

비정형 빅데이터 분석의 응용과 실습

Week-07. Search Engine - Part 2

문서 검색

Boolean Retrieval

- 쿼리 연산에 대해서 두가지 중 하나의 결과를 보여줌
 - True or False
 - Exact-match
- 일반적으로 Query는 부울 연산을 이용해 제공됨
 - AND OR NOT
- 기본 가정은, "검색된 모든 결과는 동일하게 관련된 내용이다"

문서 검색

Boolean Retrieval

- 아직도 많은 검색 시스템은 부울 연산을 활용
 - 이메일, 인스타그램
- 일부 도메인에 대해서는 매우 효과적인
 - 특허 검색
 - 법률 검색
 - 개발자 디버깅 검색

문서 검색

Boolean View

- 각 행은 특정한 단어(term)을 표현
 - 각 단어가 들어간 문서는?
- 쿼리 실행
 - 검색어로 들어온 Term을 고르고
 - Boolean 연산을 적용

Term	Doc1	Doc2	Doc3	Doc4	Doc5
Hello	0	0	0	1	0
My	1	0	1	0	1
Love	0	0	1	0	0
It	1	1	1	1	1
Is	1	1	1	1	1
Very	0	1	0	0	1
Cold	0	0	0	1	0
On	1	0	1	1	1
This	0	1	1	1	1
Island	0	1	0	0	0

문서 검색

Boolean 검색 예제

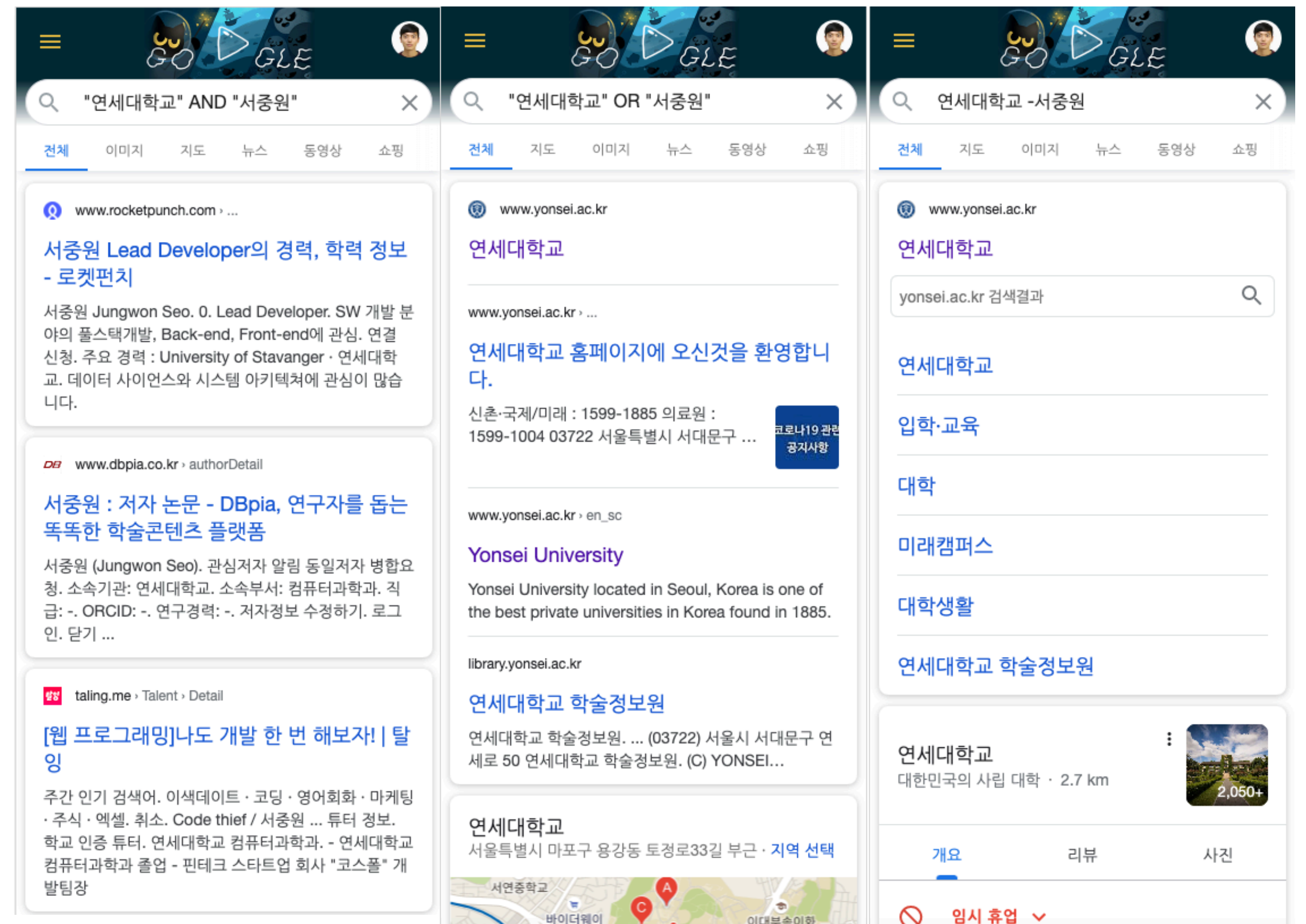
Term	Doc1	Doc2	Doc3	Doc4	Doc5
This	0	1	1	1	1
On	1	0	1	1	1

- This **AND** On : Doc3, Doc4, Doc5
- This **OR** ON : Doc1, Doc2, Doc3, Doc4, Doc5
- This **AND NOT** On : Doc2
- On **AND NOT** This : Doc1

문서 검색

구글에서 Boolean 검색 예제

- AND 연산자: "연세대학교" **AND** "서중원"
- OR 연산자: "연세대학교" **OR** "서중원"
- NOT 연산자: 연세대학교 -학술정보원



문서 검색

Boolean 검색 장/단점

- 장점

- 검색 결과에 대한 설명이 쉬움 (포함/미포함)
- 다양한 요소들이 검색에 함께 포함될 수 있음 (이미지가 포함되어있냐 아니냐 등등)
- 효율적인 연산 (시작과 동시에 많은 문서들이 제외 될 것이기 때문에)
- 관련된 문서를 절대 놓치지 않음

- 단점

- 검색 결과의 퀄리티는 사용자의 쿼리 작성에 의해 달려있음
- 문서간 랭킹이 X
- 단어가 포함되어 있지 않지만, 관련된 문서를 검색 할 수 없음

문서 검색

Rank Retrieval

- $\text{score}(d, q)$
 - 주어진 쿼리 q 에 대해서 각각의 문서의 점수를 계산
 - Query = "Hello world"
- How?
 - $\omega_{t,d}$: 문서(d)와 Term(t)과의 **가중치** 계산
 - $\omega_{t,q}$: 쿼리(q)와 Term(t)과의 **가중치** 계산
 - 그리고 그 둘의 내적을 통한 **유사도** 계산
 - 각각의 문서에 대한 점수 반환

$$\text{score}(d, q) = \sum_{t \in q} \omega_{t,d} \cdot \omega_{t,q}$$

Scoring

Example 1 - Term Frequency Weighting

- $f_{t,d}$: 문서 d에서 term t가 등장한 횟수
- $f_{t,q}$: 쿼리 q에서 term t가 등장한 횟수
- Example
 - 쿼리 : Hello, Hello world
 - 텀 : Hello
 - 문서 : "Hello, Hello, Hello world, programming is very fun"
 - $f_{t,d} : 3$
 - $f_{t,q} : 2$

$$score(d, q) = \sum_{t \in q} \omega_{t,d} \cdot \omega_{t,q}$$

$$\omega_{t,d} = \begin{cases} 1, & f_{t,d} > 0 \\ 0, & \text{otherwise} \end{cases}$$

$$\omega_{t,q} = f_{t,q}$$

Scoring

Example 2 - Log frequency Weighting

- 단순 횟수 보다 해당하는 단어의 중요도를 같이 고려하는게 좋지 않을까?

$f_{t,d}$	$W_{t,d}$
0	0
1	1
2	1.3
10	2
1000	4

$$\omega_{t,d} = \begin{cases} 1 + \log f_{t,d}, & f_{t,d} > 0 \\ 0, & \text{otherwise} \end{cases}$$

$$score(d, q) = \sum_{t \in q} \omega_{t,d} \cdot \omega_{t,q}$$

$$score(d, q) = \sum_{t \in q} (1 + \log f_{t,d}) \cdot \omega_{t,q}$$

벡터 공간 모델

단어와 문서를 표현하기 위한 방법

- 1960-70년대 정보검색 연구분야의 기반이 되는 컨셉
- 아직도 많이 활용됨
- 다음과 같은 Framework를 구현하기에 간단하고 직관적임
 - Term weighting
 - Ranking
 - Relevance feedback

벡터 공간 모델

단어와 문서를 표현하기 위한 방법

- 문서와 쿼리는 term의 weight들의 벡터로 표현됨

$$D_i = (d_{i1}, d_{i1}, \dots, d_{it})$$

- 쉽게 표현하면, 문서와 쿼리는 단어들의 중요도에 대한 벡터임

$$Q = (q_1, q_2, \dots, q_t)$$

- Term과 Document의 매트릭스로 표현

	Term ₁	Term ₂	...	Term _t
Doc ₁	d ₁₁	d ₁₁	...	d _{1t}
Doc ₂	d ₂₁	d ₂₂	...	d _{2t}
.	.			
Doc _n	d _{n1}	d _{n2}	...	d _{nt}

우리가 보관하고 있는 모든 문서를 문서-Term 매트릭스로 표현

Bag-of-Words Model

단어와 문서를 표현하기 위한 방법

- 비정형(텍스트) 데이터를 매트릭스화 할 수 있는 가장 간단한 방법
- 단점은, 순서를 고려하지 못한다는 점
 - "John is smarter than Mary" == "Mary is smarter than John"

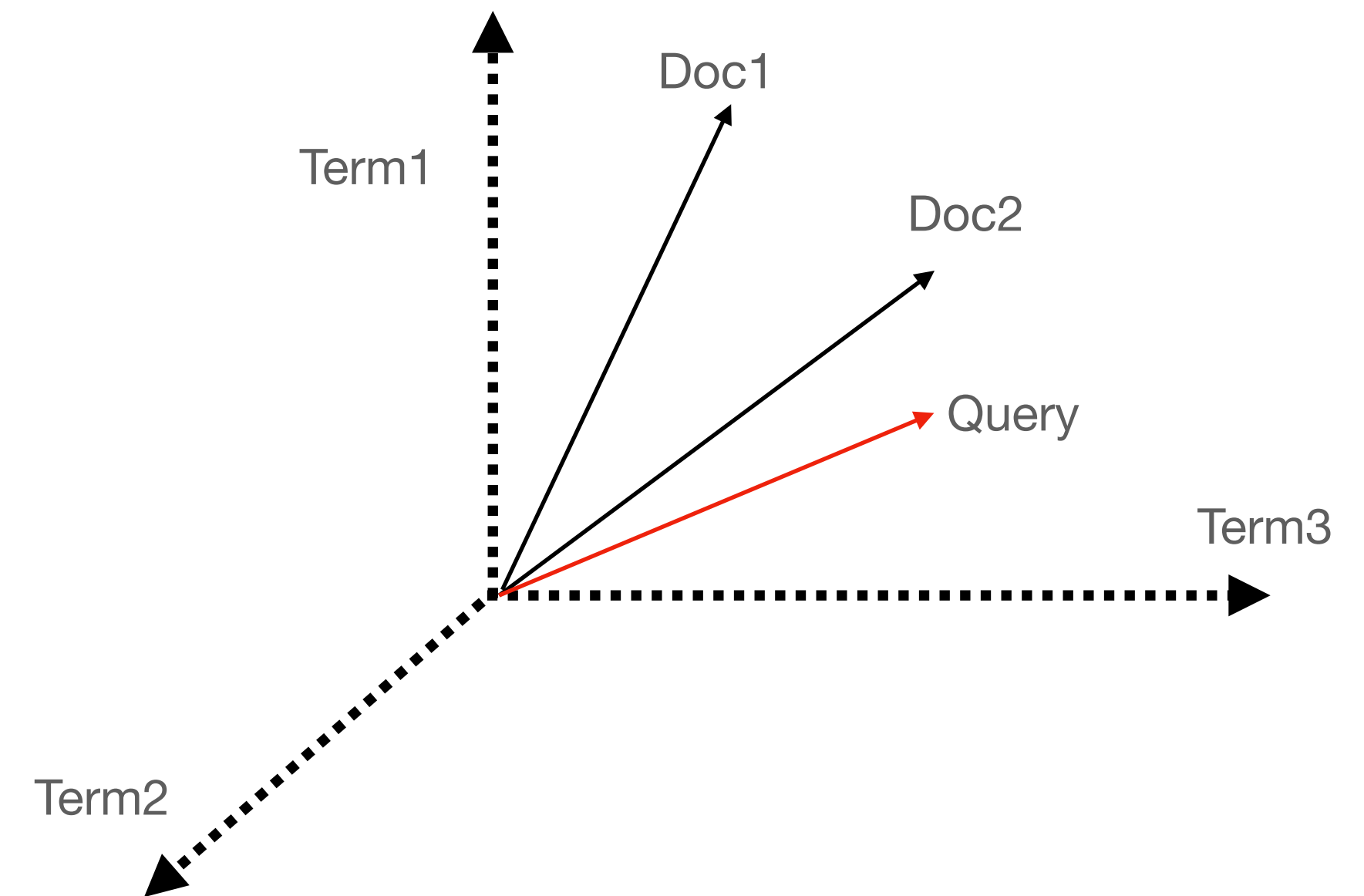
문서 점수화

이 쿼리와 가장 비슷한 문서는 무엇일까?

- 이전 슬라이드에서, 쿼리와 다큐먼트 둘 다 벡터로 표현을 했으므로, 문제를 재정의 가능
 - 두 벡터 간의 유사도는 어떻게 구할 수 있을까?

- Cosine 유사도

- $$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}},$$
- θ 가 작을 수록 1에 가까워 진다.



단어 가중치

어떤 단어가 얼마나 중요한가?

- 직관적으로 생각해보면,
 - 문서에 많이 등장하는 단어는 높은 가중치를 가져야 한다
 - 빅데이터 라는 단어가 많이 등장하면, 빅데이터에 관련된 문서일 확률이 높다.
 - 많은 문서에서 등장하는 단어는 낮은 가중치를 가져야 한다
 - 나 / 너 / 우리 / 그리고 등의 불용어들
- 수학적으로 계산을 하려면?
 - 단어 빈도수 (Term frequency) : TF
 - 역문서 빈도수 (Inverse document frequency) : IDF

TF-IDF

너무 흔하지도 않지만, 너무 희귀하지도 않음

- TF

- Binary TF = {0, 1}
- Raw frequency TF = 빈도수
- Normalized TF = 빈도수/문서길이
 - 문서길이: 문서내의 전체 단어수
- Log-normalized TF = $1 + \log(\text{빈도수})$

- ITF

- $idf_t = \log \frac{N}{n_t}$
 - N: 전체 문서수, n_t : 단어 t를 포함하고 있는 문서의 수
 - 예를 들어 N=100, $n_t=50 \rightarrow idf = \log(2)$
 - N=100, $n_t=100 \rightarrow idf = \log(1) = 0$
- 해석: IDF가 높다? 단어가 등장하는 문서가 적다

weighting scheme	document term weight	query term weight
1	$f_{t,d} \cdot \log \frac{N}{n_t}$	$\left(0.5 + 0.5 \frac{f_{t,q}}{\max_t f_{t,q}}\right) \cdot \log \frac{N}{n_t}$
2	$1 + \log f_{t,d}$	$\log \left(1 + \frac{N}{n_t}\right)$
3	$(1 + \log f_{t,d}) \cdot \log \frac{N}{n_t}$	$(1 + \log f_{t,q}) \cdot \log \frac{N}{n_t}$

<https://en.wikipedia.org/wiki/Tf%E2%80%93idf>

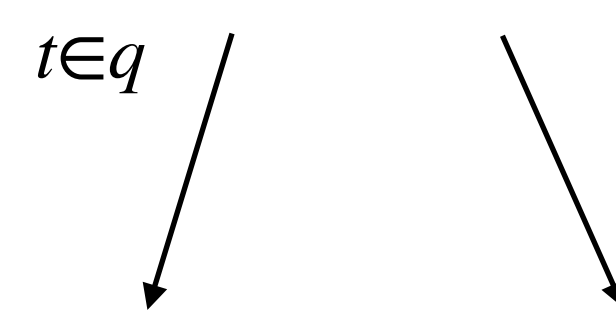
- TF-IDF

- TF와 IDF를 동시에 고려하기 위한 수식
 - $tf-idf = tf \cdot idf$
- TF는 해당 문서에서의 단어의 중요도를 나타내고
- IDF는 해당 단어의 문서 전체에서의 중요도를 나타냄

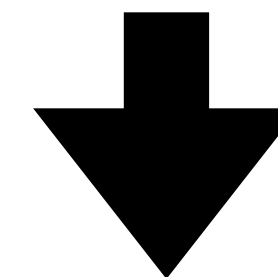
Boolean vs Term Weighting

Boolean 검색과의 차이점

- 단어의 수가 유사도에 영향을 미친다
- 단어의 가중치가 유사도에 영향을 미친다
- 무엇보다도! 문서의 Rank를 계산할 수 있다.

$$Score(q, d) = \sum_{t \in q} \omega_{t,q} \cdot \omega_{t,d}$$

$$\omega_{t,q} = \frac{tfidf_{t,q}}{\sqrt{\sum_t tfidf_{t,q}^2}} \quad \omega_{t,d} = \frac{tfidf_{t,d}}{\sqrt{\sum_t tfidf_{t,d}^2}}$$

$$cosine(d, q) = \frac{\sum_t \omega_{t,d} \cdot \omega_{t,q}}{\sqrt{\sum_t \omega_{t,d}^2} \sqrt{\sum_t \omega_{t,q}^2}}$$



$$cosine(d, q) = \frac{\sum_t tfidf_{t,d} \cdot tfidf_{t,q}}{\sqrt{\sum_t tfidf_{t,d}^2} \sqrt{\sum_t tfidf_{t,q}^2}}$$

BM25

Ranking function

- 엘라스틱 서치에서 기본적으로 사용하는 랭킹 알고리즘
- Term 가중치를 위한 세가지 핵심 원리
 - Inverse document frequency
 - Term frequency
 - Document length normalization

$$score(d, q) = \sum_{t \in q} idf_t \cdot \frac{f(t, d) \cdot (k_1 + 1)}{f(t, d) + k_1 \cdot \left(1 - b + b \cdot \frac{|d|}{avgdl} \right)}$$

Elasticsearch

오픈소스 검색엔진

- Lucene 라이브러리 기반의 검색엔진
- REST API 형태로 접근
- 가장 대중적인 엔터프라이즈 검색엔진
- 최근 ELK(Elasticsearch, Logstash, Kibana) 스택이라는 빅데이터 수집 및 분석에 많이 사용됨



Elasticsearch

RDB vs ElasticSearch

Text	Document
Big	Doc1, Doc2, ..
Data	Doc1, Doc3, ...
...	...

O(1)

Document	Content
Doc1	Big data is very big
Doc2	Data science is science
...	...

O(n)

Seach : "Big"

Elasticsearch

RDB vs Elasticsearch

관계형 데이터베이스 (mysql)	엘라스틱서치
Database	Index
Table	Type
Row	Document
Column	Field
Schema	Mapping
Index	모두 Index되어있음
SQL	Query DSL

E.O.D