vector space retrieval model의 예

(1)루트계산기: http://mwultong.blogspot.com/2007/12/root-calculator.html
(2)제곱값계산기: http://mwultong.blogspot.com/2008/01/pow-calc.html

간단하게, 다음과 같은 3개의 documents로 이루어진 문서집단 C가 있다고 생각해 보자:

d1: "new york times"
d2: "new york post"
d3: "los angeles times"

어떤 용어(term)는 두 개의 문서에 포함되어 있고, 또 어떤 것은 단지 한 개의 문서 속에만 있다. 문서(document)의 총 수는 3이다. 그러므로 이 용어들의 **역문헌빈도(idf: inverse document frequency)**는 다음과 같다:

term의 값 = log2(총문서의 수/term을 포함하고 있는 문서의 수)

angles	*log2(3/1)=1.584*
los	*log2(3/1)=1.584*
new	*log2(3/2)=0.584*
post	*log2(3/1)=1.584*
times	*log2(3/2)=0.584*
york	*log2(3/2)=0.584*

또 다른 공식은 다음과 같다:

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

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

위의 표를 보면 the 같은 단어는 거의 모든 문서에서 나타나므로, idf값은 0에 가깝게 나타날 것이다. 무슨 의밀까? 생각해 보자.

본론으로 돌아가서, 이제 C에 있는 각 document에 포함되어 있는 모든 용어의 **tf(term frequency: 용어 빈도) score**를 계산해 보면 아래와 같다: vectors에 있는 단어들이 알파벳 순으로 정렬되어있다고 가정한다.

	angeles	los	new	post	times	york
d1	0	0	1	0	1	1
d2	0	0	1	1	0	1
d3	1	1	0	0	1	0

위의 각 용어의 idf 값을 tf score에 곱하면, 아래와 같은 documents-by-terms matrix를 얻게 된다: (모든 용어는 C에 있는 각 문서에 단지 한번만 나타난다. 따라서 normalization용의 최대 값은 1 이다.)

	angeles	los	new	post	times	york
d1	0	0	0.584	0	0.584	0.584
d2	0	0	0.584	1.584	0	0.584
d3	1.584	1.584	0	0	0.584	0

“new new times”이란 쿼리가 주어졌을 때, 우리는 그 쿼리용의 tf-idf vector를 계산한 다음에, C에 있는 각 문서의 점수를 cosine similarity measure를 사용하여 이 쿼리와 비교하여 계산한다. 쿼리용어의 tf-idf 값을 계산할 때, 우리는 그 빈도를 최대 빈도수(2)로 나눈 다음에, idf 값을 곱하면 다음과 같다:

	angeles	los	new	post	times	york
q	0	0	(2/2)*0.584=0.584	0	(1/2)*0.584=0.292	0

각 문서와 쿼리의 길이를 계산하면 다음과 같다:

Length of d1 = sqrt(0.584^2+0.584^2+0.584^2)=1.011
Length of d2 = sqrt(0.584^2+1.584^2+0.584^2)=1.786
Length of d3 = sqrt(1.584^2+1.584^2+0.584^2)=2.316
Length of q = sqrt(0.584^2+0.292^2)=0.652

따라서 유사도 값(similarity values)은 다음과 같다:

cosSim(d1,q) = (0*0+0*0+0.584*0.584+0*0+0.584*0.292+0.584*0) / (1.011*0.652)
= 0.776

cosSim(d2,q) = (0*0+0*0+0.584*0.584+1.584*0+0*0.292+0.584*0) / (1.786*0.652)
= 0.292

cosSim(d3,q) = (1.584*0+1.584*0+0*0.584+0*0+0.584*0.292+0*0) / (2.316*0.652)
= 0.112

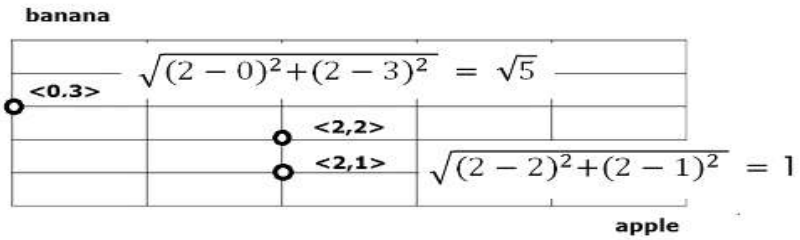
위의 유사도 값에 따라, 쿼리 결과로 나타나는 최종적인 문서의 서열은 다음과 같다:

d1,
d2,
d3.

<<중요한 설명: 길이 정규화란 ?>>

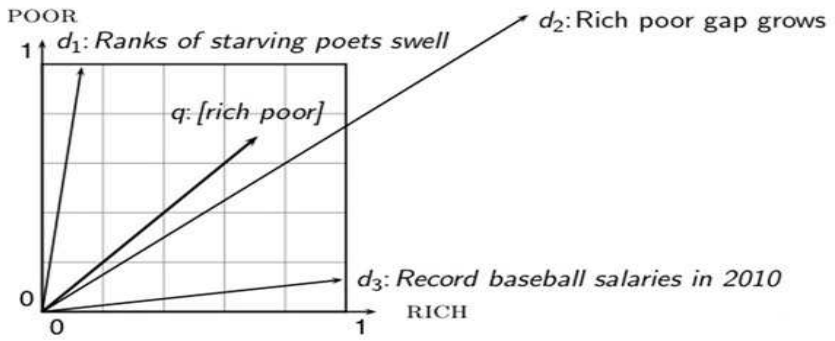
1) Euclidean distance

유클리디언 거리는, 각 문서와 질의어의 거리를 계산하는 방법이다. 계산 방법은 벡터의 내적을 구하는 것과 같다.



위 그림과 같이 유클리디언 거리를 구하면 <2,1>의 위치에 있는 문서가 더욱 유사하다는 것을 알 수 있다. (거리가 짧은 것이 더욱 유사하다는 뜻이므로)

하지만 이런 유클리디언 거리를 이용하는 방법은 그다지 좋은 방법은 아니다.
왜냐하면 문서의 방향성이 고려되지 않았기 때문이다. 만약 같은 단어의 중복으로 이루어진 문서가 있다면, 즉 질의어가 apple 4개 banana 4개로 이루어진 문서가 있다면, apple2개와 banana 2개로 이루어진 질의어와 가장 유사하지만 거리가 멀기 때문에 선택되기가 힘들다.



위 그림을 보면 질의어인 rich poor 에 대하여 가장 유사한 문서는 d2 이지만, 방향성을 무

시하고 거리만 구한다면 다른 문서가 선택될 것이다. 따라서 거리뿐만 아니라 방향성을 같이 계산할 수 있는 유사도 계산법이 필요하다.

2) Length Normalization

위의 문제점을 해결하기 위한 첫번째 방법이 길이 정규화이다.
길이 정규화는 모든 문서 벡터를 길이가 1인 단위벡터로 만드는 것이다.
길이 정규화 하는 방법은 쉽다. 모든 단어들의 스코어들의 제곱을 합한 뒤, 루트를 씌우면 된다.

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

3) Cosine Similarity : cos(q,d)

자, 이제 길이문제도 해결 되었으니, 각도를 고려한 유사도 계산법을 알아보자.
벡터 스페이스 모델에서 가장 많이 사용되는 문서와 질의어 간의 유사도 계산법이 바로, 코사인 유사도 방법이다.

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

위의 수식을 보면 뭐가 복잡하고 어려워 보이지만, 이것도 전혀 어렵지 않다.

두 벡터에서 동일한 단어들끼리의 곱의 합 / 문서 1 단어의 제곱의 루트 * 문서 2 단어의 제곱의 루트

이것이다. 즉 벡터의 내적에 단위 백터들의 곱을 나누면 되는 것이다.

<<길이 정규화 설명 끝>>