: tf.idf weighting

$$weight(t, d) = tf_{t,d} . idf_t$$

- Dot stands for multiplication
- Most common weighting scheme in information retrieval
- Combines frequency of terms in document and rarity of terms

Intermezzo: tf.idf weighting

We have four terms with their document frequencies in the Reuters RCV1 collection ($N = 806,791$) and the specific frequencies for three documents

	doc1	doc2	doc3	df_t
car	27	4	24	18,165
auto	3	33	0	6,723
insurance	0	33	29	19,241
best	14	0	17	25,235

Calculate the idf entries first

	doc1	doc2	doc3	idf_t
car				
auto				
insurance				
best				

Intermezzo: tf.idf weighting

We have four terms with their document frequencies in the Reuters RCV1 collection ($N = 806,791$) and the specific frequencies for three documents

	doc1	doc2	doc3	df_t
car	27	4	24	18,165
auto	3	33	0	6,723
insurance	0	33	29	19,241
best	14	0	17	25,235

Next, calculate the tf-idf entries

	doc1	doc2	doc3	idf_t
car				1.65
auto				2.08
insurance				1.62
best				1.50

Intermezzo: tf.idf weighting

We have four terms with their document frequencies in the Reuters RCV1 collection ($N = 806,791$) and the specific frequencies for three documents

	doc1	doc2	doc3	df_t
car	27	4	24	18,165
auto	3	33	0	6,723
insurance	0	33	29	19,241
best	14	0	17	25,235

Calculate the tf.idf entries

	doc1	doc2	doc3	idf_t
car	4.01	2.64	3.93	1.65
auto	3.07	5.24	0	2.08
insurance	0	4.08	3.99	1.62
best	3.22	0	3.35	1.50

Scoring function, first step: tf.idf weighting

$$score(q, d) = \sum_{t \in q \cap d} tf_{t,d} . idf_t$$

- What does *idf* do with single term queries?
- What is unsatisfying with respect to this scoring function?

Scoring function, first step: tf.idf weighting

$$score(q, d) = \sum_{t \in q \cap d} tf_{t,d}.idf_t$$

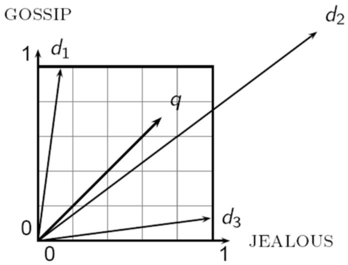
- What does *idf* do with single term queries?
- *Answer: nothing*
- What is unsatisfying with respect to this scoring function?
- *Answer: it favours long documents*
- *Approach: vector space model*

Vector space model

- Generalization of tf-idf scoring
- Each document d can be represented by a D-dimensional vector, where D is the number of all known terms
- In other words: each document d is a point in $\mathbb{R}^D$
- In an analogous way: each query q is a point in $\mathbb{R}^D$
- These vectors are generally sparse
- D is very large, several tens of millions for the web
- We have a notion of proximity for vectors from linear algebra: Euclidian distance ...
- .. but we will arrive at a more sophisticated approach to determine the similarity between a query q and a document d

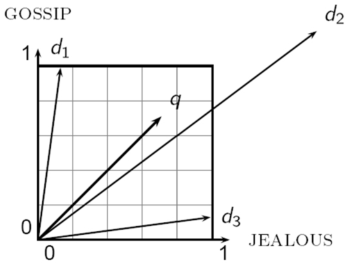
Similarity of vectors: Euclidian distance?

For the moment, $D = 2$.



Which document vector is most similar to query vector q?

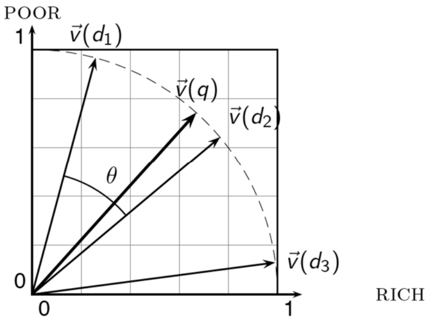
Similarity of vectors: angle



Which document vector has the smallest angle with query vector q?

Similarity of vectors: angle

Let us consider normalized vectors (length $= 1$):



Which document vector is the most similar to query vector q?

Similarity of vectors: inner product

Suppose we have two vectors:[1]

$$x^T = < x_1, x_2, \ldots, x_D >$$
$$y^T = < y_1, y_2, \ldots, y_D >$$

The inner product of two vectors with dimension D is (algebraically):

$$x \bullet y = \sum_{i=1}^{D} x_i . y_i$$

The inner product of two vectors is (geometrically):

$$x \bullet y = \parallel x \parallel . \parallel y \parallel . \; cos(\theta)$$

where θ is the angle between x and y

[1] T stands for *transposed*: the vector is written as a row

Similarity of vectors: inner product

The Euclidean length (L_2-norm) of a vector x is

$$\| x \| = \| x \|_2 = \sqrt{\sum_{i=1}^{D} x_i^2}$$

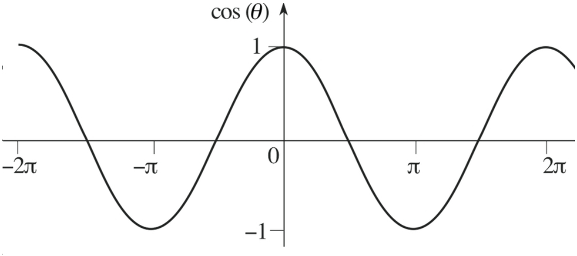
We can rewrite the geometric definition of the inner product:

$$cos(\theta) = \frac{x \bullet y}{\| x \| . \| y \|}$$

If x and y are normalized ($\| x \| = \| y \| = 1$), then

$$cos(\theta) = x \bullet y$$

Similarity of vectors: cosine



- Note that only the interval $[0, \pi/2]$ is relevant for our purposes, because all vectors have non-negative components
- In our context, an inner product can only be zero if the two vectors have a mismatch on all the terms

Intermezzo: inner product

Suppose we have the following vectors:

$x^T = < 1, 2, 2 >$
$y^T = < 0, 4, 3 >$

- Calculate the inner product of x and y
- Calculate the lenghts of x and y and normalize them
- Calculate the cosine of the angle between x and y

Intermezzo: inner product

Suppose we have the following vectors:

$x^T = <1, 2, 2>$
$y^T = <0, 4, 3>$

- $x \bullet y = 14$
- $\| x \| = 3$
- $\| y \| = 5$
- Normalized: $(x')^T = <1/3, 2/3, 2/3>$
- Normalized: $(y')^T = <0, 4/5, 3/5>$
- $cos(\theta) = x' \bullet y' = 14/15 = 0.9333$

Vector space model: variations in weighting schemes

Term frequency		Document frequency	
n (natural)	$\mathrm{tf}_{t,d}$	n (no)	1
l (logarithm)	$1 + \log(\mathrm{tf}_{t,d})$	t (idf)	$\log \frac{N}{\mathrm{df}_t}$
a (augmented)	$0.5 + \frac{0.5 \times \mathrm{tf}_{t,d}}{\max_t(\mathrm{tf}_{t,d})}$	p (prob idf)	$\max\{0, \log \frac{N-\mathrm{df}_t}{\mathrm{df}_t}\}$
b (boolean)	$\begin{cases} 1 & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$		
L (log ave)	$\frac{1+\log(\mathrm{tf}_{t,d})}{1+\log(\mathrm{ave}_{t \in d}(\mathrm{tf}_{t,d}))}$		

Normalization	
n (none)	1
c (cosine)	$\frac{1}{\sqrt{w_1^2+w_2^2+\ldots+w_M^2}}$
u (pivoted unique)	$1/u$ (Section 6.4.4)
b (byte size)	$1/\mathit{CharLength}^{\alpha}$, $\alpha < 1$

SMART notation: variant coding *aaa.bbb* for document and query

Standard weighting scheme: lnc.ltn
Document: logarithmic, no, cosine
Query: logarithmic, idf, none

Vector space model: example

Example with weighting scheme lnc.ltc

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	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

Exercise: what is N, the number of docs?

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

Score = 0+0+0.27+0.53 = 0.8

Ranking: other considerations

- We have seen methods to determine which documents d in our collection match query q best
- But if someone submits a query $q = extraterrestrial\ life$, we might prefer pages from SETI, NASA, Wikipedia or *.edu in our top-k ...
- .. above pages from niburu.co

Ranking: other considerations

- We have seen methods to determine which documents d in our collection match query q best
- But if someone submits a query $q = \textit{extraterrestrial life}$, we might prefer pages from SETI, NASA, Wikipedia or *.edu in our top-k ...
- .. above pages from niburu.co
- Can we model the *importance* of a site in some way?
- Yes, we can: PageRank!

References

Manning:

- chapter 6.2 - 6.4.3

"-" means: up to and including