Assignment #2

Transformer 모델을 활용한 Apple 주가 예측

소속 : 소프트웨어융합대학 소프트웨어학부

학번 : 20191610

이름 : 성정규

문제 설명

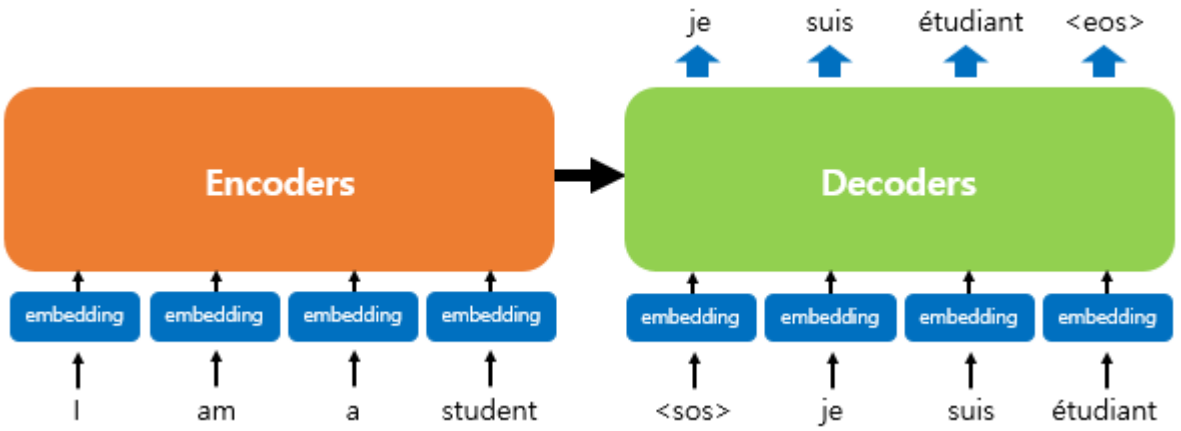
너 우크라이나 러시아가 전쟁이 일어날거라 생각해?

저는 인공지능 모델이기 때문에 개인적인 의견이나 예측을 할 수 없습니다. 전세계의 정치적, 사회적, 경제적 상황은 매우 복잡하고 예측하기 어렵습니다. 전쟁이 발생할 가능성이 있는지 여부는 현재의 정치적 상황과 국제 관계에 따라 달라지며, 이는 전문가들이 심도있게 분석해야 하는 주제입니다. 정확한 정보와 신뢰할 수 있는 소식 매체를 통해 최신 동향을 파악하시는 것이 좋습니다.

문제에 대한 접근은 인공지능 모델이 전쟁과 같은 외부 상황으로 인한 주가 하락을 예측할 수 있을까? 라는 발상에서 시작하였습니다. 2021년 말, 군 전역 이후 모았던 군 적금이 만기된 저는 모든 돈을 주식에 투자하였습니다. 허나 2022년 2월, 우크라이나와 러시아의 전쟁이 시작되며 저는 투자 수익률 -70%를 기록하며 모았던 돈의 절반 이상을 잃어버리고 말았습니다. 인공지능 모델이 외부 상황으로 인한 주가 변동에 대해 어느정도 예측할 수 있다면 이렇게 모아뒀던 돈을 다 잃는 불상사가 없지 않을까 해서 이 주제를 선정하였습니다.

저는 이 주제를 RNN(Recurrent Neural Network)과 LSTM(Long Short-Term Memory)을 사용하여 해결하려 했었습니다. 허나 학습 결과가 제가 생각했던 것보다 너무 떨어졌기 때문에 Transformer 모델을 활용하여 개선해보려 합니다. 2000년 1월 1일부터 2022년 2월 24일까지의 애플 주식 데이터를 수집하여 모델을 학습시키고, 특정 시점부터 모델이 예측하는 주가와 실제 데이터와 비교하여 올바르게 예측했는지 확인할 생각입니다. 2022년 2월 24일까지의 데이터를 사용하는 이유는 이 날 러시아가 우크라이나를 침공하며, 애플의 주식이 급락하였기 때문입니다. 모델이 특정 시점에서부터 예측을 시작하여, 2월 24일에 주가가 급락하는 것까지 예측해낸다면 외부 상황에 의한 주가 변동을 예측할 수 있다는 것이고 이는 굉장히 긍정적인 결과입니다. 결과적으로 Transformer 모델로 예측해보고, RNN과 LSTM으로 예측했을 때 보다 성능이 더 좋은지, 주가 변동을 예측할 수 있는지 확인할 예정입니다.

작동 원리



Transformer 모델은 자연어 처리와 시계열 데이터 등 다양한 분야에서 주로 사용되는 딥러닝 모델입니다. 주로 번역, 요약, 질의응답 등의 작업에 활용됩니다. Transformer 모델은 기존의 순환 신경망(RNN)이나 컨볼루션 신경망(CNN)과는 다른 구조를 가지고 있습니다. 기존의 모델은 순차적으로 데이터를 처리하는 반면, Transformer 모델은 입력 시퀀스를 한 번에 처리합니다.

Transformer 모델의 핵심 개념은 어텐션(Attention)입니다. 어텐션은 입력 시퀀스 내의 단어나 시간적인 관계에 주목하여 중요한 정보를 강조하는 메커니즘입니다. Transformer 모델은 여러 개의 어텐션 레이어를 쌓아올린 형태로 구성됩니다.

Transformer 모델의 구조는 크게 인코더(Encoder)와 디코더(Decoder)로 나뉩니다. 인코더는 입력 시퀀스의 특성을 추출하고, 디코더는 인코더에서 추출한 특성을 바탕으로 출력 시퀀스를 생성합니다.

인코더는 여러 개의 어텐션 레이어와 feed-forward neural network 레이어로 구성됩니다. 각 어텐션 레이어는 입력 시퀀스의 모든 위치에 대해 어텐션 값을 계산하고, 이를 조합하여 특성 벡터를 생성합니다. 인코더의 주요 역할은 입력 시퀀스의 단어 간 관계를 학습하여 특성 벡터를 생성하는 것입니다.

디코더는 인코더에서 생성된 특성 벡터를 사용하여 출력 시퀀스를 생성합니다. 디코더는 인코더와 유사한 구조를 가지고 있으며, Self-Attention과 Encoder-Decoder Attention 두 가지 종류의 어텐션을 사용합니다.

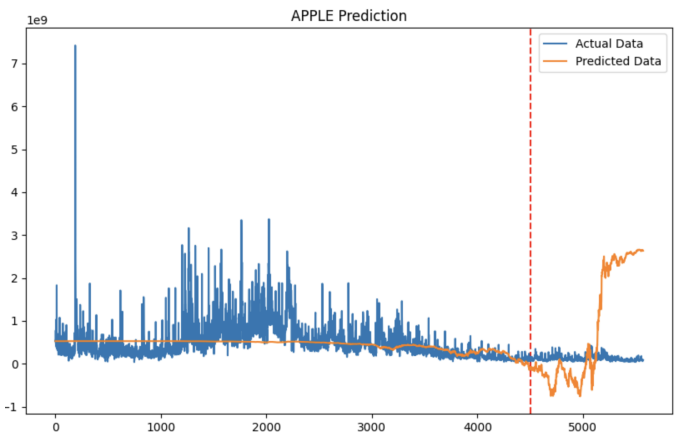
Code의 작동 원리

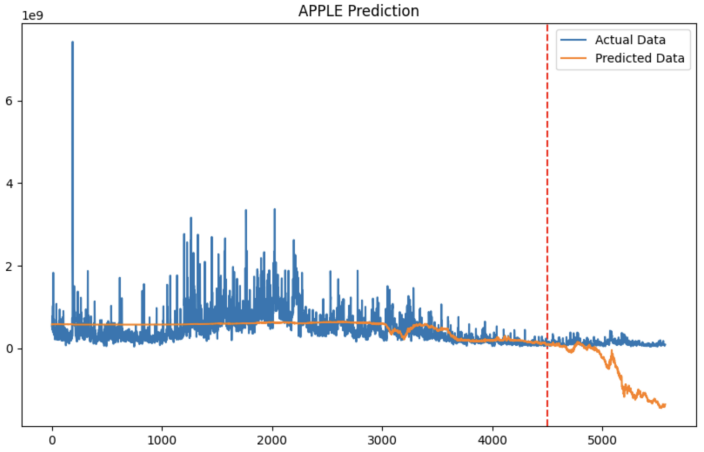
1. 라이브러리 및 모듈 임포트:
   코드의 첫 부분에서는 필요한 라이브러리 및 모듈을 임포트합니다. NumPy, Pandas, Matplotlib, datetime, yfinance 등과 같은 라이브러리를 사용하여 데이터 처리 및 시각화 작업을 수행합니다. 또한, PyTorch를 사용하여 Transformer 모델을 구현하기 위한 모듈도 임포트합니다.
2. 하이퍼파라미터 설정:
   모델의 하이퍼파라미터 값들을 설정합니다. 예를 들어, 학습 에포크 수, 학습률, 입력 크기, 은닉 크기, 레이어 수, 클래스 수 등이 여기에 해당합니다.
3. 주식 데이터 가져오기:
   yfinance 라이브러리를 사용하여 주식 데이터를 다운로드합니다. 이 코드에서는 'AAPL' (애플 주식) 데이터를 2000년 1월 1일부터 2022년 2월 24일까지 가져옵니다.
4. 입력 데이터와 출력 데이터 준비:
   다운로드한 주식 데이터에서 필요한 특성들을 선택하여 입력 데이터와 출력 데이터로 나눕니다. 이 코드에서는 'Open', 'High', 'Low', 'Close', 'Volume' 열을 입력으로 사용하고, 'Close' 열을 출력으로 사용합니다. 또한, 입력 데이터와 출력 데이터를 스케일링합니다.
5. 학습 데이터와 테스트 데이터 분할:
   전체 데이터를 학습 데이터와 테스트 데이터로 나눕니다. 이 코드에서는 전체 데이터 중 80%를 학습 데이터로 사용하고, 나머지 20%를 테스트 데이터로 사용합니다.
6. 데이터를 텐서로 변환하고 모델에 적합한 형태로 재구성:
   데이터를 PyTorch의 텐서로 변환하고 모델에 적합한 형태로 재구성합니다. 이 코드에서는 입력 데이터는 3차원 텐서로 변환하고, 출력 데이터는 2차원 텐서로 유지합니다.
7. Transformer 모델 정의:
   nn.Module을 상속받아 Transformer 모델을 정의합니다. 이 모델은 입력 데이터를 임베딩한 후 nn.Transformer를 사용하여 시계열 데이터의 시간적인 관계를 학습합니다. 마지막으로, 출력 데이터의 예측을 위해 선형 레이어를 사용합니다.
8. 손실 함수와 최적화 알고리즘 설정:

평균 제곱 오차(Mean Squared Error, MSE)를 손실 함수로 사용하고, Adam 최적화 알고리즘을 설정합니다.

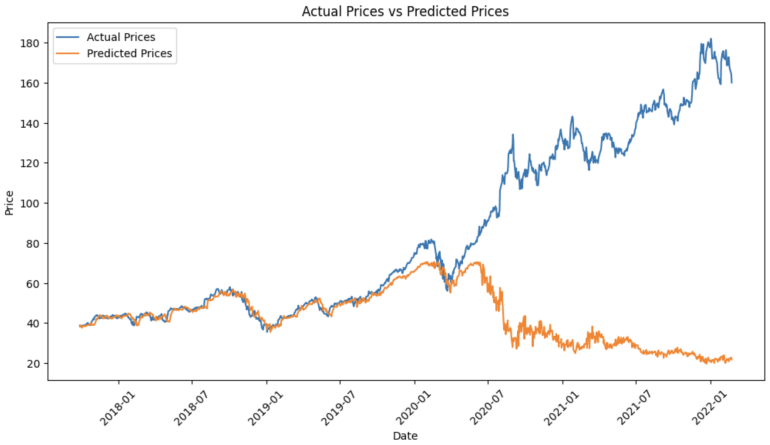
9. 학습 시작:
   주어진 에포크 수만큼 모델을 학습시킵니다. 학습 데이터를 미니배치로 나누고, 순방향 전파를 통해 예측을 계산한 후 손실을 계산합니다. 역전파와 최적화를 수행하여 모델을 업데이트합니다.
10. 모델을 평가 모드로 전환:
    학습이 완료된 후 모델을 평가 모드로 전환합니다.
11. 테스트 데이터 예측:
    테스트 데이터를 사용하여 모델의 예측을 계산합니다.
12. 예측 결과를 실제 값의 스케일로 변환:
    스케일링된 예측 결과를 원래의 데이터 스케일로 변환합니다.
13. 실제 데이터와 예측한 데이터 시각화:
    실제 주식 가격과 모델의 예측 결과를 그래프로 시각화하여 비교합니다.

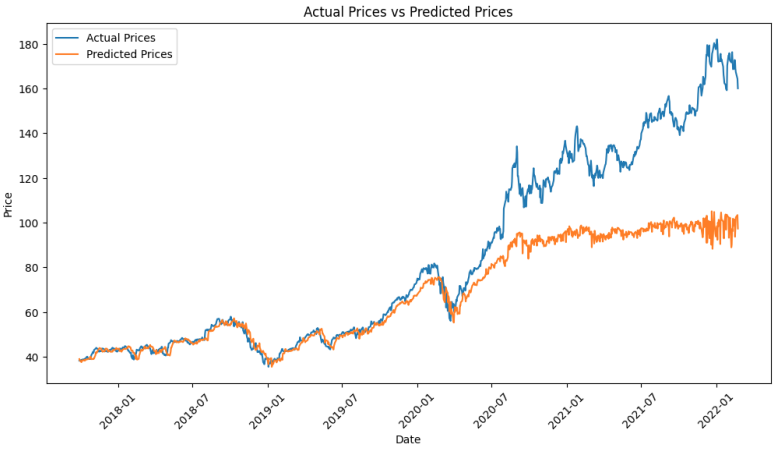
결과 분석





지난 RNN 모델과 LSTM 모델을 활용하여 주가를 분석하였을 때의 결과입니다.
좌측이 RNN 모델의 실제 데이터와 모델이 예측한 데이터를 비교하여 시각화한 자료이고, 우측이 LSTM 모델의 실제 데이터와 모델이 예측한 데이터를 비교하여 시각화한 자료입니다. 빨간 점선부터가 모델이 예측한 값인데, 그래프의 차이가 많이 날 뿐더러 데이터의 역변환도 제대로 이루어지지 않았습니다. 특정 시점부터 모델이 예측하였던 부분을 나눈 것이 문제라 생각하여 Transformer 코드를 작성할 땐 모델이 학습한 뒤 예측 결과를 한번에 그래프로 시각화하도록 코드에 변화를 주었습니다.





사진은 Transformer 모델로 학습했을 때의 결과입니다. 좌측은 Epoch을 3000번 돌린 결과이고 우측은 Epoch을 10000번 돌린 결과인데, 데이터의 역변환을 성공하여 실제 주가와 모델이 예측한 주가를 제대로 비교해 볼 수 있었을 뿐더러, Epoch을 돌릴 수록 예측 성능이 올라가는 결과를 보여주었습니다.

```
Epoch: 29800/30000, Loss: 0.00257907179184258
Epoch: 29900/30000, Loss: 0.0025646535214036703
Epoch: 30000/30000, Loss: 0.0025520636700093746
```

```
Epoch: 29800/30000, Loss: 0.0023332154378294945
Epoch: 29900/30000, Loss: 0.0023325050715357065
Epoch: 30000/30000, Loss: 0.0023317947052419186
```

```
Epoch [9980/10000], Loss: 0.0000000
Epoch [9990/10000], Loss: 0.0000000
Epoch [10000/10000], Loss: 0.0000000
```

위 사진은 순서대로 RNN, LSTM, Transformer의 Loss 값을 출력한 결과입니다. Transformer 모델의 Loss 값에 대한 출력을 소수점 8번째 자리에서 반올림하도록 출력하여 0.0000000이라고 출력되지만 더 적은 Epoch임에도 불구하고 RNN과 LSTM보다 시계열 데이터를 예측하는 데에서 뛰어난 성능을 보인다는 것을 알 수 있습니다.

한계 및 문제점, 발전 방안

RNN, LSTM 모델보다 Transformer 모델로 학습시켜 주가를 예측하는 것이 성능이 뛰어나다는 것은 보여지나 여전히 외부 상황에 대한 주가 변동을 예측하지 못했다는 한계를 보입니다. 이를 극복하기 위해 Transformer 모델의 Epoch을 15000, 30000까지 올려서 학습시키려 여러번 시도하였으나 시간이 굉장히 오래 걸려 매번 Colab의 할당량을 다 사용하여 학습을 완료하지 못하였습니다.
문제점을 해결하기 위해 보고서를 제출하더라도 계속 학습을 시도하여 어느 정도까지 예측 성능을 보이는 지 확인해 볼 생각입니다. Epoch을 올려 학습을 진행하였는데도 원래 목표였던 우크라이나 러시아 전쟁이 터지기 직전의 주가 변동을 예측하지 못한다면, 모델을 하나 더 활용하여 외부 상황을 학습할 수 있는 뉴스 데이터나 키워드를 학습시켜 성능을 더 끌어올려보려 합니다.