Chapter_07_요약_question-answering_v2.ipynb

transformer · 2024. 8. 16. 15:51

sequence classification
text classification

[0.8, 0.2] -> 확률분포
[31.2, 21.7] ->logit 하나의 위치(CLS) token에서만 logit vector를 뽑음
head
[representation vector, context vector]
[ ] [ ] [ ] [ ] [ ]
1 2 3 4 5
[CLS] i love this soup .....

representation vector : 문장의 의미를 전부담고있는 vector
context vector :문장의 의미를 전부담고있는 vector
[CLS] token을 맨앞에 두면 일관성있게 작업가능. CLS token 위치에 있는 vecor가 문장의 의미를 담고있음.

NER
token classification

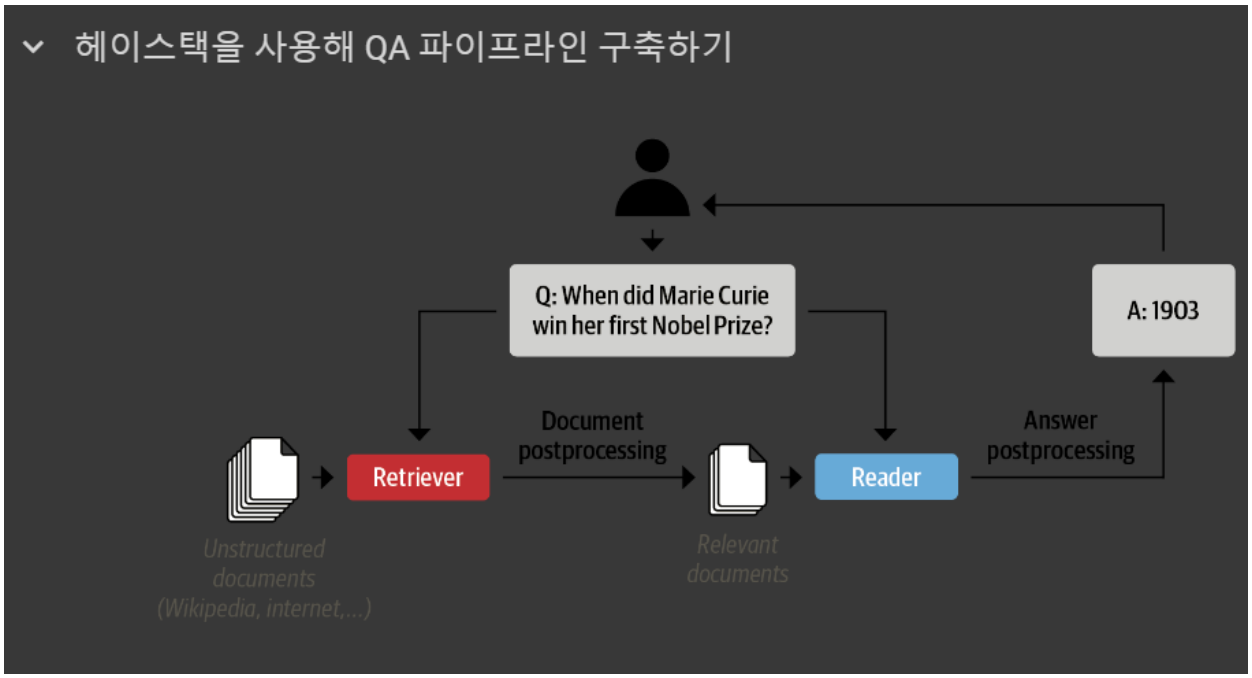
[logits] [logits] [logits] [logits] [logits]
head head head head -> 각단어에 head붙여서 각위치에서 logit vector를 뽑음

[ ] [ ] [ ] [ ] [ ]
1 2 3 4 5
[CLS] Pack love this store .....

QA (주어진 텍스트에서 찾을때만)

start and end logit -> 이렇게 해서 값이 2개씩나옴 -> 답변이 어디있는지 시작, 끝
[v1,v2] [v1,v2] [v1,v2] [v1,v2] [v1,v2] => [v1, v1, v1, v1, v1], [v2, v2, v2, v2, v2]
head head head head -> 5개모아서 이중 값이 큰게 시작위치, 뒤쪽백터는 가장큰값이 끝위치
[ ] [ ] [ ] [ ] [ ]
1 2 3 4 5
[CLS] Pack love this store .....

QA 파이프라인
해이스택은 QA파이프라인을 구축하기 편리하게 해주는 패키지
Retriever는 많은 문서중 관련 문서를 찾아서 reader(자연어모델)가 답을 찾아서 사용자에게 돌려준다.



(아래그림) 문서단어행렬
전체vocab을 열로 취하고, 문서를 row로 취해서 , 각 문서에 나온 단어갯수를 counting.

1. 문서 단어 행렬(Document-Term Matrix, DTM)의 표기법

문서 단어 행렬(Document-Term Matrix, DTM)이란 다수의 문서에서 등장하는 각 단어들의 빈도를 행렬로 표현한 것을 말
다. 쉽게 생각하면 각 문서에 대한 BoW를 하나의 행렬로 만든 것으로 생각할 수 있으며, BoW와 다른 표현 방법이 아니라 B
표현을 다수의 문서에 대해서 행렬로 표현하고 부르는 용어입니다. 예를 들어서 이렇게 4개의 문서가 있다고 합시다.

문서1 : 먹고 싶은 사과
문서2 : 먹고 싶은 바나나
문서3 : 길고 노란 바나나 바나나
문서4 : 저는 과일이 좋아요

띄어쓰기 단위 토큰화를 수행한다고 가정하고, 문서 단어 행렬로 표현하면 다음과 같습니다.

	과일이	길고	노란	먹고	바나나	사과	싶은	저는	좋아요
문서1	0	0	0	1	0	1	1	0	0
문서2	0	0	0	1	1	0	1	0	0
문서3	0	1	1	0	2	0	0	0	0
문서4	1	0	0	0	0	0	0	1	1

(아래그림)
tf 는 위 문서단어 행렬에서 나온 특정단어 t의 횟수
idf 는 df의 역수(반비례)
df가(특정단어)가 0일수도 있어서 분무가 0이 되면 문제되서 1을 더해준것
예를들어 the라는 특정단어인경우,
df(t) 가 n에 가까워지며서,
log(n/1+dft)는 log 1이라서
0값이 된다.

(1) tf(d,t) : 특정 문서 d에서의 특정 단어 t의 등장 횟수.

생소한 글자때문에 어려워보일 수 있지만, 잘 생각해보면 TF는 이미 앞에서 구한 적이 있습니다. TF는 앞에서 배운 DTM의 예제에서 각 단어들이 가진 값들입니다. DTM이 각 문서에서의 각 단어의 등장 빈도를 나타내는 값이었기 때문입니다.

(2) df(t) : 특정 단어 t가 등장한 문서의 수.

여기서 특정 단어가 각 문서, 또는 문서들에서 몇 번 등장했는지는 관심가지지 않으며 오직 특정 단어 t가 등장한 문서의 수에만 관심을 가집니다. 앞서 배운 DTM에서 바나나는 문서2와 문서3에서 등장했습니다. 이 경우, 바나나의 df는 2입니다. 문서3에서 바나나가 두 번 등장했지만, 그것은 중요한 게 아닙니다. 심지어 바나나란 단어가 문서2에서 100번 등장했고, 문서3에서 200번 등장했다고 하더라도 바나나의 df는 2가 됩니다.

(3) idf(t) : df(t)에 반비례하는 수.

$$idf(t) = log(\frac{n}{1+df(t)})$$

df(t) 가 n에 가까워지며서,
log(n/1+dft)는 log 1이라서
0값이 된다.

내 컴퓨터를 guys

(1) tf(d,t) : 특정 문서 d에서의 특정 단어 t의 등장 횟수.

생소한 글자때문에 어려워보일 수 있지만, 잘 생각해보면 TF는 이미 앞에서 구한 적이 있습니다. TF는 앞에서 배운 DTM의 예제에서 각 단어들이 가진 값들입니다. DTM이 각 문서에서의 각 단어의 등장 빈도를 나타내는 값이었기 때문입니다.

(2) df(t) : 특정 단어 t가 등장한 문서의 수.

여기서 특정 단어가 각 문서, 또는 문서들에서 몇 번 등장했는지는 관심가지지 않으며 오직 특정 단어 t가 등장한 문서의 수에만 관심을 가집니다. 앞서 배운 DTM에서 바나나는 문서2와 문서3에서 등장했습니다. 이 경우, 바나나의 df는 2입니다. 문서3에서 바나나가 두 번 등장했지만, 그것은 중요한 게 아닙니다. 심지어 바나나란 단어가 문서2에서 100번 등장했고, 문서3에서 200번 등장했다고 하더라도 바나나의 df는 2가 됩니다.

(3) idf(t) : df(t)에 반비례하는 수.

$$idf(t) = log(\frac{n}{1+df(t)})$$

바나나	ln(4/(2+1)) = 0.287682
사과	ln(4/(1+1)) = 0.693147
싶은	ln(4/(2+1)) = 0.287682
저는	ln(4/(1+1)) = 0.693147
좋아요	ln(4/(1+1)) = 0.693147

문서3에서 길고노란 바나나 바나나
의 바나나가 많이나온것에 tf=2이고,
바나나가 전체문서중 2개문서에서 나와서
idf 가 가중치를 곱해서

문서의 총 수는 4이기 때문에 ln 안에서 분자는 늘 4으로 동일합니다. 분모의 경우에는 각 단어가 등장한 문서의 수(DF)를 의미하는데, 예를 들어서 '먹고'의 경우에는 총 2개의 문서(문서1, 문서2)에 등장했기 때문에 2라는 값을 가집니다. 각 단어에 대해서 IDF의 값을 비교해보면 문서 1개에만 등장한 단어와 문서2개에만 등장한 단어는 값의 차이를 보입니다. IDF는 여러 문서에서 등장한 단어의 가중치를 낮추는 역할을 하기 때문입니다.

TF-IDF를 계산해보겠습니다. 각 단어의 TF는 DTM에서의 각 단어의 값과 같으므로, 앞서 사용한 DTM에서 단어 별로 위의 IDF값을 곱해주면 TF-IDF 값을 얻습니다.

	과일이	길고	노란	먹고	바나나	사과	싶은	저는	좋아요
문서1	0	0	0	0.287682	0	0.693147	0.287682	0	0
문서2	0	0	0	0.287682	0.287682	0	0.287682	0	0
문서3	0	0.693147	0.693147	0	0.575364	0	0	0	0
문서	0.693147	0	0	0	0	0	0	0.693147	0.693147

아래그림)

출처:https://wikidocs.net/31698

04-04 TF-IDF(Term Frequency-I...

이번에는 DTM 내에 있는 각 단어에 대한 중요도를 계산할 수 있는 TF-IDF 가중치에 대해서 알아보겠습니다. TF-IDF를 ...

wikidocs.net

문서 하나의 vocab 에 있는 단어를 몇십에서 몇백개밖에 쓰지 않아서 (TF, IDF 행렬을 쓰면 0을 갖는 단어는 항상0이라서 곱해서 0) sparse representation임

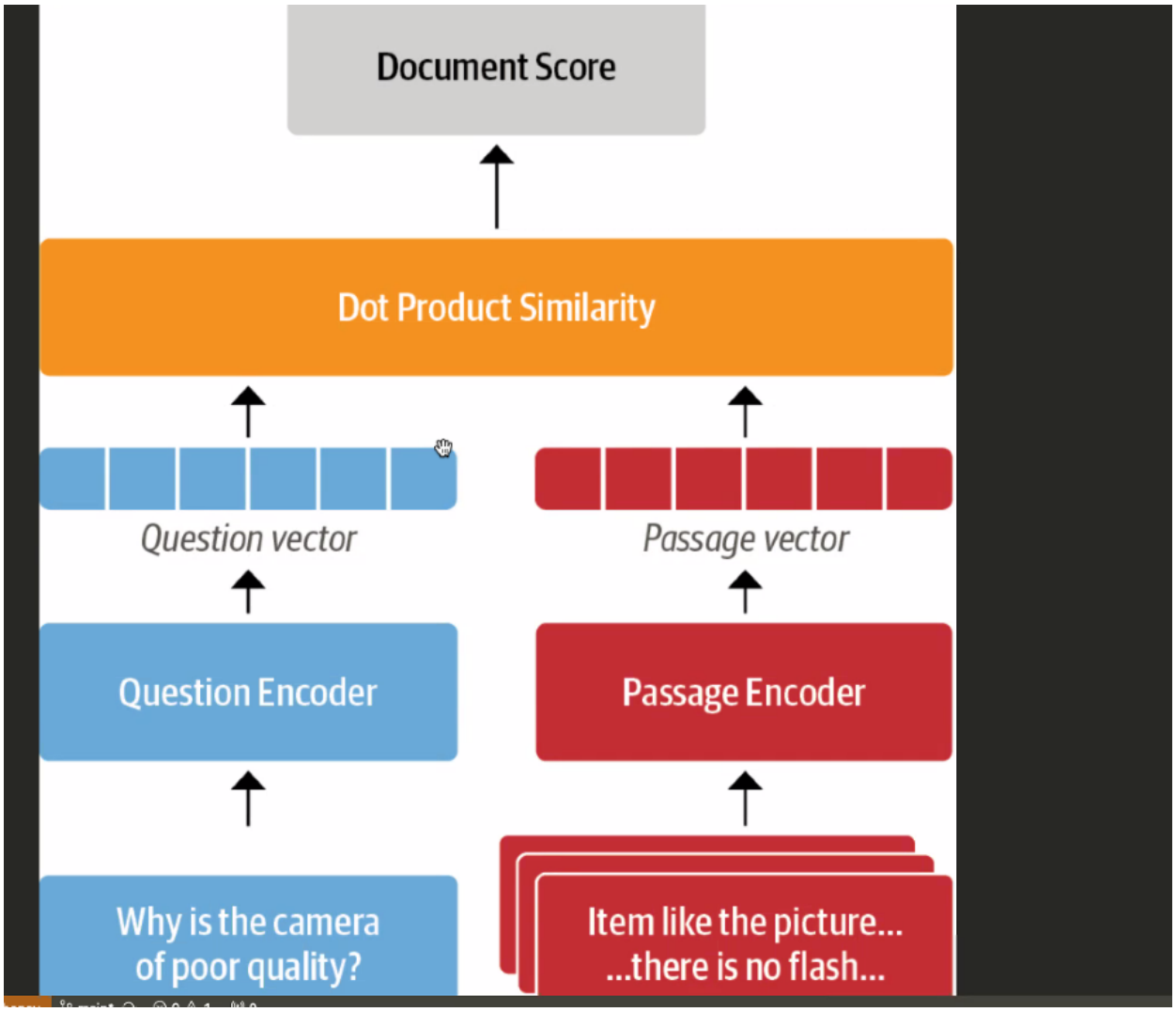
목적

문서를 vector화 하기 위한것.
vector하면 내적해서 similarity를 구할수 있기때문에 문서를 vactor화 한다.
문제를 vector 화해서 유사도를 관련문서와(retreiver) 유사도가 높은 것을 찾아내기위한것
-> Keyword (답높은 것찾는것)

을 곱해주면 TF-IDF 값을 얻습니다.

	과일이	길고	노란	먹고	바나나	사과	싶은	저는	좋아요
문서1	0	0	0	0.287682	0	0.693147	0.287682	0	0
문서2	0	0	0	0.287682	0.287682	0	0.287682	0	0
문서3	0	0.693147	0.693147	0	0.575364	0	0	0	0
문서4	0.693147	0	0	0	0	0	0	0.693147	0.693147

아래그림)
Dense retreiver
sparse retreiver는 자원이 많이드는 반면,
dense retreiver는 반대임. 효용성이 높음. (
DPR
DENSE PASSAGE RETRIEVER



DPR

DENSE PASSAGE RETRIEVER

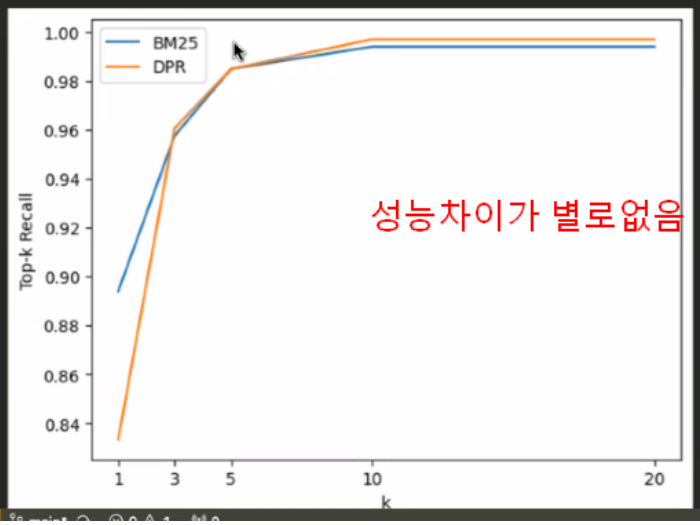
BM25(sparse retreiver)DPR 와 별차이없음

```
dpr_retriever = DensePassageRetriever(document_store=document_store,
    query_embedding_model="facebook/dpr-question_encoder-single-nq-base",
    passage_embedding_model="facebook/dpr-ctx_encoder-single-nq-base",
    embed_title=False)
```
[61] ✓ 2.6s

```
document_store.update_embeddings(retriever=dpr_retriever)
```
[62] ✓ 15.7s

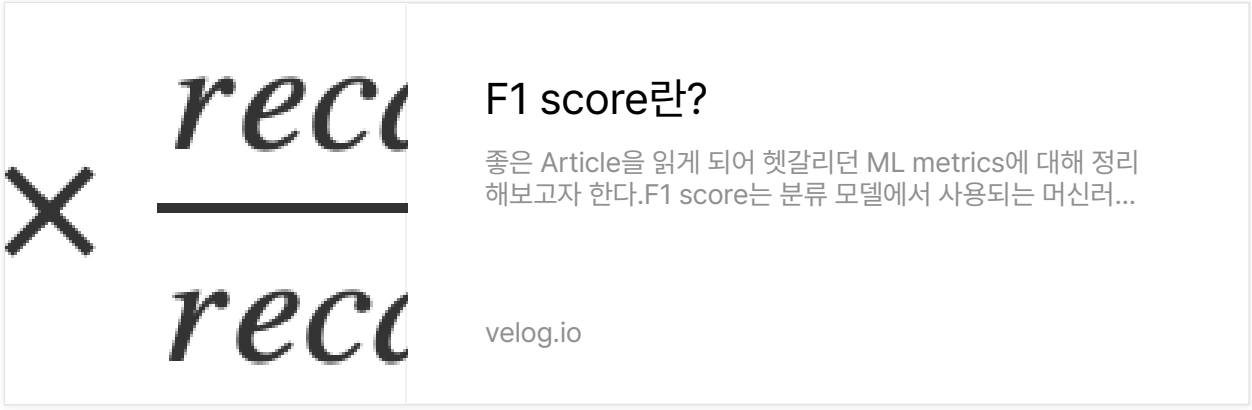
```
Updating embeddings: 10000 Docs [00:15, 638.22 Docs/s]
```

```
dpr_topk_df = evaluate_retriever(dpr_retriever)
plot_retriever_eval([bm25_topk_df, dpr_topk_df], ["BM25", "DPR"])
```
[63] ✓ 21.4s



F1 score 소개

출처:https://velog.io/@jadon/F1-score%EB%9E%80



출처: https://images.app.goo.gl/1wXmAShHXcnZP6Yz6

$$\text{F1 Score} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$Accuracy = \frac{\#\ of\ correct\ predictions}{\#\ of\ total\ predictions}$$

아래그림)

F1 Score : 성능지표 (prescision과 recall의 조화평균)

pred: about 6000 hours

label: 6000 hours

precision = 모델이 예측한 답 중에 맞은 것의 비율

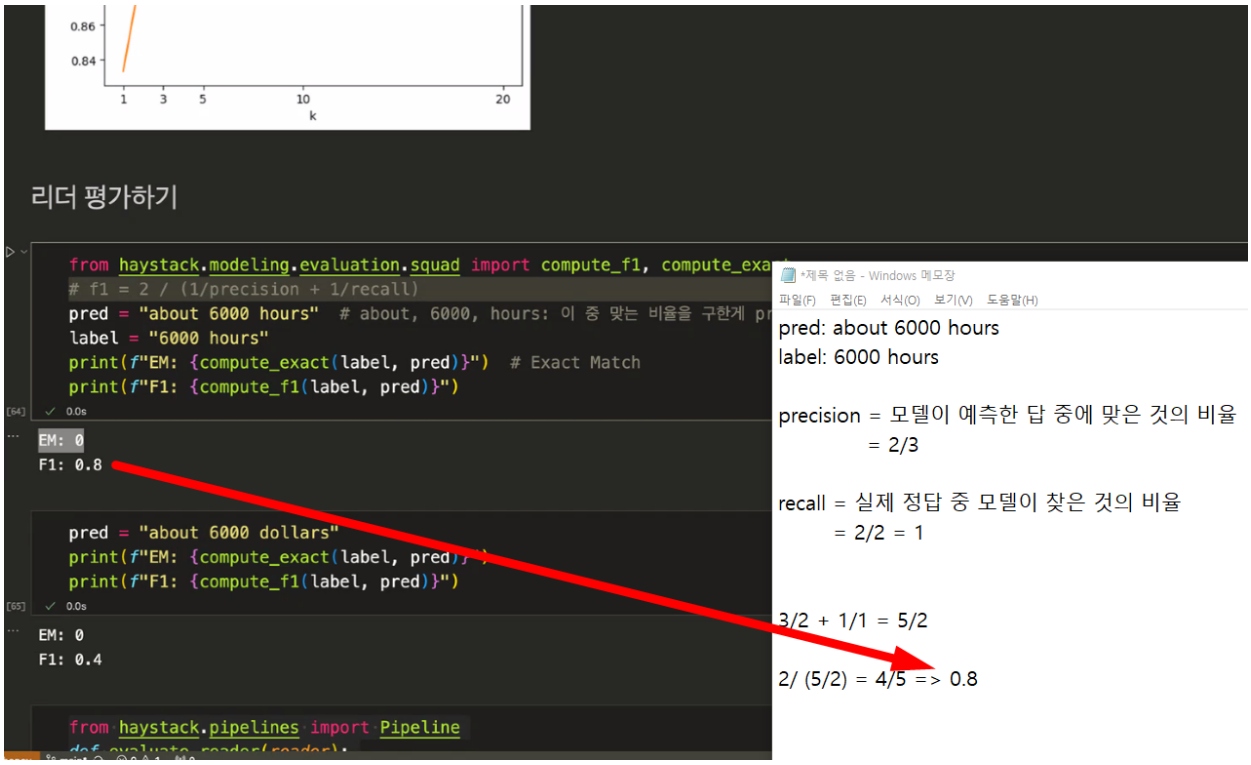
= 2/3

recall = 실제 정답 중 모델이 찾은 것의 비율

= 2/2 = 1

3/2 + 1/1 = 5/2

2/ (5/2) = 4/5 => 0.8

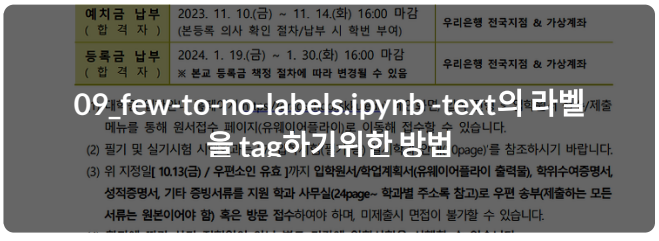


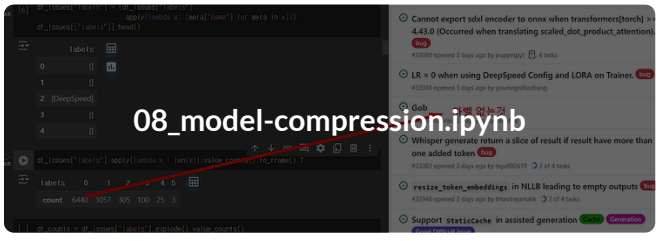
'transformer' 카테고리의 다른 글

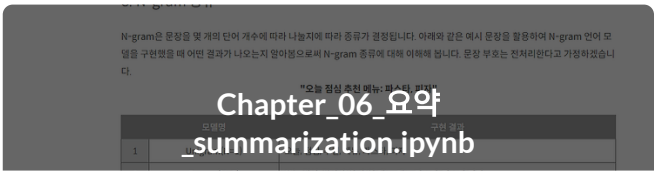
09_few-to-no-labels.ipynb -text의 라벨을 tag하기위한 방법 (0)	2024.08.30
08_model-compression.ipynb (0)	2024.08.26
Chapter_06_요약_summarization.ipynb (0)	2024.08.07
Chapter_05_요약_text-generation.ipynb (0)	2024.07.31
Chapter_04_요약_multilingual-ner.ipynb (0)	2024.07.26

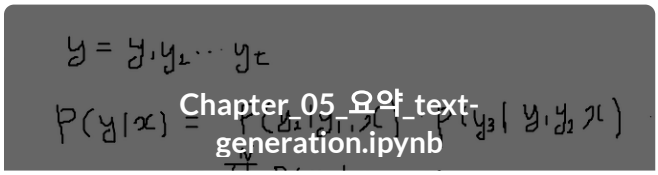
관련글

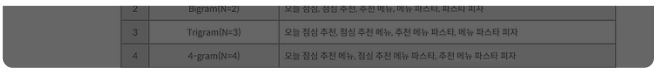
관련글 더보기



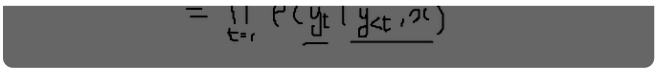








2	Bigram(N=2)	오늘 점심, 점심 추천, 추천 메뉴, 메뉴 파스타, 파스타 피자
3	Trigram(N=3)	오늘 점심 추천, 점심 추천 메뉴, 추천 메뉴 파스타, 메뉴 파스타 피자
4	4-gram(N=4)	오늘 점심 추천 메뉴, 점심 추천 메뉴 파스타, 추천 메뉴 파스타 피자



댓글 0