**자연어처리 및 정보검색**

Text Ranking Homework



20164245

홍진

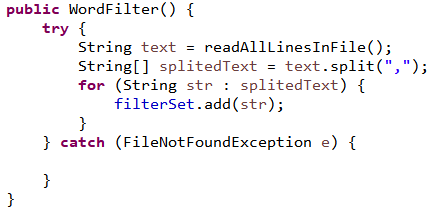
**Contents**

1. How to Design vector space model
2. Ranking
3. How to improve Ranking
4. **How to design vector space model with N dimension**

* Vector space model이란 개별 단어에 대응되는 차원을 가진 space model이다.

하지만 단어의 개수가 N개가 되면 dimension이 너무 높아져 매우 spares한 벡터가 되어 대부분 값이 0이 될 것이다.

따라서 사용자가 원하는 쿼리 역시 vector로 표현해주고 document vector의 각도의 편차를 이용하여 유사도를 나타낸다.

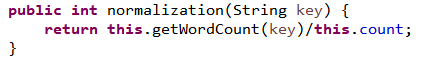


따라서 각각의 word를 tokenize한 다음, stopword를 제거해주어 불 필요한 단어들을 줄였다.

또한 각각의 쿼리와 document들의 cosine similarity 값을 구하기 위하여 가중치를 계산하였다.



각각의 document에서 단어마다 term frequency(tf)를 구하고, inverse index를 통하여 구한 inverse document frequency(idf)를 구하여 둘을 곱해주어 가중치를 계산하여 cosine similarity에 적용하였다.

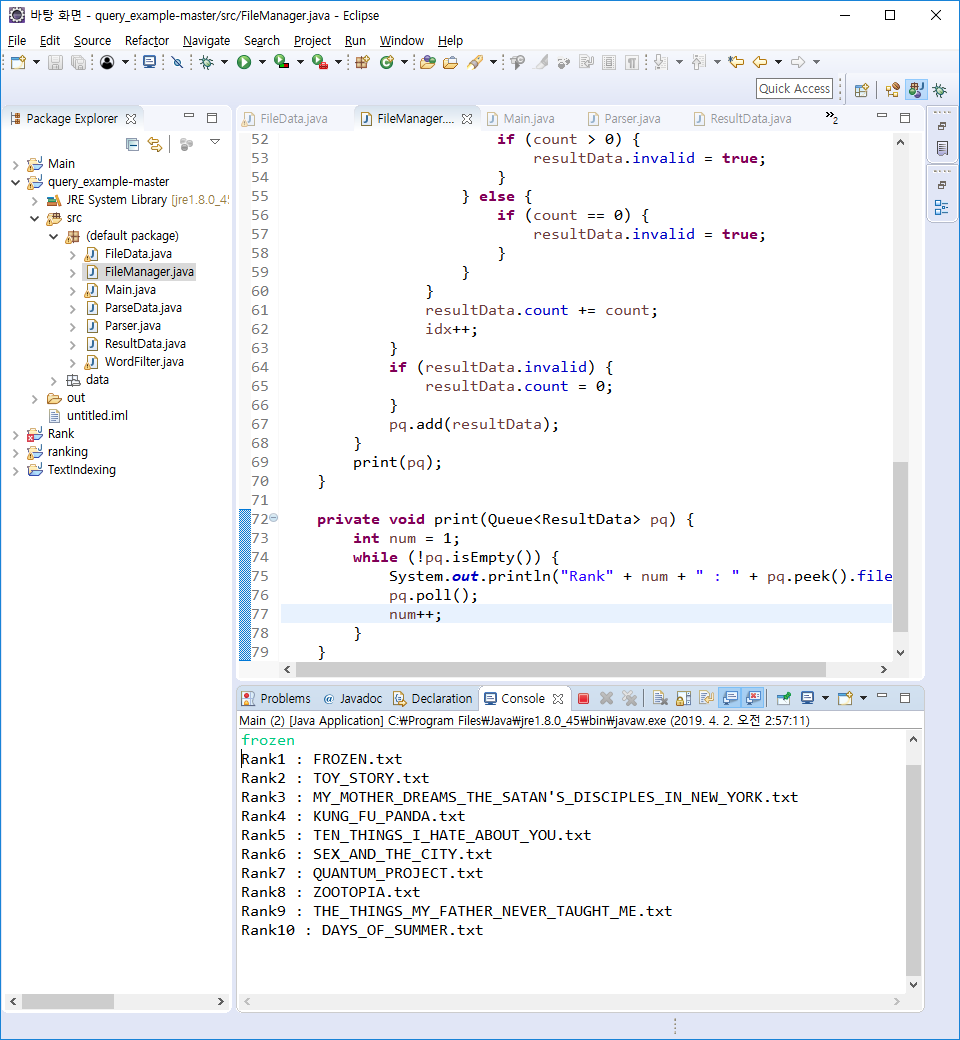


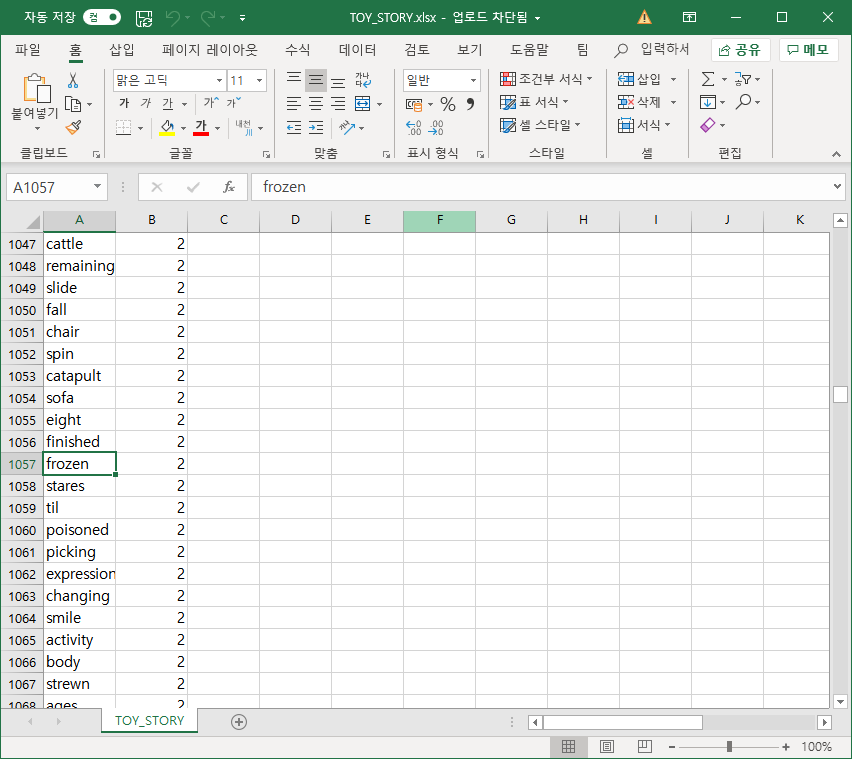
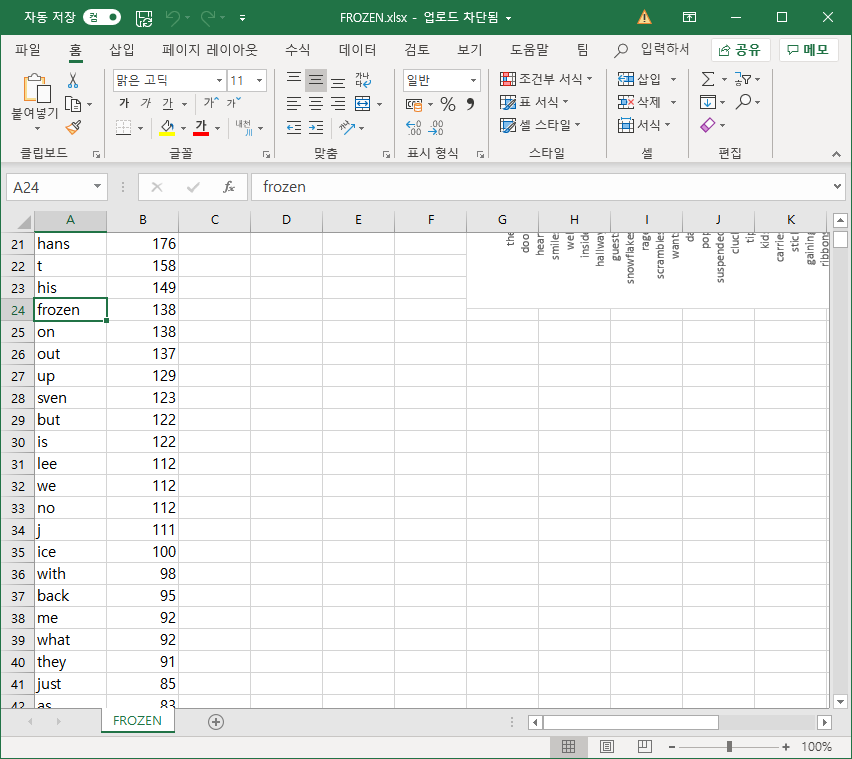
이후 L2 norm을 이용한length-normalized를 통하여 문서의 길이와 상관없이 가중치를 사용할 수 있도록 하였다.

빈도가 낮은 단어들은 ranking에 빈도가 높은 단어들보다 영향이 적을 것이라 생각하여 빈도가 높은 단어들에 가중치를 두어서 dimension을 줄여 performance를 줄이려고 하였다.



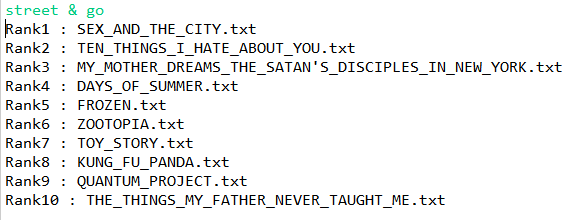
1. **Test your rankings with your own queries.**
2. 1단어의 쿼리문

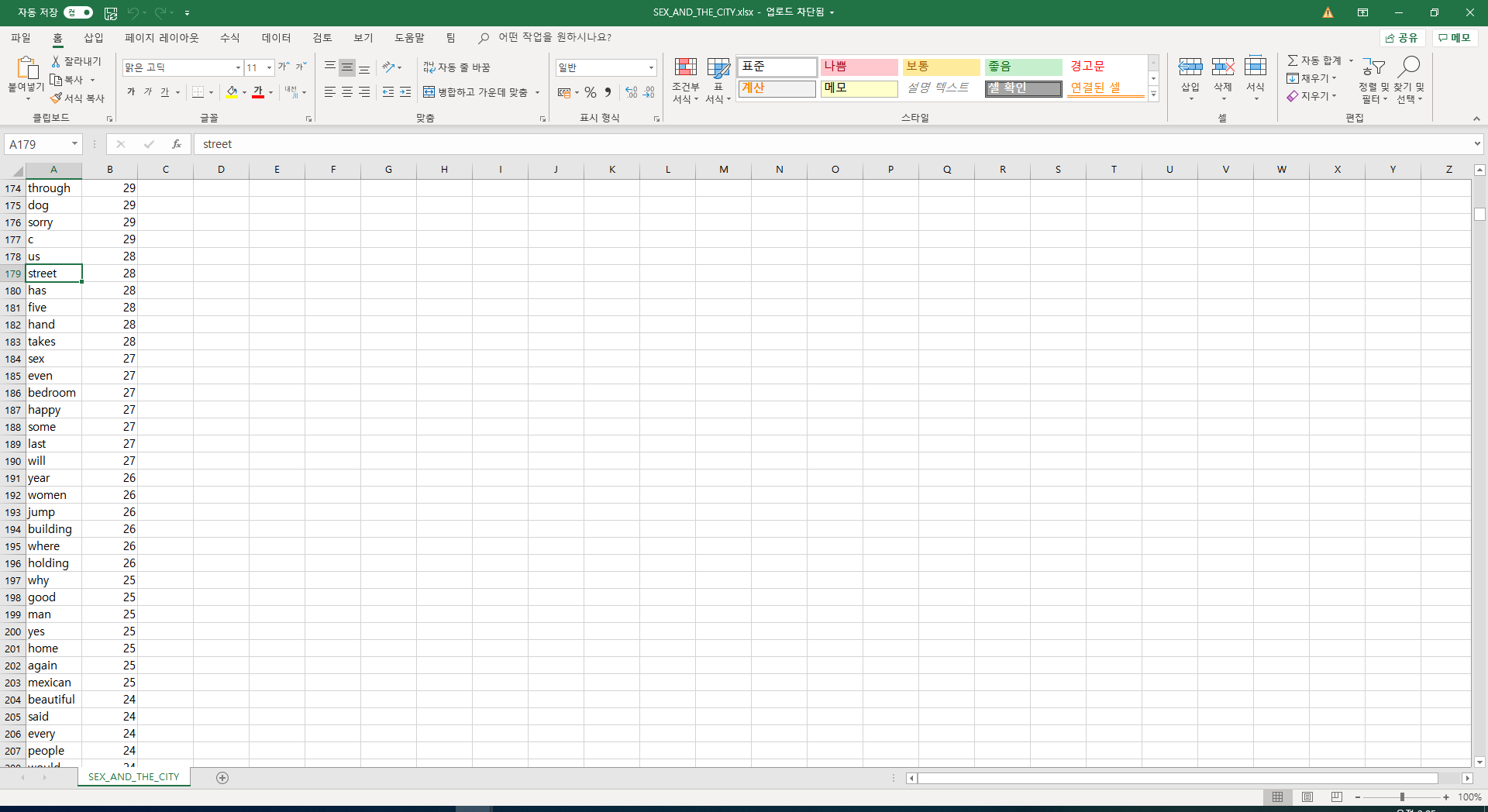


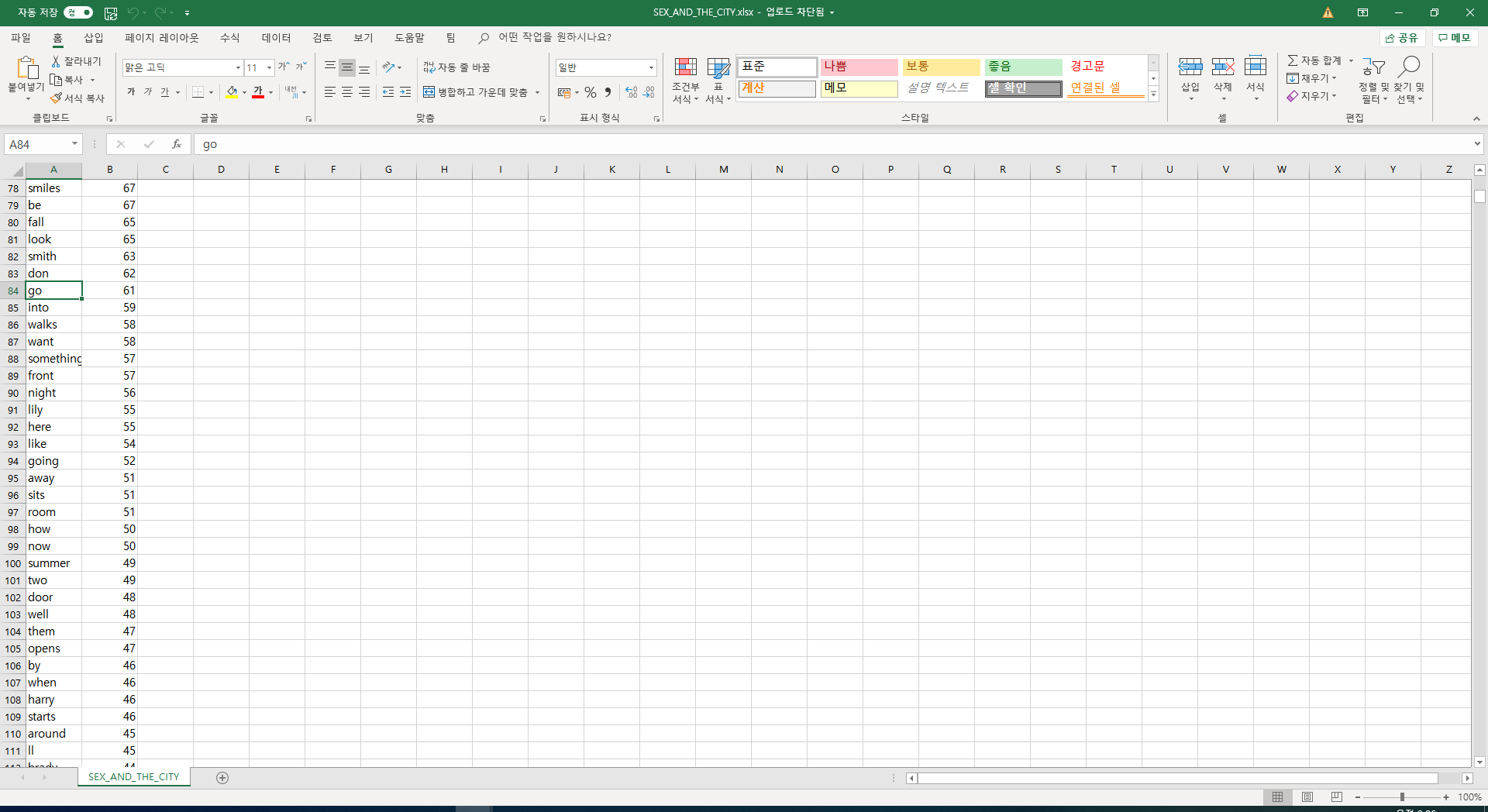
* 위의 사진과 같이 원하는 결과의 값이 나오는 것을 확인할 수 있었다.
* 아래의 엑셀파일과 같이 원하는 ranking 1위는 frozen이었고, 2위는 toy story임을 알 수있다.
* 

1. 2단어 쿼리문 (and)

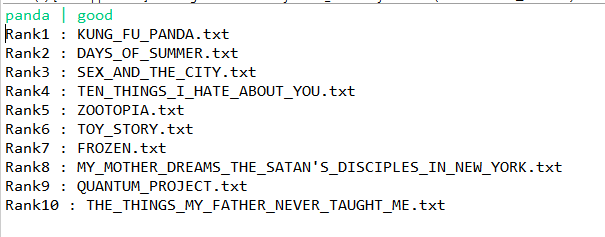
* “street & go” 라는 쿼리문을 실행하였을 때 역시 원하는 결과가 나옴을 확인할 수 있었다.





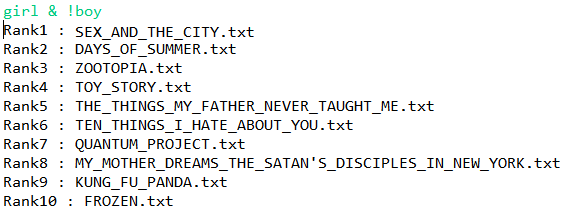


1. 2단어 쿼리문 (OR)



* Or 쿼리문을 실행하여 “panda | good” 쿼리를 실행한 결과 rank 1은 원하는 kung fu panda라는 document가 출력되었지만, rank2가 zootopia가 나올 것이라는 expectation과는 달리 days of summer가 나왔다.

1. Not 쿼리문



* Not 쿼리문 역시 위와 같은 결과가 나오는 것을 확인할 수 있었다.
* 하지만 위의 “girl & !boy”의 쿼리문 실행 결과로 1번 rank가 “days of summer”일 줄 알았지만 예상과는 다르게 rank 6에 있음을 알 수 있었다.

Pros

* 간단한 쿼리문을 통하여 원하는 정보를 얻을 수 있어서 좋다.
* 모델이 단순하여 쉽게 계산하여 결과값을 알아 낼 수 있었다.
* 문서와 쿼리간의 유사도를 연속적인 값으로 계산이 가능하다.
* 부분 일치가 고려 가능하다.

Cons

* 길이가 긴 문서는 낮은 유사도 값을 가져서 잘 표현할 수가 없었다.
* 검색 키워드가 문서내의 단어와 정확히 일치해야 한다.
* 단어가 벡터로 표현되기 때문에 단어의 순서를 고려하지 않아 의미 유사성이 낮다.
* 종속적인 단어는 키워드로 찾아보기 어려운 단점이 있다.

1. **What have you tried to improve your ranking?**



* Ranking을 높이기 위해서 흔한 단어보다 희귀한 단어에 더 가중치를 부여하여 계산한다면 향상된 결과를 얻으려고 하였다.

따라서 high-idf query term을 고려한다면 low-idf term들이 많은 문서가 있을 때 이러한 문서를 제거함으로써 시간을 절약하여 향상하였다

또한 multi-term 쿼리에서 적어도 몇 개 이상의 query term을 포함하는 문서만 계산하여 시간을 줄였다.

Top-K 번째 components를 찾을 때, max heap을 사용하여 위의 root부터 값을 반환하여 쓸 수 있도록 하면 훨씬 효율적일 것이다.

위와 같은 방법 적용 이후 위의 Not 쿼리문을 다시 시행해보면 아래와 같이 원하는 결과를 얻을 수 있음을 확인할 수 있다.

