	1st trial	2nd trial	3rd trial
문장 임베딩 벡터 크기	5000	5000	5000
DNN 구조	5000×1024×256×256×1	5000×2048×256×64×1	5000×64×64×1
(epoch, iteration)	(30, 15만)	(30, 15만)	(30, 15만)
Average Loss	0.25	0.25	0.26
Training Time(분)	208.75	347.74	137.85
Test Accuracy(%)	83.30	83.52	84.20

코퍼스에서 출현한 단어들 중, 출현 빈도가 높은 단어 5000개를 선택한다.
그리고 한 문장을 5000개 단어를 이용해 Bag of Word 형태로 임베딩처리한다.
예를 들어, it is a cat이란 문장이 있을 때, 'it', 'is', 'a' 'cat'으로 문장을 토큰화한 뒤,
각 단어의 빈도수를 구한 다음, 문장 임베딩 벡터에서 해당 단어의 인덱스 번호에 해당하는 부분을 그 단어의 빈도수로 정한다.
따라서, it이 0번 index, is가 2번 index, a가 3번 index, cat이 5번 index라면, 위 문장은 [1, 0, 1, 1, 0, 1, ...] 이렇게 될 것이다. 이렇게 생성한 문장 임베딩 벡터들을 DNN에 통과시켜, 긍정 문장인지, 부정 문장인지 분류한다.

학습 데이터는 평점이 이진 분류로 처리되어 있지 않기 때문에, 평점이 5 이상인 데이터들을 전부 긍정으로 분류하였으며, 평점이 5보다 작은 데이터들은 전부 부정으로 분류했다. 그리고 DNN은 이진 분류를 위해 Binary Cross Entropy 방식을 사용해서 Loss를 사용했다.

3가지 DNN 모델을 사용했다. 1st , 2nd 모델은 3rd 모델보다 더 복잡하고 더 깊으므로 학습 시간이 더 길었다. 그러나 1st , 2nd , 3rd 모델의 Average Loss는 거의 유사했다.
또한 Test Accuracy를 비교한 결과 오히려 가벼운 모델의 3rd 모델이 좀 더 Accuracy가 높게 나온 것을 확인할 수 있었다.

결론적으로, DNN 모델을 이용하여 문장 분류 시스템이 준수하게 동작하는 것을 확인할 수 있었다. DNN을 통한 문장 분류는 단어의 조합과 출현 빈도가 문장을 분류하는데 사용될 수 있는지 확인하고자 하는 것이 목표였다.
예를 들어, 문장 안에서 '좋아'라는 단어가 여러번 등장할 경우 좀 더 긍정 문장으로 분류되게 할 수 있는지, '별로'라는 문장이 등장하면 부정 문장으로 분류되게 할 수 있는지를 확인하는 것이 목표였으며, 이번 실험을 통해서 어느 정도 가능함을 확인할 수 있었다.