

2장

2. Vector Space Model

* Information Retrieval

- large corpus에서 특정 information needs를 만족하는 문서 찾기

\hookrightarrow query로 표현됨.

- User가 관련있는 text를 가질수 있도록 하는 방식

① Pull mode : User가 조기화.

② Push mode : system이 조기화. User가 필요로 만한 걸 알아서 알려줌.

* Pull mode의 입력 방식

- Querying : User가 직접 입력.

- Browsing : User는 문서 구조의 흐름과 경로를 따라 정보를 탐색함.
: ex) Pagerank

* 탐색 (Retrieval) 종류

- Text Retrieval : Unstructured text., 결과가 ambiguous, relevant한 문서.

- DB Retrieval : Structured data, 결과가 well-defined, matched records가 나옴.

- Text Retrieval은 Empirically-defined problem.

- 무언가 좋다, 나쁘다는 사용자에 의해 결정.

* Text Retrieval

- Vocabulary $V = \{w_1, \dots, w_n\}$

Query $q = \{q_1, \dots, q_m\}$

Document $d_i = \{d_{i1}, \dots, d_{im}\}$

Collection $C = \{d_1, \dots, d_m\}$

word sequence 3 표현.

- $R(q) \subseteq C$: q와 관련있는 문서. User-dependent 함.

- 우리의 목표 : $R(q)$ 을 모으니까, 이의 특성을 $R'(q)$ 구하기.

- $R'(q)$ 구하기

"관련도"의 정의 필요.

정답률 표기.

① Document Selection : 관계 있다, 없다는 classification. (absolute relevance)

② Document Ranking : $R'(q) = \{d \in C \mid f(d, q) > \delta\}$. δ 는 user-defined 값.
(relative relevance). 무가 무보다 더 영관성 있다.

+,- 노드는 같이 중요.

- Document Selection 문제점

- Under-constrained query : 관련있는 문서가 너무 많음.

- Overt-constrained query : 관련있는 문서가 없음.

- relevance는 상대적인 것. prioritization 필요.

- Document Ranking

- 관련 문서들이 ranking되어 순위가 나옴.

- Probability Ranking Principle : $p(R=1 \mid d, q)$: 확률적 방법.
(PRP)

- $f(d, q)$ 을 정의해야 함.

- def of relevance 정의 필요.

- vector space model : $f(d, q) = \text{sim}(d, q)$

- probabilistic model : $f(d, q) = p(R=1 \mid d, q)$

- language model : $f(d, q) = p(q \mid d)$

* Retrieval Models

각 단어들에 대한 합.

$$f(q, d) = \sum_{w \in q} g(w, d)$$

~ TF와 IDF를 고려한 점수.

- TF (Term frequency) : 문서 d 내에서 빈도수.

- IDF (Document Frequency) : 전체 문서 set에서 빈도수.

- Document Length 또한 고려할 사항. 그러면 TF가 유리할 가능성이 높음.

* IR 과정

- ① d_1, d_2 를 vector로 표현.
- ② vector간의 similarity 측정.

* Vector Space Model

- 각 d_i 는 concept vector로 표현 가능.
 - Concept은 각 dimension. 수치화하면 각 word.
 - 각 값은 그 단어의 중요도. weight이 됨. 쉽게 하면 발생 빈도수.
- 문제
 - "basic concept" 정의? Orthogonal한게 좋은 Bag of Words로.
 - Weights of concept 할당? TF/IDF/OL로.
 - 두 vector 사이 distance 정의?

* Bag of Words Model

- 단어가 concept 단위가 됨.
 - 장점 : 단순함. 문서의 topics을 나타내기도 함.
 - 단점 : 문장의 구조를 잊어버림. 구문도 나타내기 X.
 - 단어 말고 각 term을 concept 단위로 하자.
 - tokenization : 언어마다 다른 기준. (한글 : 형태소, morpheme)
 - 불필요한 단어 처리 : stop words. Normalization 등
-) pre-processing

* BoW with N-grams

- BoW의 구조, 구조가 업데이트되는 단점 극복.
- 각 concept 단위 = 연속된 N개의 단어들.
- 장점 : local dependency 살리기 가능.
- 단점 : Vocab size가 $O(V^N)$ 이 됨.
(bigram : $O(V^2)$, --- : Vector size가 커짐)

단계) - Tokenization \rightarrow Stemming, Normalization \rightarrow N-gram \rightarrow Stop word filtering.

* BoW Unigram (단어 하나)

- 가중치 : 모든 단어는 independent : 틀림.
- 장점 : simple
- 단점 : 각 vector는 linearly independent 하지 않다.
 - : 물법, 구문 등 구조를 알아버림.
 - : 길이가 $|V|$ 로 너무 길다.

그래도
간단해니까
쓰기로 함.

* Assign Weights to words

- Corpus-wise : IDF : corpus 전체에서 등장 freq.
- document-wise : TF : doc 내부 등장 freq.
- $\text{식} = \frac{\text{TF}}{\text{DF}} = \underbrace{\text{TF}}_{\text{inverse}} \times \underbrace{\text{IDF}}_{\text{높은수록 증음}}$

* Term Frequency

- 그 문장에서 많이 등장하는 term의 수를 중요함.

- 식

$$\cdot \text{tf}(t, d) = C(t, d)$$

- doc에서 t가 등장한 횟수 count.

- Normalization

- 문서 길이가 길면 TF 값이 높다.

• 1번 등장 << 2번 등장 이지만, 100번 등장 \approx 200번 등장. \Rightarrow log-scale 사용

- Sub-linear tf scaling

$$\cdot \text{tf}(t, d) = \begin{cases} 1 + \log(C(t, d)) & \text{if } C(t, d) > 0 \\ 0 & \text{otherwise} \end{cases}$$



- Maximum tf scaling

$$\cdot \text{tf}(t, d) = \alpha + (1-\alpha) \cdot \underbrace{\frac{C(t, d)}{\max_t C(t, d)}}_{\approx 1}, \quad \text{if } C(t, d) > 0$$



* Document Frequency

- 같은 문서에서 등장한 term의 수를 출현함.

- IDF 속

$$\cdot \text{idf}(t) = \log_{10} \left(\frac{N}{\text{df}(t)} \right)$$

→ N : 총 문서 개수
→ $\text{df}(t)$: 키워드 t 가 등장한 문서 개수
→ Non-linear Scaling. \log 안쓰으면 줄로드 크기가 너무 커짐.

- ex) 10개의 문서 중 키워드

d_1 에서 3번, d_2 에서 2번, d_3 에서 1번 등장함.

$$\Rightarrow \text{idf}(t) = \log_{10} \left(\frac{10}{3} \right)$$

문서들은 상관X.
→ d_1, d_2, d_3 : 3개. 등장 횟수는 상관X

- IDF는 query가 단어 하나면, 의미 없다. 2개 이상 term으로 구성된 query才有.

- Collection frequency

- 나타난 횟수. DF는 나타난 문서 개수를 의미.
- DF가 더 의미 있음.

term 만이 input.

* TF-IDF weighting

$$- w(t, d) = \text{tf}(t, d) \times \text{idf}(t)$$

- tf와 idf에 다양한 function으로 다양한 조합 가능.

- Document Length 고려

- 더 긴 문서일수록 높은 TF를 가질 확률이 높다.
- pivoted length normalizer

$$\text{normalizer} = 1 - b + b \cdot \frac{|d|}{\text{avdl}}, b \in [0, 1]$$

→ $|d|$: average document length.

이제 블로고 들어감 → $|d| < \text{avdl}$: norm ↓ : (블로고) TF ↑

→ $|d| > \text{avdl}$: norm ↑ : TF ↓

최소값 = $1 - b$. 최대 = 1 이상.

* Okapi BM25

- 대표적인 TF-IDF VSM

$$f(q_i, d) = \sum_{t \in q_i \cap d} \frac{(k_i + 1) \cdot C(t, d)}{C(t, d) + k_i \cdot (1 - b + b \cdot \frac{|d|}{avgdl})} \log \frac{M+1}{df(t)}$$

TF X IDF

* Measuring the Similarity

- 두 문서 간의 Similarity 측정 법?

① Euclidean distance 방향성 X

- 두 vector 사이 거리.

- 길이만 생각됨. 중요한건 방향성인데, 고려가 잘 안됨.

(동쪽에 등장하는게 더 중요)

$$\text{dist}(d, d') = \sqrt{\sum_{t \in V} [tf(t, d) \cdot idf(t) - tf(t, d') \cdot idf(t)]^2}$$

• 긴게 더 높은 값.

② Inner Product 길이 norm \rightarrow cosine sim

- 단점 : Vector 간의 길이가 다르므로, 특징이 잘 안나타남.

- Need Normalization

• Cosine Similarity : inner product의 normalization.

$$\cos(d, d') = \frac{d \cdot d'}{|d| |d'|}$$

한 vector를 다른 vector의 projection.
또거나 두 vector의 overlap 정도를 measure.

$$\cdot \cos(q, d) = \frac{q}{\|q\|_2} \cdot \frac{d}{\|d\|_2}$$

Using L2 norm. $\sqrt{\sum x_i^2}$ ↪ > ↪

그냥 두 normalized vector의 inner product.