Question Answering with Keyword

Team : 욕심 많은 민트 초코

Team member : 김정학(2015147557), 박성현(2015147565), 유현석(2015147533)

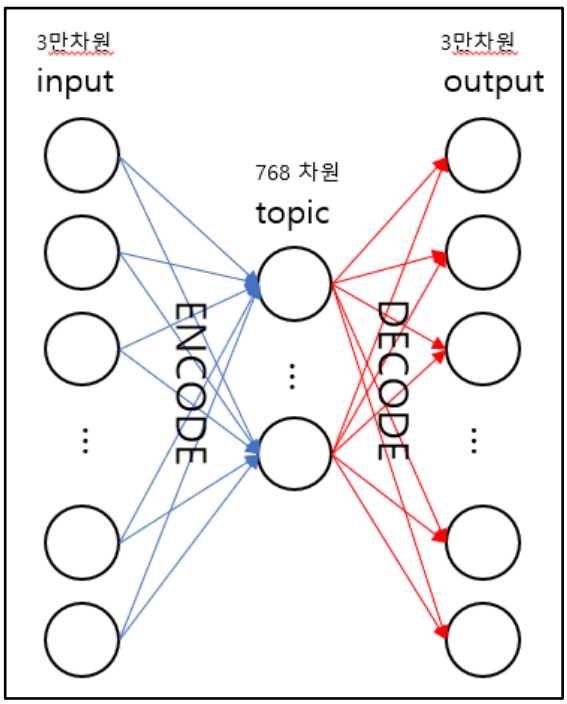
Contents

1. 기존 연구 소개

기존 연구는 BM25 + Topic, Bert + Topic 모델로 이루어져 있다.

① BM25 + Topic

BERT를 사용하지 않고 쿼리(질문)와 도큐먼트(문서)를 매칭해주는 모델은 단어들과 각 단어들이 가지고 있는 토픽을 이용한다. 이 정보들을 BM25 알고리즘을 통해서 문서와 쿼리를 매칭해준다. 주어진 데이터는 쿼리와 문서들로 이루어져 있고 각 단어들이 어떤 토픽을 가지고 있는지는 딥러닝을 통해서 학습을 한다. 이를 위해서 Autoencoder를 사용한다. 쿼리와 문서를 나타내는 방식은 두가지가 있다. 첫번째는 binary 방식이고 두번째는 tf-idf 알고리즘의 idf를 이용하는 것이다. 예를 들어 총 5개의 단어를 가지고 있는 코퍼스속에서 1,1,2,4번 단어로 이루어져있는 문서같은 경우는 바이너리로는 (1,1,0,1,0), idf 방식으로는 다른 문서에서 빈도수에 따라서 바이너리에서 1인 요소에 idf를 곱해준다. idf 방식의 경우에는 다른 문서에서 출현하지 않을수록 값이 높아진다. 주어진 데이터속에서 전체 단어의 수는 총 3만개이다. 따라서 오토인코더는 3만차원의 인풋을 받고 이를 인코더를 통해서 압축화를 하고 압축화되어 있는 텐서를 디코더를 통해서 최대한 원래의 인풋으로 복원을 해야 한다. 이를 도식화하면 아래와 같다.



모델은 (쿼리, 정답 문서, 거짓 문서)를 하나의 데이터로 사용해서 단어가 어떤 토픽을 가지고 있는지를 학습한다. 따라서 쿼리, 정답문서, 거짓문서를 각각 위 모델을 통과시킨후 그 결과값으로 loss를 구해야한다. Z_q를 쿼리를 인코더만 통과시켰을 때의 값, Z_p 정답문서를 인코더만 통과시켰을 때의 값, Z_n을 거짓문서를 인코더만 통과시켰을 때의 값이라고 하자. Loss의 값은 총 3가지 값의 합으로 구해진다. 첫번째로 Z_q Z_p Z_n을 디코더를 통과시켰을 때 인풋과의 차이점을 사용한다. 이는 인풋이 토픽으로 압축이 잘되기 위함이다. 두번째로는 (Z_q-Z_n) - (Z_q-Z_p)을 사용한다. 이는 쿼리와 정답문

서의 토픽 추출이 비슷해지고 쿼리와 거짓문서의 토픽이 멀어지게 하기 위함이다. 마지막으로 Z_q, Z_p, Z_n의 각각 768차원의 값이 1로 수렴하게 하기위한 loss를 사용한다. 이는 학습을 안정화시키기 위함이다. 위 방식으로 모델은 문서와 쿼리의 토픽을 768차원으로 뽑아낼 수 있다. 모델을 학습시킨 후에 단어가 나타내는 토픽을 찾는 방법은 다음과 같다. decode부분의 768차원의 뉴런에서 3만개로 매칭시켜주는 각 가중치를 고려한다. 예를 들어 1번째 뉴런은 3만개의 가중치를 가지고 있는데, 여기서 topk(=10)번째로 값이 큰 가중치까지 정보를 저장한다. 즉 1번째 뉴런에서 가중치가 높은 상위 10개의 단어들이 1,3,5,6,...,11(10개)의 단어들이라면 반대로 1번째 단어는 1번째 토픽을 가진다고 할 수 있다. 이방식으로 각 단어들에 토픽을 연결시켜준다. 이 결과로 어떤 단어는 어려가지 토픽을 가질 수 있고 어떤 단어는 토픽이 아예 매칭이 안될 수도 있다. 이후에 각 문서의 키워드 뒤에 각 단어들이 가지고 있는 토픽을 단순히 더해 데이터를 변형시킨다. 변형된 데이터로 BM25 알고리즘 매칭을 진행하였다.

② Bert + Topic

Bert를 사용한 모델은 먼저 기본 Bert의 동작 방식에 따라 주어진 데이터 Input을 임베딩하는 과정을 거치게 된다.

1) Token Embedding :

각 단어는 word vector table을 모델 입력으로 설정하여 1차원 벡터로 변환된다. 단어를 더 세밀한 단어 조각으로 자르는 것으로 목록에 없는 단어를 해결한다. 예를 들어 playing을 play + ing으로 자르는 방법이다.

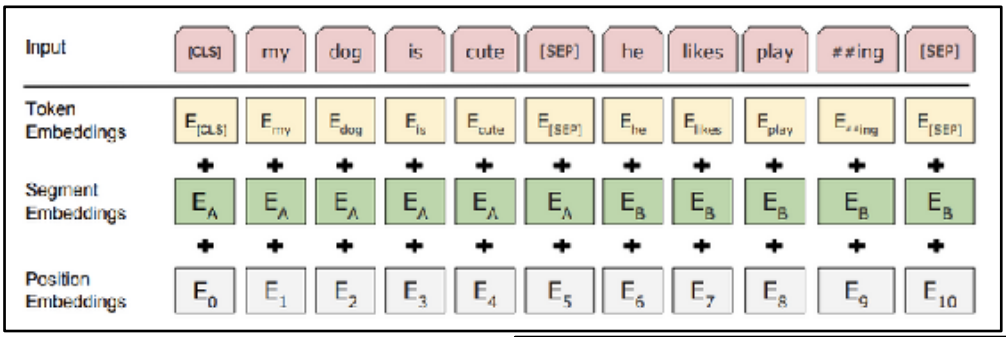
아래의 그림과 같이 "I like dog"를 입력하면 Token Embeddings layer 보내기 전에 두 개의 특수 토큰이 [CLS]의 시작과 [SEP] 끝에 삽입된다. 그 후, 각 단어를 768 차원 벡터로 변환된다.

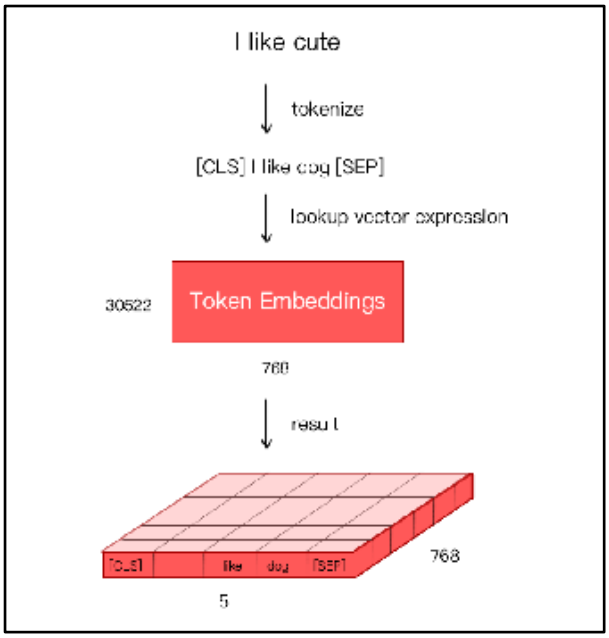
2) Segment Embedding :

Segment Embedding은 토큰화시킨 단어들을 다시 하나의 문장으로 만드는 과정이다.. 이를 통해서 두 텍스트가 의미적으로 유사한지 아닌지 판단할 수 있게 해준다. 한 쌍의 두 문장을 간단히 연결한 다음 첫 번째 문장의 각 토큰에 0을 할당하는 것이고 두 번째는 각 토큰에 1을 할당하는 것이다.

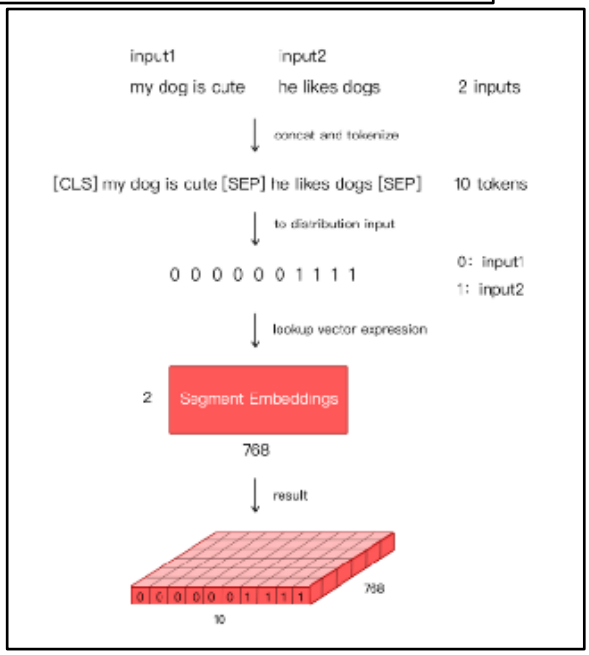
3) Position Embedding :

텍스트의 위치에 따라 의미가 달라지기도 하고 그 자체로도 의미가 달라지기도 한다. 따라서 단어 그 자체가 아닌 의미에 따라서 벡터를 생성하게 도와준다.

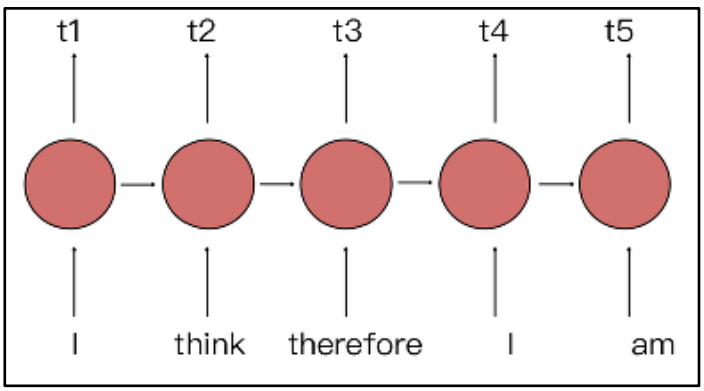




<Token Embedding>



<Segment Embedding>

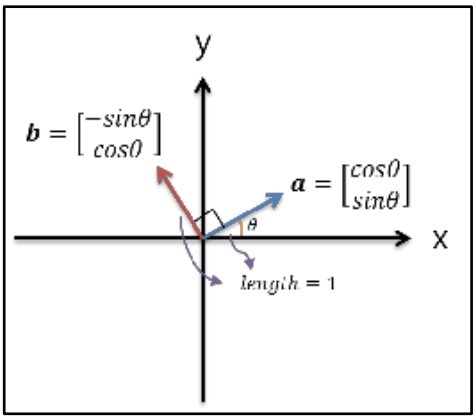


<Position Embedding>

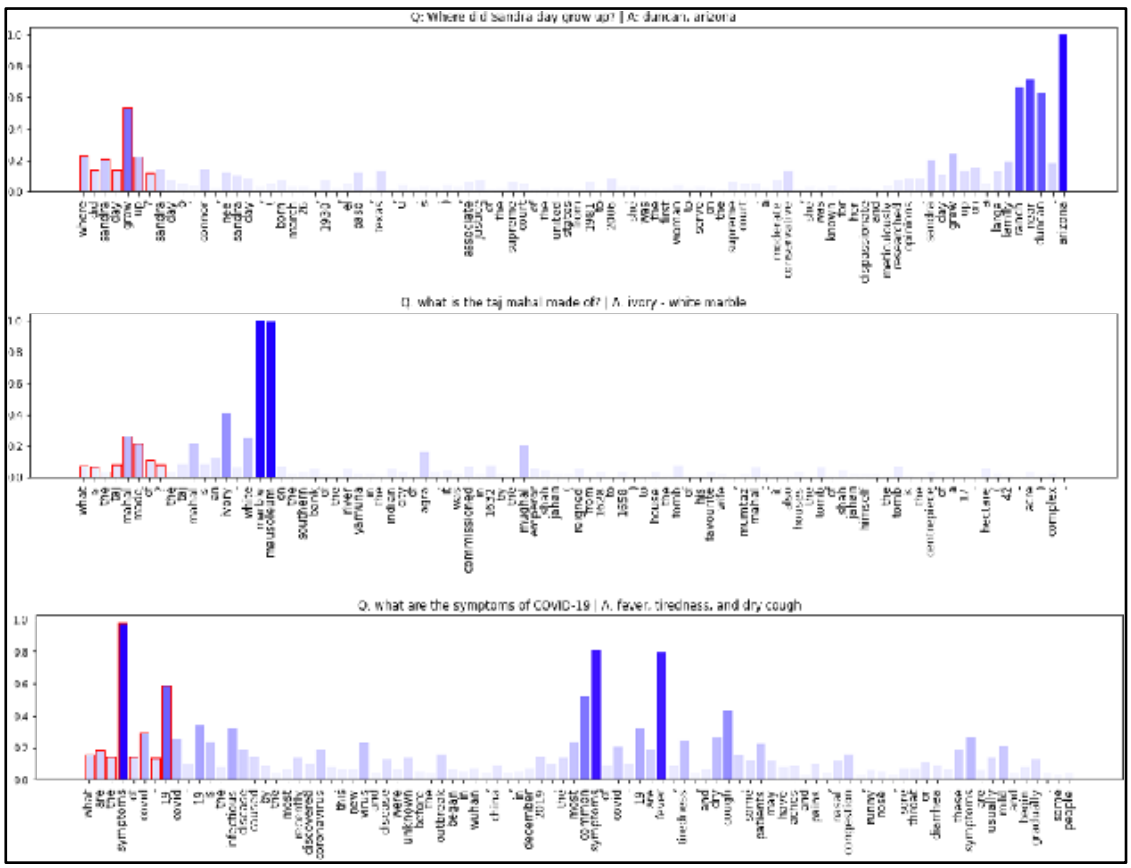
기존 모델은 query 부분, 관련 문서 부분(positive), 관련되지 않는 문서 부분(negative)을 각각 임베딩하여 훈련시킬 데이터를 모두 인코딩을 한 후, pre-training을 진행한다.

위와 같은 과정을 통해서 얻어지는 총 768차원 vector에 위에서 설명한 BM25에 사용한 Topic의 768

차원을 합쳐 새로운 vector를 만든다. 그 과정에서 orthogonal loss을 이용하여 Bert만을 사용했을 때 vector와 Topic에서 얻은 새로운 vector에 모두 적합할 수 있도록 학습을 시킨다.



최종 결과물은 아래의 형태와 같게 각각의 단어들에 대한 값으로 산출되어지게 된다. 즉 관련도가 높은 단어일수록 주위 단어들에 비해서 높은 값을 갖게 된다. 문장을 뽑아낼 때에는 관련도가 높은 시작점부터 다시 낮아지는 끝나는 점까지 설정해주면 된다.



2. 기존 연구 문제점

① BM25 + Topic

기존의 BM25 모델로 토픽을 학습했을 때, 토픽이 치우쳐지는 현상을 발견하였다. 예를 들면, 기존의 모델은 '3'이 426개, 'two'가 311, 'is'가 227개의 토픽을 가지고 있다. 즉 많이 사용되는 단어들이 토픽으로 뽑히는 경향이 있었다. 상대적으로 희귀한 단어들은 아예 토픽이 없는 경우가 대다수였다. 따라서 토픽을 이용해서 데이터를 변형시킬 때, 토픽이 주는 정보가 거의 없을 수가 있다. 기존 BM25모델은 토픽과 키워드를 적절히 사용해서 매칭을 시키는 것이 목표였지만, 사실은 키워드만을 가지고 매칭을 시키고 있을 수가 있었다. 또 다른 문제로는 토픽을 단순히 문서 뒤에 더하기만 한다는 것이다. 각 단어가 나타내는 특성이 다를 수 있는데, 이를 단순히 더하는 것은 특성을 다 활용하지 않는 것이라고 결론을 내렸다.

② Bert + Topic

기존 Bert 모델은 확실히 BM25 모델보다는 월등한 성능을 보여주었다.

	기존 BM25 모델	기존 Bert 모델
Score (R@1000)	94.20	99.97

[R@1000(recall@1000, 재현률) : 관련순으로 1000개의 문서를 ranking을 매겼을 시 실제 관련 문서 중 1000등 안에 들어간 비율]

그러나 1000등 안에 들어가는 것을 가지고 평가 지표를 삼았기 때문에 99.97 수치이지만 성능 향상의 여지는 충분히 보였다.

	R@1000	R@500	R@100
Score	99.97	99.70	94.39

실제로 평가 ranking 범위를 500, 100까지 줄이니 94.39까지 떨어졌다.

그리고 1000개의 문서 안에서 답의 rank를 세부적으로 보았을 때 BM25의 답 순위가 우위를 보이는 경우가 확인됐다. Bert가 문맥 상의 자연어 처리 성능에 대해 뛰어나지만 키워드 매칭에선 약간 부족한 성능을 보인다고는 이미 알려져 있고 실제로 위 Bert모델의 경우를 분석한 결과 키워드적으로 BM25에 비해 부족한 부분이 있어 일부 성능에서 낮은 순위를 보인다고 예측할 수 있었다.

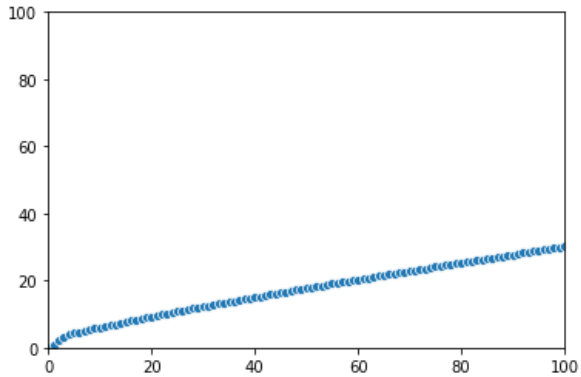
3. 연구 방법

※ 개발 환경 : 구글 colab, jupyter notebook

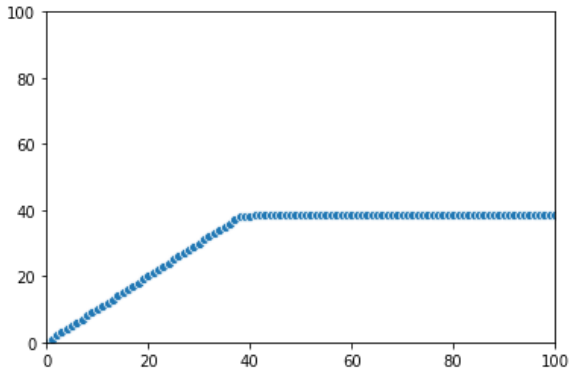
① BM25 + Topic

기존 BM25모델의 토픽 치우침을 개선하기 위해서 topk의 수를 높이는 방식을 도입했다. 하지만 단순히 topk를 높이게 되면 발생하는 문제가 크게 두가지가 있다. 첫번째로는 topk가 높아지면 토픽을

더할 때 데이터의 크기가 너무 커지게 된다. 키워드보다 토픽의 길이가 더 길어져 토픽으로만 문서매칭을 할 수가 있다. 또 다른 문제로는 희귀한 단어들이 토픽을 가질 수 있게도 되지만 기존에 토픽을 많이 가지고 있던 단어들은 심하면 768개의 토픽을 모두 가질 수 있다. 따라서 토픽의 개념이 무의미해진다. 이를 해결하기 위해서 topk의 크기를 늘이지만 각 단어가 가지고 있는 토픽수를 임의로 줄이는 방식을 사용했다. 예를 들어 'is'단어가 700개의 토픽과 매칭이 된다면 700개를 다 사용하지 않고, 700/(log(700)개만 사용한다. 또한 만약 'water'가 5개의 토픽과 매칭이 된다면 5개를 다 사용한다. 이를 위해서 첫번째로 사용한 함수는 로그함수이고, $min(x, x/\log_4 x)$개 만큼 사용해서 토픽이 많으면 줄이고 적으면 그대로 사용한다. 두번째로 사용한 함수는 시그모이드 함수이고, $min(x, sigmoid(x, m))$개 만큼 사용해서 토픽이 많으면 토픽의 개수를 줄인다. 밑의 그래프는 토픽수가 늘어남에 따라서 각 함수가 얼만큼의 토픽 개수를 사용하는지 나타내준다.



- 로그함수



- 시그모이드 함수

토픽을 그대로 더하는 것을 해결하기 위해서 다음과 같은 가정을 하였다. 토픽을 적게 포함하고있는 단어는 그 토픽에 가중치를 많이 주어 토픽을 뚜렷하게 해야하고, 토픽을 많이 포함하고 있는 단어는 토픽보다 키워드에 가중치를 주어서 같은 토픽을 많이 가지고 있는 문서보다 키워드에 더 가중치를 주어야 매칭이 더 잘 될 것이라는 가정이다. 따라서 각 단어가 포함하고 있는 토픽의 수에 따라서 토픽수가 적으면 토픽을 두번 더하고 토픽수가 많으면 키워드를 두번 더하는 방식을 도입하였다. 토픽이 적고 많음의 기준으로 다음의 두가지 방식을 사용하였다. 첫번째는 임의로 토픽수가 3,20을 기준으로 나누는 방식, 두번째는 Quantile을 사용해서 Q1,Q3를 기준으로 나누는 방식이다.

그리고 마지막으로 토픽이 너무 많이 연관되어 있는 경우는 제외하는 방식도 도입하였다. 이를 도입한 이유는 "토픽이 많이 연결되어 있는 단어는 중요하지 않을 것이다"라는 가정을 했기 때문이다.

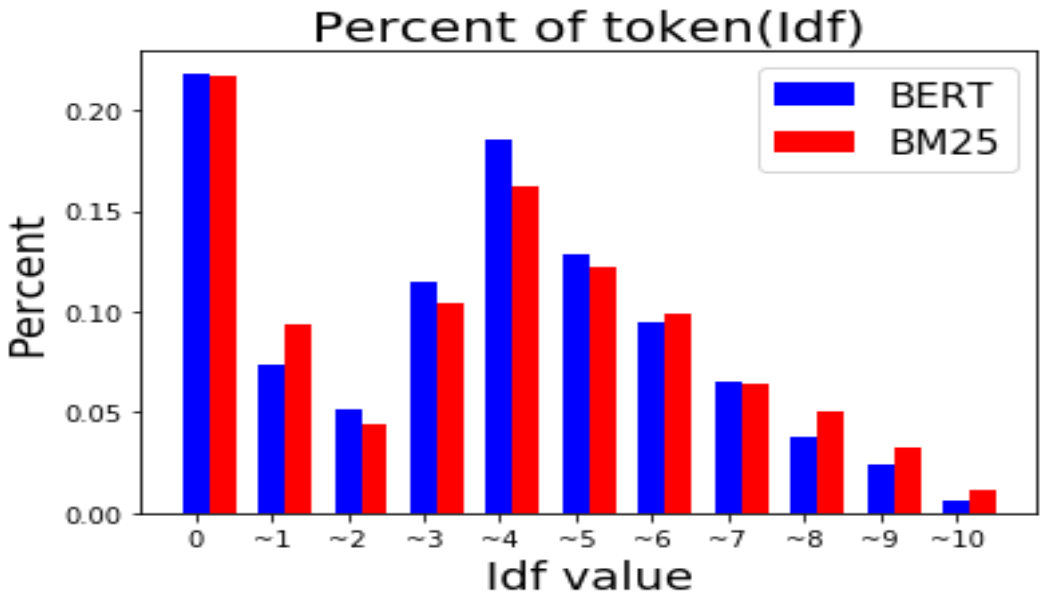
② Bert + Topic

BERT의 성능이 Bm25보다 월등히 높았기 때문에 저희는 BERT를 베이스로 두고 BM25의 장점을 넣어보기로 했다. BERT와 BM25모델의 차이를 내게 하는 요인을 찾기 위해서 첫번째로 토큰에 주목해보았다. 토큰의 개수는 총 3만개인데 이는 모델의 인풋이 3만 차원이기 때문이다. 따라서 실제로 데이

터셋에는 3만개 이상의 단어들이 있다. 빈도수로 정렬했을 때 3만번째 이후부터는 토큰으로 사용하지 않는다. 따라서 이 사용되지 않는 단어들을 사용해보기로 했다. 이렇게 사용하게 된 이유는 사라지는 단어들 중에서 빈도수는 적지만 매칭을 해주는데 있어서 키워드가 될 가능성이 있다고 생각했기 때문이다. 실제로 분석한 결과 토큰화 과정에서 사라지는 단어가 질문에 존재하는 경우가 633개의 질문중에서 92개였다. 따라서 이러한 질문들을 대상으로 사라진 단어가 포함된 문서의 이 순위를 올려줌으로써 개선을 하였다.

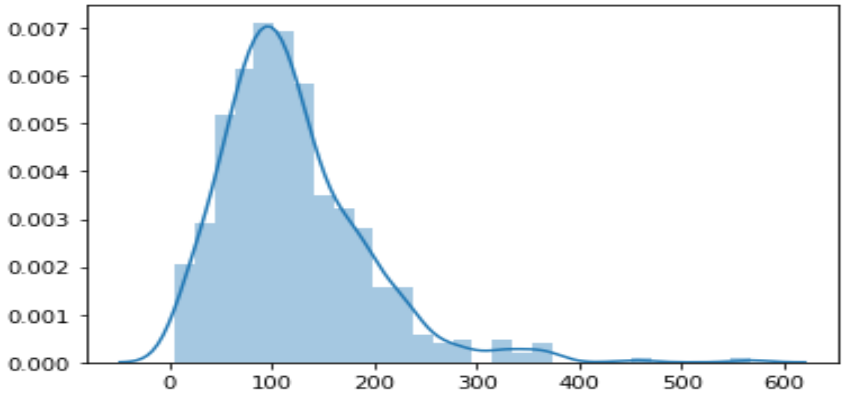
그리고 이 외에 BM25와 BERT의 순위 차이를 주는 영향을 찾아보기로 했다. 이 차이에 영향을 주는 요인으로 idf값과 문서의 길이를 생각해보았다. 첫번째로 idf값의 경우를 분석한 결과, idf값이 중간인 토큰들이 질문에 있는 경우에는 BERT가 우세하고 idf값이 높은 단어들이 질문에 있을 때는 BM25의 성능이 더 우세하였다. 두번째로는 문서의 길이를 생각해보았다. 이를 분석해 본 결과, 문서의 길이가 짧을수록 BERT의 성능이 우세한 것을 확인할 수 있었고, 반대로 문서의 길이가 길수록 BM25의 성능이 우세한 것을 확인할 수 있었다.

1. Idf값을 바탕으로 총 10개의 범위로 나눈 후 해당 범위에 속한 비율을 그래프로 표현

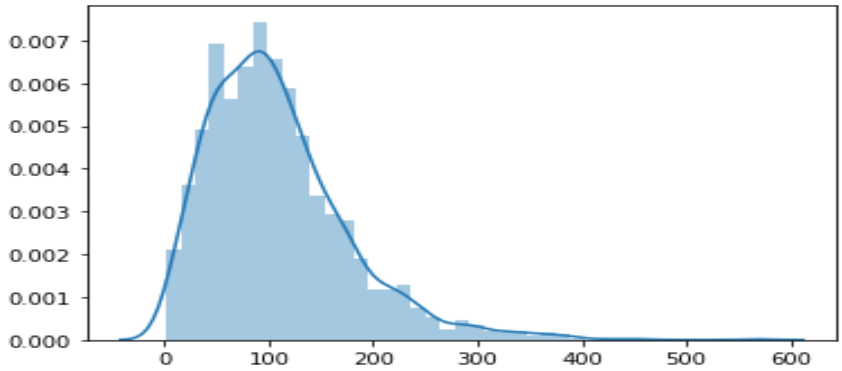


2. 문서길이를 기준 결과가 더 좋았던 질문들에 대한 정답 비율로 분석한 자료 (bm25, bert 따로)

<bm25>

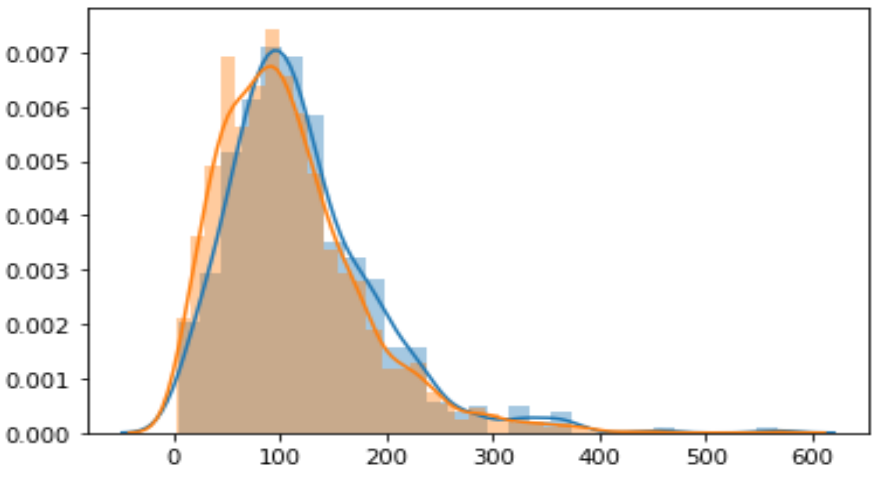


<bert>



3. 문서길이를 기준 결과가 더 좋았던 질문들에 대한 정답 비율로 분석한 자료 (bm25, bert 하나로)

<Bert(주)> <BM25(파)>



하지만 idf 값은 사용하지 않기로 하였는데, 그 이유는 다음과 같다. 지금 사용하고 있는 모델의 목적은 질문이 들어왔을 때 그에 맞는 문서를 매칭시켜주는 역할이다. 질문 데이터셋은 새로운 질문이 자주 들어올 수 있는 성격을 지니고 있는데 이는 질문 데이터셋은 계속 변화할 수 있음을 의미한다. 그런데 여기에 idf값을 주로 사용하게 된다면, idf값이 자주 변화할 것이고 이는 문제가 될 수 있다. 따라서 변화하지 않는 문서의 길이에 비해서 idf값들의 일관성이 떨어진다. 또한 idf값을 분석해 보았을 때 0~1사이의 값들이 차지하는 비율이 너무 높아서 기준으로 잡기가 애매한 이유도 있다.

4. 결과

① BM25 + Topic

기존 BM25에 새로운 알고리즘을 도입했더니 다음과 같은 결과가 확인되었다.

각 모델은 3가지 방식으로 표현된다. 첫 번째로, 토픽의 수를 줄이는 데 로그함수를 사용했는지, 시그모이드를 사용했는지, 두번째로 토픽의 수가 너무 많은 경우 그 단어를 제외를 했는지 안했는지, 마지막으로 기준을 나누는데 있어서 3,20으로 Rough하게 기준을 나누었는지 아니면 Quantile을 사용해서 기준을 나누었는 지이다.

	R@1000	R@500	R@200	R@100	R@30
BaseLine	92.99	84.62	78.89	75.52	69.57
중간발표 [illegible]함수 제외x Rough	**94.78**	91.08	84.86	79.21	68.98
시그모이드함수	92.97	87.35	76.18	66.01	53.03
로그함수 제외o Quantile	92.59	89.36	84.73	78.36	61.04
로그함수 제외x Quantile	94.75	**91.81**	87.15	82.26	73.37
로그함수 제외o rough	92.59	89.10	**87.30**	**83.60**	**74.57**

기존에는 R@1000만 비교를 했었는데, topk수를 줄여서 R@30까지 비교를 해보았다. 결과는 R@1000일 때는 중간발표의 모델이 제일 좋았지만, topk수를 줄임에 따라 맨 마지막 모델이 제일 성능이 뛰어났다.

이를 통한 최종 결론은 다음과 같다.

1) 시그모이드 함수보다 로그 함수가 더 뛰어난 성능을 보인다.

2) Rough하게 하는 것이 Quantile보다 더 성능이 뛰어나다. 즉, 학습할 때 마다 스스로 기준을 학습하

는 것보다 미리 정해주는 것이 더 효율적이다.

3) 랭킹 수를 줄일수록 토픽이 많은 단어들을 제외하고 매칭하는 것이 성능이 더 뛰어나다. 즉, 가정이 어느정도 맞다고 결론을 내린다.

② Bert + Topic

BERT모델을 개선한 결과는 다음과 같다.

	R@1000	R@500	R@200	R@100	R@30
Base	99.97	99.70	97.52	94.39	86.12
Base + Token 이용	99.97	99.70	97.60	94.50	86.48
Base + BM25 (by ?이)	99.97	99.70	97.65	94.51	86.15
Base + Token +BM25 (by 신이)	99.97	99.70	97.73	94.53	86.43
Base +BM25 + Token (by ?이)	99.97	99.70	97.73	94.61	86.51

위에서부터 모델에 대해서 설명을 하면, 첫번째는 Base모델이다. 두번째는 베이스모델에 토큰화에서 제거된 단어들을 통해서 순위를 조정한 모델이다. 세번째는 베이스모델에 BM25에서 나온 결과를 문서 길이 기준으로 앙상블해서 만든 결과이다. 세번째와 네번째는 세가지를 모두 합친 모델인데, 토큰화에서 제거된 단어들을 통해서 먼저 순위조정을 하고 다음에 BM25와의 앙상블을 통해서 나온 모델이 세번째 모델이고 다음은 순서를 바꿔서 앙상블을 먼저 진행하고 순위조정을 한 모델이다. 베이스모델이 토큰화에서 제거된 단어들을 통해서 순위를 조정한 결과에 BM25와 앙상블을 한 결과가 미세하지만 우세한 성능을 보이는 것을 알 수 있다.

이를 통한 최종 결론은 다음과 같다.

1) BERT가 BM25에 비해 월등한 결과를 보이기 때문에 BRET의 순위를 기준으로 설정한다.

2) BM25와 랭킹을 합치는 과정에서 문서의 길이를 기준으로 문서의 길이가 길수록 BM25의 비중을 크게 설정한다.

3) 정확한 키워드 매칭에 약한 모습을 보이는 BERT에 특징을 보완하고자 토큰화가 안된 특정 단어를 포함할 시 순위를 상승시킨다.

5. 문제점

현재 이 모델의 경우에는 Natural Question Dataset에서만 학습시키고 모델을 평가하였다. 따라서 다른 데이터셋을 대상으로 모델을 평가를 한다면 성능이 떨어질 가능성이 존재한다.

6. 팀원 구성 및 역할

김정학(조장) : bm25 성능 향상 알고리즘 구상 및 코드 구현

박성현(조원) : bm25, bert 모델 결과 데이터 정제 및 분석

유현석(조원) : bert 모델 결과 정제, 데이터 세부 분석 및 결과 시각화

7. 참고 자료(그림 출처)

https://developpaper.com/interpretation-of-the-paper-the-theory-of-bert/

https://sacko.tistory.com/4

https://towardsdatascience.com/explainbertqa-687abe3b2fcc