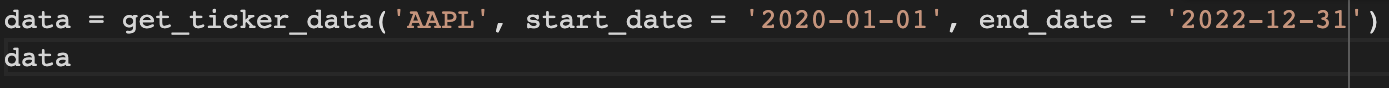
<Transformer 기반 주가 예측>

0. 연구 목적 및 데이터셋 소개

최근 Transformer를 활용한 시계열 예측 연구가 많이 진행되고 있다. 시계열 데이터를 처리할 때 Transformer의 encoder 구조와 self-attention 기법을 사용하면 시간 정보를 고려하면서 데이터를 처리할 수 있기 때문에 활용하여 주가 예측 모델을 구축할 수 있다. 결과적으로 Transformer 모델을 활용하면 이전 일정 기간 동안의 주가 데이터를 입력으로 사용하고, 해당 일자의 주가를 출력으로 예측하는 것이 가능하다. 시계열 데이터에 Transformer를 적용하는 이점은 장기적인 의존성을 파악할 수 있는 구조와 병렬 처리가 가능한 구조를 가지고 있기 때문이다. 이를 통해 기존의 시계열 예측 방법보다 더 정확하고 효과적인 예측이 가능하다. 또한, Transformer는 입력 데이터의 길이에 상관없이 일관된 예측 성능을 보여주는 특징이 있어 다양한 시계열 데이터에 적용할 수 있다. 데이터셋은 Yahoofinancial을 통해 불러온다. Yahoofinancial은 Python 라이브러리로, Yahoo finance에서 제공하는 API를 활용해 시계열 주가 데이터를 쉽게 불러올 수 있다. Ticker라는 종목명과 원하는 날짜를 입력하면, 해당 기간의 주가 데이터를 가져올 수 있다. 사용된 데이터셋은 AAPL이라는 ticker를 가진 종목(애플)의 주가를 2018년 1월 1일부터 2022년 12월 31일까지 가져와 학습에 사용한다. 데이터셋의 각 열은 date, high, low, open, close, volume, adjclose로 구성되어 있다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Date | High | Low | Open | Close | Volume | Adjclose |
| 날짜 | 고가 | 저가 | 시가 | 종가 | 거래량 | 수정 주가 |

위의 데이터 중에서 종가를 기준으로 주가를 예측하고자 한다. 다음 코드는 2020년 1월 1부터 2022년 12월 31일까지의 애플 주가를 가져오는 python code이다.



학습에 사용된 데이터는 2018년 1월 1일부터 2022년 12월 31일까지의 데이터를 사용했다. 또한 2023년 1월 1일부터 2023년 6월 14일까지의 데이터를 활용해 실제 예측을 진행했다. 다음 그래프는 이를 시각화한 것이다.

라인, 그래프, 텍스트, 폰트이(가) 표시된 사진

자동 생성된 설명

연구는 다음과 같은 순서로 진행된다.

1. 데이터 불러오기 및 시각화

2. 데이터 전처리

3. 모델 구축 및 학습

4. 모델 평가 및 결과 도출

1. Transformer

Transformer는 자연어 처리(Natural Language Processing, NLP) 분야에서 주로 사용되는 딥러닝 모델 중 하나로, 2017년 NIPS에서 발표된 <Attention is all you need>라는 논문에서 처음 제안된 모델이다. 기존의 순차적인 처리 방식을 가진 순환 신경망(Recurrent Neural Network) 모델 대신에, 전체 문장을 한 번에 처리하는 방식을 도입하여 효율적인 학습과 생성을 가능하게 했다. Transformer 모델은 크게 두 가지 주요 구성 요소로 이루어져 있다. 첫 번째는 Self-Attention 메커니즘이고, 두 번째는 Positional Encoding이다.

Self-Attention은 입력 문장의 모든 단어 간의 상관 관계를 계산하는 메커니즘이다. 각 단어는 쿼리(Query), 키(Key), 값(Value) 세 가지 벡터로 표현되며, 입력 문장의 모든 단어 간의 유사도를 계산하여 각 단어가 다른 단어에 얼마나 관심을 가지는지를 판단한다. Self-attention의 계산 과정은 다음과 같다.

1. 입력 시퀀스 임베딩(Embedding)

먼저 입력 시퀀스의 단어들은 임베딩 과정을 거쳐 벡터로 표현된다.

2. 쿼리(Query), 키(Key), 값(Value) 생성

각 단어의 임베딩 벡터를 통해 쿼리(Query), 키(Key), 값(Value) 벡터를 생성한다. 이때, 세 벡터는 입력 시퀀스의 모든 단어에 대해 독립적으로 생성됩니다. 각각의 쿼리, 키, 값을 계산하는 방법은 모든 단어의 임베딩 벡터 행렬과 쿼리, 키, 값에 해당하는 각각의 가중치 행렬을 행렬 곱 연산을 통해 모든 단어에 대해 병렬적으로 계산한다. 이 때 가중치 행렬을 여러 개 사용해 여러 개의 쿼리, 키, 값을 얻음으로써 다양한 특징을 학습할 수 있도록 하는 것을 multi-head attention이라고 하며, head의 수만큼 최종적으로 attention 행렬을 얻게 된다.

3. 유사도 계산

쿼리(Query) 벡터와 키(Key) 벡터 간의 유사도를 계산한다. 일반적으로 내적(dot product)을 사용하여 유사도를 계산하는데, 이는 두 벡터가 얼마나 유사한 방향을 가지는지를 나타낸다. 유사도는 단어 간의 관련성을 측정하는 점수(score)로 사용된다.

4. 스케일링 및 Softmax

유사도를 계산한 후에는 스케일링 작업을 수행한다. 이를 통해 유사도 값을 조정하고, 특정한 범위에 맞춘다. 일반적으로는 유사도를 특정 상수의 제곱근으로 나누어 스케일링을 수행한다. 그리고 Softmax 함수를 통해 정규화하여 각 단어에 대한 가중치를 얻는다. Softmax 함수를 통과한 결과는 확률값으로 해석된다.

5. 가중합 계산

Softmax 함수를 통과한 가중치와 값(Value) 행렬을 곱하여 각 단어의 가중합 행렬을 계산한다. 이 가중합 행렬은 다른 단어들과의 상관관계를 고려한 각 단어의 임베딩 표현으로 볼 수 있다.

폰트, 텍스트, 화이트, 서예이(가) 표시된 사진

자동 생성된 설명

도표, 스크린샷, 텍스트, 라인이(가) 표시된 사진

자동 생성된 설명

6. 출력 계산

각 단어의 가중합을 모두 합하여 최종적인 Self-Attention 행렬을 얻는다. 이 출력 벡터는 입력 시퀀스의 각 단어에 대한 정보를 종합한 결과이다. Multi-head attention의 경우 head의 수만큼 attention 행렬을 얻게 되는데 여러 개의 attention 행렬을 단순히 concatenation 후 최종적으로 output 가중치를 곱해 최종 output 행렬을 얻게 된다. 이 때 연산을 수행한 뒤에도 차원은 동일하게 유지된다.

텍스트, 도표, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

텍스트, 라인, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명스크린샷, 텍스트, 도표, 라인이(가) 표시된 사진

자동 생성된 설명

텍스트, 폰트, 화이트, 타이포그래피이(가) 표시된 사진

자동 생성된 설명

Positional Encoding은 단어의 위치 정보를 모델에 전달하기 위한 메커니즘이다. Transformer는 단어의 순서 정보를 전혀 고려하지 않는 구조를 가지기 때문에, Positional Encoding을 통해 단어의 상대적인 위치를 모델에 알려준다. 이를 통해 모델은 입력 문장에서 단어의 순서를 인식할 수 있게 된다. 결과적으로 각 단어의 임베딩 벡터와 문장 내 단어의 위치 정보 반영을 위한 Positional encoding 벡터를 더해 사용한다.

텍스트, 폰트, 화이트, 서예이(가) 표시된 사진

자동 생성된 설명

Transformer는 여러 개의 인코더(Encoder)와 디코더(Decoder) 층으로 구성된다. 인코더는 입력 문장을 Embedding 하고, Self-Attention과 Feed-Forward 신경망 계층을 거쳐 문장의 특성을 추출한다. 따라서, multi-head attention layer와 feed-forward layer 여러 개가 반복적으로 나타나는 구조를 가진다. 디코더 또한, 마찬가지로 인코더와 같이 multi-head attention layer와 feed-forward layer를 반복적으로 사용하는 구조를 갖는다. 디코더는 인코더의 출력과 이전에 생성된 단어들을 입력으로 받아 다음 단어를 예측한다. 최종적으로 이를 반복하여 문장을 생성하게 된다. 이 때 성능 향상을 위해 residual learning을 활용한다.

Transformer 모델에서 사용되는 residual Learning(잔차 학습)은 네트워크의 깊이를 늘리고 학습을 더욱 쉽게 만들기 위한 기법이다. Residual Learning은 네트워크의 중간 층에서 입력 신호와 출력 신호 간의 차이를 학습하는 것을 의미한다. 일반적인 딥러닝 네트워크에서는 층을 통과할 때마다 변환된 특징이 중첩되는 구조이다. 이러한 중첩은 층이 깊어질수록 그 차이가 커지고, 학습이 어려워질 수 있다. Residual Learning은 이러한 문제를 해결하기 위해, 네트워크의 각 층이 입력 신호에 대한 변화를 학습하는 대신, 입력과 출력 사이의 잔차(차이)를 학습하는 방식을 사용한다. 즉, 입력 신호를 그대로 전달하고 변화된 부분만을 학습하는 것입니다. 이를 구현하기 위해, Transformer 모델에서는 skip connection을 사용한다. Skip connection은 각 층의 입력과 출력을 더하는 방식으로 이루어진다. 이를 통해 학습이 안정화되고, 네트워크의 깊이를 더욱 효과적으로 활용할 수 있게 된다. 잔차 학습은 네트워크의 깊이를 증가시킬 때 그 이점을 극대화하고, 그래디언트 소실(gradient vanishing)과 같은 문제를 완화하여 더 깊고 복잡한 모델의 학습을 가능하게 한다. 이는 Transformer와 같이 매우 깊은 구조를 가지는 모델에서 특히 유용하며, 자연어 처리 작업에서 성능 향상을 이끌어낸다.

결과적으로, Transformer는 학습과 추론 단계에서 모두 효과적으로 동작한다. 학습할 때는 입력 문장과 실제 출력 문장을 이용하여 모델의 매개 변수(Parameter)를 업데이트하고, 추론할 때는 이전에 생성한 단어를 입력으로 받아 다음 단어를 예측한다.

Transformer는 기존의 순환 신경망과 비교하여 학습 속도가 빠르고 병렬 처리가 가능하며, 긴 문장에 대해서도 잘 동작한다. 그래서 최근의 다양한 자연어 처리 작업에 널리 사용되고 있으며, 기계 번역(Machine Translation), 챗봇(Chat Bot), 문장 생성 등 다양한 자연어 처리 과제에 상용화되어 활용되고 있다.

텍스트, 도표, 스크린샷, 평면도이(가) 표시된 사진

자동 생성된 설명

2. 데이터 전처리

먼저, 최댓값과 최솟값을 활용해 min-max scaling을 진행했다. 각 데이터(종가)에서 최솟값을 빼고 이를 최댓값에서 최솟값을 뺀 값으로 나눠준다. Sklearn.preprocessing.MinMaxScaler를 사용해 0과 1 사이의 값으로 변환한다. 이후, 데이터셋을 8개의 timestep 간격으로 분할한다. Transformer의 경우 입력 sequence의 길이가 달라져도 가능하지만, 이전 8일의 종가를 기준으로 예측을 진행하기 위해 8일 간격으로 분할을 진행했다. 이 때 train : validation은 8 : 2로 구성했다.

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

3. 모델 학습

사용한 모델은 Transformer의 인코더 부분만을 사용했다. Python을 활용해 다음과 같이 구현했다.

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

Build\_model 함수는 다음과 같이 입력 parameter로 head의 개수, encoder의 수 등을 받아서 모델을 구성한다.

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

또한, 학습이 잘 되도록 하기 위해 learning rate decay를 사용했다. Learning rate decay(학습률 감소)는 딥러닝 모델의 학습 과정에서 학습률(learning rate)을 점진적으로 감소시키는 기법이다. 이는 학습 초기에는 큰 학습률을 사용하여 빠르게 수렴하고, 학습이 진행됨에 따라 학습률을 조정하여 더 정교한 최적화를 수행하는데 도움을 준다. 학습률은 가중치 업데이트에 사용되는 스케일링 인자로, 모델이 학습을 통해 최적의 파라미터를 찾을 때 얼마나 크게 업데이트할지를 결정합니다. 너무 큰 학습률을 사용하면 수렴하지 않는다. 반대로 너무 작은 학습률은 수렴에 시간이 오래 걸리거나 local minimum에 빠질 수 있다. 따라서 적절한 학습률 설정은 모델의 성능과 수렴 속도에 영향을 미친다. 또한, patience는 10으로 설정해 MSE 기준으로 성능 향상이 없으면 early stopping 하도록 설정했다.

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

따라서 다음과 같이 모델을 구성했다. 임베딩 벡터 사이즈는 46, attention head의 수는 60, Optimizer는 Adam을 사용했다.

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

4. 모델 평가

Training loss와 validation loss의 그래프는 다음과 같다. 또한 실제 주가와 예측된 주가의 그래프는 다음과 같다. 그래프를 보면, training loss와 validation loss가 거의 0에 가깝기 때문에 거의 정확하게 예측하는 것이 가능하다는 것을 알 수 있다.

텍스트, 스크린샷, 도표, 그래프이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 그래프, 라인이(가) 표시된 사진

자동 생성된 설명