8월 22일

[[(51, 2, 78), (51, 26, 78), (51, 36, 78), (78, 17, 57), (57, 36, 51)], [(40, 17, 74), (74, 17, 51), (74, 23, 51), (74, 36, 51), (51, 36, 40)], [(3, 2, 85), (3, 7, 85), (3, 10, 85), (3, 17, 85), (3, 21, 85), (3, 26, 85), (3, 36, 85), (85, 17, 62), (62, 36, 3)]]

[(51, 2, 78), (51, 26, 78), (51, 36, 78), (78, 17, 57), (57, 36, 51)] 처럼 하나의 cycle passage에 대해서 triple이 5개가 들어있다.

이것을 [(51, 2, 78), (78, 17, 57), (57, 36, 51)],[(51, 26, 78), (78, 17, 57), (57, 36, 51)], [(51, 36, 78), (78, 17, 57), (57, 36, 51)] 3개로 만들어야한다

```
def transform_cp(cp):
    result = []
    for sublist in cp:
        triple_combinations = []
        for triple in sublist[:-2]:
            new_sublist = [triple, sublist[-2], sublist[-1]]
            triple_combinations.append(new_sublist)
        result.extend(triple_combinations)
    return result
```

[(25, 17, 52), (65, 9, 25), (65, 36, 25)]

transform_cp를 작동하면

[(25, 17, 52), (52, 24, 65), (52, 28, 65), (52, 37, 65), (65, 9, 25), (65, 36, 25)] 이 [(25, 17, 52), (65, 9, 25), (65, 36, 25)] 로 바뀌었는데 이건 옳지 않아

왜냐하면 (25,17,52) 다음에는 52노드가 있는 triple이 나와야해. 규칙은 (head1, relation1, tail1), (head2, relation2, tail2), (head3, tail3, head3) 이라면 head1 = head3, tail1 = head2, tail2 = head3 가 되야해. 그래야 길이 3의 사이클 그래프에 속하는 노드 와 에지의 triple이 완성되지

BM25

BM25(a.k.a Okapi BM25)는 주어진 쿼리에 대해 문서와의 연관성을 평가하는 랭킹 함수로 사용되는 알고리즘으로, TF-IDF 계열의 검색 알고리즘 중 SOTA 인 것으로 알려져 있다. IR 서비스를 제공하는 대표적인 기업인 엘라스틱서치에서도 ElasticSearch 5.0서부터 기본(default) 유사도 알고리즘으로 BM25 알고리즘을 채택하였다.

BM25 step by step 살펴보기

BM25는 Bag-of-words 개념을 사용하여 쿼리에 있는 용어가 각각의 문서에 얼마나 자주 등장하는지를 평가한다.

키워드 $q_1, ..., q_n$을 포함하는 쿼리 Q가 주어질 때 문서 D에 대한 BM25 점수는 다음과 같이 구한다.

$$score(D,Q) = \sum_{i=1}^{n} IDF(q_i) * \frac{f(q_i,D) * (k_1+1)}{f(q_i,D) + k_1 * (1 - b + b * \frac{|D|}{avgdl})}$$

문서 D에서 q_i의 term frequency / 문서 D의 길이 / 파라메터 / 문서 집합의 평균 문서 길이

q_i : 쿼리에서 i번째 토큰 (형태소 / bi-gram /BPE 등을 사용할 수 있음)

IDF(q_i) : 쿼리의 i번째 토큰에 대한 inverse document frequency

- 자주 등장하는 단어는 penalize 하여 지나치게 높은 가중치를 가지게 되는 것을 방지함.
- 예를 들어 "the"라는 단어는 영어 문서에서 아주 자주 등장하는 단어이기 때문에 쿼리에 있는 the라는 글자가 문서에 자주 나타났다고 해서 의미 있는 것이 아님.
- 루씬(Lucene)에서는 다음과 같은 식으로 IDF를 계산함:

$$IDF(q_i) = \ln(1 + \frac{(docCount - f(q_i) + 0.5)}{f(q_i) + 0.5})$$

총 문서의 개수 / 해당 단어를 포함하는 문서의 개수

- 예를 들어 총 10개의 문서가 있을 때 10개의 문서에서 모두 등장하는 단어(the 등)는 IDF값이 0.05, 한 번만 등장하는 단어는 1.99의 값을 가짐.
- BM25 스코어링 식에서 IDF값이 곱해진다는 것은 자주 등장하지 않는 단어에 더 큰 가중치를 둔다는 것을 의미함.

$|D|/avgd_l$: 문서의 길이에 대한 부분

- 해당 문서가 평균적인 문서 길이에 비해 얼마나 긴지를 고려함.
- 이 항은 점수 계산에 있어 분모에 있기 때문에, 평균 대비 긴 문서는 penalize됨.

- 열 페이지가 넘는 문서에서 어떤 단어가 한 번 언급되는 것과, 트위터같은 짧은 문장에서 단어가 한 번 언급되는 것 중에서는 두 번째 경우가 더 의미 있다고 보는 것.

b : 일반적으로 0.75 사용

- b가 커지면 문서의 길이가 평균 대비 문서의 길이에 대한 항의 중요도가 커짐.
- b가 0에 가까워지면 문서의 길이에 대한 부분은 무시됨.

f(q_i, D) : 쿼리의 i번째 토큰이 문서 D에 얼마나 자주 나타나는가 (Term Frequency)

- 쿼리에 있는 키워드가 문서에 자주 나타나면 점수가 높아짐

k1 : TF의 saturation을 결정하는 요소 (1.2~2.0의 값을 사용하는 것이 일반적)

- 하나의 쿼리 토큰이 문서 점수에 줄 수 있는 영향을 제한함.
- 직관적으로 k1은 어떤 토큰이 했을 때, 이전까지에 비해 점수를 얼마나 더 높여주어야 하는가를 결정함.

  한 번 더 등장

- 어떤 토큰의 TF가 k1보다 작거나 같으면 TF가 점수에 주는 영향이 빠르게 증가하는데, k1번 등장하기 전 까지는 해당 키워드가 문서에 더 등장하게 되면 문서의 BM25 스코어를 빠르게 증가시킨다는 뜻임.
- 반대로 토큰의 TF가 k1보다 크면 TF가 점수에 주는 영향이 천천히 증가하고, 해당 키워드가 더 많이 등장해도 스코어에 주는 영향이 크지 않음.
- k_1의 존재로 인해 BM25에서 term frequency는 어느정도 이상부터는 점수에 주는 영향이 거의 증가하지 않음.