**[Project 2]** **IMDb 리뷰 분류**



|  |  |
| --- | --- |
| 교과목명 | 딥러닝 및 실습 |
| 담당교수님 | 홍두화 교수님 |
| 학과 | 컴퓨터‧메카트로닉스공학부  소프트웨어전공 |
| 학번 | 2019101412 |
| 이름 | 이예찬 |
| 제출일 | 2022. 12. 12. 월요일 |

1. 실험 목표

: 수업 시간에 RNN을 학습하면서 매주 실습을 진행했지만 확실하게 구조를 이해하기에는 부족했다. 따라서 이번 실험을 진행하면서 RNN의 구조를 이해하고 각각의 층에서 어떤 역할을 하며, 하이퍼 파라미터가 성능을 높이기 위해 각각 어떤, 그리고 얼만큼의 역할을 하는지 알고자 한다. 또한 IMDb 데이터를 사용해서 구현한 모델의 성능을 실험해 보면서 IMDb 데이터가 어떻게 구성되어 있는지, IMDb 데이터의 구조에 따라 어떻게 모델을 구성해야 하는지에 대해 연구함으로써 딥러닝 구조에 대한 이해를 높이고자 한다.

1. 모델 비교 및 각 모델에 대한 시사점

* 모델 설계 초반 단계
* 단어 사이의 연관도가 중요한 리뷰 데이터이기 때문에 먼저 Embedding layer를 모든 모델에 더해줌으로서 숫자형을 벡터형으로 바꿔주고자 했다.
* 리뷰 데이터는 문맥을 보는 것도 중요하기 때문에 Bidirectional RNN을 사용해서 이전과 이후의 시점 모두를 고려해 현재 시점의 예측을 더욱 정확하게 할 수 있도록 고안했다.
* 사이의 층들을 조금씩 바꿔주면서 실험을 진행했다.
* 중요하게 생각했던 부분은 사이의 층의 구조와 optimizer, 마지막에 truncating을 바꿔서 시도해봤다.
* 이 리뷰 데이터는 영어로 구성되어 있기 때문에 truncating = post가 더 유리할 것으로 예상했지만 큰 영향을 주진 않았다.

텍스트이(가) 표시된 사진

자동 생성된 설명

1. 'rmsprop','binary\_crossentropy', batch\_size=64/ loss: 0.4433 - acc: 0.8309/ truncating = post
2. 'rmsprop','binary\_crossentropy', batch\_size=64/ / truncating = pre
3. "adam", "binary\_crossentropy", batch\_size=32/ loss: 0.6852 - acc: 0.8171/ truncating = post
4. "adam", "binary\_crossentropy", batch\_size=64/ loss: 1.1675 - acc: 0.8117/ truncating = post

* optimizer가 adam이라고 해서 높은 성능을 내는 건 아니라는 결과를 얻을 수 있음.

텍스트이(가) 표시된 사진

자동 생성된 설명

- rmsprop/ loss: 0.7046 accuracy: 0.8050/truncating = post

* LSTM을 GRU를 바꿔서 시도했다. 학습속도는 빠르지만 성능은 비슷하다고 학습했기 때문에 시도해봤으나 LSTM보다 뛰어난 성능을 보이지 않는 것으로 보아 다른 모델을 시도했다.

텍스트이(가) 표시된 사진

자동 생성된 설명

1. rmsprop/ loss: 0.7667 - accuracy: 0.8359/truncating = post
2. adam/ loss: 0.6106 - accuracy: 0.8296/truncating = post

* 학습층을 조금 줄여서 시도해 보았다. 층이 많다고 해서 좋은 것은 아니고 그만큼의 많은 학습 데이터가 필요하기 때문에 시도해 보았더니 더 좋은 성능을 얻을 수 있었다.

텍스트이(가) 표시된 사진

자동 생성된 설명

1. adam/ loss: 1.4457 - accuracy: 0.6804/ truncating = post
2. adam/ loss: 0.8966 - accuracy: 0.8164/ truncating = pre

* 이번에는 층을 많이 늘려봐서 시도를 해봤더니 예상했던 대로 loss가 크고 accuracy가 높지 않음을 알 수 있었다.
* 또한 truncating을 pre로 바꿔서 시도해봤는데 생각보다 높은 정확도를 보임을 알 수 있었다. 그렇지만 loss가 높았기 때문에 좋은 시도는 아니라고 생각한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

- adam/ loss: 0.6116 - accuracy: 0.8132/truncating = post

* 성능이 좋은 LSTM층을 추가해서 시도해보았다. 성능이 좋은 층을 추가하면 성능이 매우 높아질 것이라고 예상했지만 그렇지 않았다.

텍스트이(가) 표시된 사진

자동 생성된 설명

1. Rmsprop/loss: 0.5155 - accuracy: 0.8489/ truncating="pre"
2. Rmsprop/loss: 0.5494 - accuracy: 0.8450/ truncating="post"

* 최종적으로 좋은 모델을 사용해서 복잡하지 않은 모델을 만들고자 했고, 시도해본 결과 아주 조금 정확도가 상승했음을 볼 수 있었다. Dense층 대신에 LSTM층으로 대체했고, 다른 층도 많이 쓰지 않았다. 그렇지만 기본적으로 자연어 처리에 효율적이라고 생각했던 Embedding 층과 bidirectional 층을 유지하고 진행하게 되었다.

1. 실험 후 느낀 점

: 역시 모델을 실험하는데 저번 프로젝트와 마찬가지로 한 번 실험하는데 오래 걸리는 것이 가장 큰 문제였다. 그래도 다양한 모델을 시험하는 것에 굉장히 의미가 있었다고 생각한다. LSTM과 GRU같은 최신 모델을 실험하고 조합해 보면서 다양한 성능과 모델의 구조에 대한 이해까지 높일 수 있는 기회였다고 생각한다. 첫번째 프로젝트에서는 층이 많고 복잡할수록 높은 성능을 보이는 양상을 볼 수 있었는데 이번 프로젝트에서는 무조건 층을 많게 잡고 복잡할수록 좋은 성능을 내는 것이 아니라는 결과를 알 수 있었고, 옵티마이저도 adam이 항상 가장 좋은 성능을 내는 것이 아니라는 것을 배웠다. 모델에 따라 적절한 하이퍼 파라미터를 적용시켜야 된다는 것을 알 수 있었다.