Multimedia Information Retrieval and Technology

Lecture 7. Scoring

By : Fangyu Wu

Room: SD555

Xi’an Jiaotong-Liverpool University

西交利物浦大学

Exercise

Consider following documents with the stop word list: [when, in, the, and, I]

Doc 1: when walking in the rain

Doc 2: rain stopped walk, I ran, rain stop.

Doc 3: stop walking and run

tf_{stop}

0

2

1

$df_{stop} = 2$

$idf_{stop} = \log_{10} \frac{N}{df}$

$= \log_{10} \frac{3}{2}$

$= 0.176$

*Determine the term frequency for the term **stop**.*

*Determine the document frequency and idf for the term **stop**.*

Xi'an Jiaotong-Liverpool University
西交利物浦大学

Recap: Log-frequency weighting

The log frequency weight of term t in d is

$$w_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{tf}_{t,d}, & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0, & \text{otherwise} \end{cases}$$

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

Recap: idf weight

- df_t is an inverse measure of the informativeness of t
- $\mathrm{df}_t \leq N$

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

$$\mathrm{idf}_t = \log_{10}(N/\mathrm{df}_t)$$

- We use log (N/df_t) instead of N/df_t to "dampen" the effect of idf.

Recap: Cosine(query,document) Similarity

Check the question description to decide whether do normalization or not.

Unit vectors

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

q_i is the tf-idf weight of term i in the query;
d_i is the tf-idf weight of term i in the document.

Cosine similarity of $\vec{q}$ and $\vec{d}$ … or, equivalently, the cosine of the angle between $\vec{q}$ and $\vec{d}$.

Xi'an Jiaotong-Liverpool University 西交利物浦大学

- Computing vector scores
- Variant weighting schemes
  - Maximum *tf* normalization

Exercise

We now consider the query ***best car insurance*** on a fictitious collection with N = 1,000,000 documents where the document frequencies of auto, best, car and insurance are respectively 5000, 50000, 10000 and 1000.

What is the score (cosine similarity) for this query with a document "car insurance auto insurance"? logarithmic term weighting (wf columns) for query and raw term frequency for document, idf weighting for the query only, and length normalization for document only.

tf-idf example: 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											
best											
car											
insurance											

Key to columns:

- tf-raw: raw (unweighted) term frequency,
- tf-wt: logarithmically weighted term frequency,
- df: document frequency,
- idf: inverse document frequency,

Xi'an Jiaotong-Liverpool University 西交利物浦大学

tf-idf example: 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						1				
best	1						0				
car	1						1				
insurance	1						2				

Key to columns:
- tf-raw: raw (unweighted) term frequency,
- tf-wt: logarithmically weighted term frequency,
- df: document frequency,
- idf: inverse document frequency

Xi'an Jiaotong-Liverpool University 西交利物浦大学

tf-idf example: 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			1	1			
best	1	1	50000	1.3			0	0			
car	1	1	10000	2.0			1	1			
insurance	1	1	1000	3.0			2	1.3			

Key to columns:

- wt: the final weight of the term in the query or document,

tf-idf example: 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		1	1	1		
best	1	1	50000	1.3	1.3		0	0	0		
car	1	1	10000	2.0	2.0		1	1	1		
insurance	1	1	1000	3.0	3.0		2	1.3	1.3		

Key to columns:

- n'lized: document weights after cosine normalization,
- product: the product of final query weight and final document weight

Xi'an Jiaotong-Liverpool University 西交利物浦大学

tf-idf example: 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		
best	1	1	50000	1.3	1.3	0.34	0	0	0		
car	1	1	10000	2.0	2.0	0.52	1	1	1		
insurance	1	1	1000	3.0	3.0	0.78	2	1.3	1.3		

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

Xi'an Jiaotong-Liverpool University
西交利物浦大学

tf-idf example: 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

Score = 0+0+0.27+0.53 = 0.8

Xi'an Jiaotong-Liverpool University
西交利物浦大学

Computing vector scores

We have

1) a collection of documents each represented by a vector
2) a free text query represented by a vector
3) a positive integer *K*.

Object:

To seek the *K* documents of the collection with the highest vector space scores on the given query.

Xi'an Jiaotong-Liverpool University 西交利物浦大学

The basic algorithm for computing vector space scores?

Top K documents of Score[];

Xi'an Jiaotong-Liverpool University
西交利物浦大学

The basic algorithm for computing vector space scores

```
CosineScore($q$)
1   float $Scores[N] = 0$
2   float $Length[N]$
3   for each query term $t$

7   Read the array $Length$
8   for each $d$
9   do $Scores[d] = Scores[d]/Length[d]$
10  return Top $K$ components of $Scores[]$
```

The basic algorithm for computing vector space scores

The array **Length** holds the lengths (normalization factors) for each of the N Documents

Whereas the array **Scores** holds the scores for each document.

Step5-6: update the score of each document by adding in the contribution from term t.

Xi'an Jiaotong-Liverpool University
西交利物浦大学

The basic algorithm for computing vector space scores

```
CosineScore($q$)
float $Scores[N] = 0$
float $Length[N]$
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 $Scores[d] += w_{t,d} \times w_{t,q}$
Read the array $Length$
for each $d$
do $Scores[d] = Scores[d] / Length[d]$
return Top $K$ components of $Scores[]$
```

Xi'an Jiaotong-Liverpool University 西交利物浦大学

The basic algorithm for computing vector space scores

This process of adding in contributions one query term at a time is known as ***term-at-a-time scoring*** or ***accumulation***.

- TAAT = "Term At A Time"
  - Scores for all docs computed concurrently, one query term at a time
- DAAT = "Document At A Time"
  - Total score for each doc (incl all query terms) computed, before proceeding to the next

Xi'an Jiaotong-Liverpool University
西交利物浦大学

Summary – vector space ranking

STEPs:

1. Represent the query as a weighted (tf-idf) vector
2. Represent each document as a weighted (tf-idf) vector
3. Compute the cosine similarity score for the query vector and each document vector.

4. Rank documents with respect to the query by score.
5. Return the top *K* (e.g., *K* = 10) to the user

- Computing vector scores
- Variant weighting schemes
  - Maximum *tf* normalization

Xi'an Jiaotong-Liverpool University 西交利物浦大学

Variant weighting schemes

$$score(\vec{q},\vec{d}) = \frac{\vec{q} \bullet \vec{d}}{|\vec{q}||\vec{d}|} = \frac{\vec{q}}{|\vec{q}|} \bullet \frac{\vec{d}}{|\vec{d}|}$$

is fundamental to information retrieval systems that use any form of vector space scoring.

Variations : **specific choices of weights** in the vectors $\vec{d}$ and $\vec{q}$.

Xi'an Jiaotong-Liverpool University
西交利物浦大學

Weighting may differ in queries vs documents

A number of alternatives to tf and tf-idf have been considered.

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

Exercise

With normalized term weights vectors for three documents: Doc1=(0.897,0.125,0,0.423), Doc2=(0.076,0.786,0.613,0), Doc3=(0.595,0,0.706,0.383), rank the three documents by computed score for the query **car insurance**, for each of the following cases of term weighting in the query:

1. The weight of a term is 1 if present in the query, 0 otherwise (No normalization).
2. Euclidean normalized tf-idf.

term	df_t	idf_t
car	18,165	1.65
auto	6723	2.08
insurance	19,241	1.62
best	25,235	1.5

Xi'an Jiaotong-Liverpool University
西交利物浦大学

Solution

1)

	query	Doc1	Doc2	Doc3
car	1	0.897	0.076	0.595
qauto	0	0.125	0.786	0
insurance	1	0	0.613	0.706
best	0	0.423	0	0.383

Score(q,doc1)=1*0.897+1*0=0.897

Score(q,doc2)=1*0.076+1*0.613=0.689

Score(q,doc3)=1*0.595+1*0.706=1.301

Rank: doc3,doc1,doc2

Solution

2)

	Query				Doc1	Doc2	Doc3
	tf	idf	w_{tq}	Nor’			
car	1	1.65	1.65	0.714	0.897	0.076	0.595
auto	0	2.08	0	0	0.125	0.786	0
insurance	1	1.62	1.62	0.701	0	0.613	0.706
best	0	1.5	0	0	0.423	0	0.383

Length=2.31

Score(q,doc1)=0.714*0.897=0.64

Score(q,doc2)=0.714*0.076+0.701*0.613=0.4839

Score(q,doc3)=0.714*0.595+0.701*0.706=0.9197

Rank: doc3,doc1,doc2

- Computing vector scores
- Variant weighting schemes
  - Maximum *tf* normalization

Maximum *tf* normalization

Normalize the *tf* weights of all terms occurring in a document by the maximum *tf* in that document.

For each document d,

$$tf_{max}(d) = max_{t \in d} tf_{t,d}$$

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

Xi'an Jiaotong-Liverpool University
西交利物浦大学

Maximum *tf* normalization

A normalized term frequency for term t in document d is computed by:

$$ntf_{t,d} = a + (1 - a)\frac{tf_{t,d}}{tf_{max}(d)}$$

Where a is a value between 0 and 1, and is generally set to 0.4;

“a” is a smoothing term whose role is to damp the contribution of the second part, which act as a scaling down of *tf* by the largest *tf* value in *d*

Xi’an Jiaotong-Liverpool University
西交利物浦大學

Exercise

$$ntf_{t,d} = a + (1-a)\frac{tf_{t,d}}{tf_{max}(d)}$$

Consider following documents with the stop word list: [when, in, the, and, I]

Doc 1: when walking in the rain max=1

Doc 2: rain stopped walk, I ran, rain stop. max=2

Doc 3: stop walking and run, run, run max=3

Maximum tf weighting (a=0.3) for the term ***stop****.*

$$\text{Doc1: } 0.3 + 0.7 \cdot \frac{0}{1} = 0.3$$

$$\text{Doc2: } 0.3 + 0.7 \cdot \frac{2}{2} = 1$$

$$\text{Doc3: } 0.3 + 0.7 \cdot \frac{1}{3} \approx 0.53$$

Xi'an Jiaotong-Liverpool University 西交利物浦大学

Doc 1: max=1

Doc 2: max=2

Doc 3: max=3

Maximum tf weighting (a=0.3)for the term stop.

$$ntf_{t,d} = a + (1 - a)\frac{tf_{t,d}}{tf_{max}(d)}$$

Doc 1: 0.3+0.7*0=0.3

Doc 2: 0.3+0.7*2/2=1

Doc 3: 0.3+0.7*1/3=0.53

Xi'an Jiaotong-Liverpool University 西交利物浦大学

Exercise

We now consider the query ***best car insurance*** on a fictitious collection with N = 1,000,000 documents where the document frequencies of auto, best, car and insurance are respectively 5000, 50000, 10000 and 1000.

What is the score (cosine similarity) for this query with a document "car insurance auto insurance"? logarithmic term weighting (tf-wt columns) for query and **Maximum *tf* weighting （a=0.3）** for document, idf weighting for the query only, and length normalization for document only.

Xi'an Jiaotong-Liverpool University
西交利物浦大学

example:

Document: car insurance auto insurance
Query: best car insurance

Term	Query					Document			Prod
	tf-raw	tf-wt	df	idf	wt	tf-raw	tf-wt	n’lize	
auto									
best									
car									
insurance									

Key to columns:

- tf-raw: raw (unweighted) term frequency,
- tf-wt: weighted term frequency,
- df: document frequency,
- idf: inverse document frequency,

Xi’an Jiaotong-Liverpool University 西交利物浦大学

example:

Document: car insurance auto insurance
Query: best car insurance

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

Key to columns:

- tf-raw: raw (unweighted) term frequency,
- tf-wt: weighted term frequency,
- df: document frequency,
- idf: inverse document frequency,

Xi'an Jiaotong-Liverpool University
西交利物浦大学

example:

Document: car insurance auto insurance
Query: best car insurance

Term	Query					Document			Prod
	tf-raw	tf-wt	df	idf	wt	tf-raw	tf-wt	n'lize	
auto	0	0	5000	2.3	0	1	0.65		
best	1	1	50000	1.3	1.3	0	0.3		
car	1	1	10000	2.0	2.0	1	0.65		
insurance	1	1	1000	3.0	3.0	2	1		

Xi'an Jiaotong-Liverpool University
西交利物浦大学

tf-idf weighting has many variants

The mnemonic for representing a combination of weights takes the form ddd.qqq (It is quite common to apply different normalization functions to the document vector and query vector.)

where the first triplet gives the term weighting of the document vector, while the second triplet gives the weighting in the query vector.

tf-idf weighting has many variants

The mnemonic for representing a combination of weights takes the form ddd.qqq (It is quite common to apply different normalization functions to the document vector and query vector.)

in each triplet: specifies

1) the term frequency component of the weighting,
2) the document frequency component,
3) the form of normalization used.

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)	$1/\mathit{CharLength}^{\alpha}$, $\alpha < 1$
L (log ave)	$\frac{1 + \log(\mathrm{tf}_{t,d})}{1 + \log(\mathrm{ave}_{t \in d}(\mathrm{tf}_{t,d}))}$				

SMART notation for tf-idf variants.

Xi'an Jiaotong-Liverpool University
西交利物浦大学

Weighting may differ in queries vs documents

(for both effectiveness and efficiency reasons

A very standard weighting scheme is: lnc.ltc

Document: logarithmic tf (l as the 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

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

Score = 0+0+0.27+0.53 = 0.8

Xi’an Jiaotong-Liverpool University
西交利物浦大学