Calculating TF-IDF Cosine Scores

tf-idf weighting has may variants

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

- 빨간색의 수식이 가장 잘 사용되는 방법
- 다른 방법들은 어떻게 normalizing을 할 것이냐에 따라 조금씩 다름
  - a(augmented) $\rightarrow$ 0.5~1의 값이 나오도록
  - p(prob idf) $\rightarrow$ term이 전체 문서에서 반 이상나온다면 가중치 0을 주어 불용어 취급

Weighting may differ in queries VS. documents

- Many search engines allow for different weightings for queries VS. documents
- SMART Notation : denotes the combination in user in an engine, with the notation ddd.qqq, using the acronyms from the previous table
- A very standard weighting scheme is : lnc(document).ltc(query) or **lnc.ltn**
- Document : logarithmic tf, no idf and cosine normalization
  - no idf를 사용하는 이유 : you've already put in an IDF factor for the same words in the query.
- query : logarithmic tf, idf, cosine normalization

tf-idf exmaple : 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?

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

Score = 0+0+0.27+0.53 = 0.8

- prod = query's n'lize * document's n'lize

Computing Cosine Scores

Computing cosine scores

```
CosineScore(q)
 1  float Scores[N] = 0
 2  float Length[N]
 3  for each query term t
 4  do calculate w_{t,q} and fetch postings list for t
 5      for each pair(d, tf_{t,d}) in postings list
 6      do Scores[d]+ = w_{t,d} × w_{t,q}
 7  Read the array Length
 8  for each d
 9  do Scores[d] = Scores[d]/Length[d]
10  return Top K components of Scores[]
```

- **length normalization of the query is actually unnecessary** because the query vector has some length and for whatever it is, the effective length normalization would just be a rescaling that applies to all query document calculation and wouldn't change the final result.
- 효율적인 알고리즘은 아니지만 가장 general idea 알고리즘

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 to the user

출처

- 스탠포드 IR 강의 (https://www.youtube.com/watch?v=k1tD7pYKWuM&list=PLaZQkZp6WhWwoDuD6pQCmgVyDbUWl_ZUi&index=13)