**@Transformer**

지난 시간 단어의 의미 추론을 분류로 학습시킴

연관 개념 - RNN, LSTM, encoder와 decoder, Attention

#RNN

언어데이터의 특징 - sequentiality

이런 특징을 잘 다루는 것이 RNN 네트워크

순서로 연결되는 데이터 - sequential한 데이터  
Ex) 음향, 언어…

히든 레이어가 순환되어 sequential한 정보를 연결시킴

RNN은 short term 메모리로 인해 이전 데이터를 기억하지 못함

#LSTM

RNN의 vanishing gradient를 해결하는 모델

RNN은 히든 레이어 하나만 있었던 반면, LSTM은 셀이라는 추가된 long-term 메모리 레이어가 있음

무엇을 forget하고 keep할지를 결정

시그모이드, 탄젠트 하이퍼볼리 등의 함수들을 각 게이트마다 추가함

Forget, input, output… 게이트를 가짐

1. 시그모이드 값을 써서 잊어야 할 것을 0으로 만들어서 잊게 만듦

2. 하나는 시그모이드, 하나는 탄젠트하이퍼볼리를 통과함 (새로운 특징들이 더해짐)  
 잊어야 할 값과 더해져야할 값이 셀로 넘어감

3. 아웃풋 게이트는 다음 스테이트는 어떻게 될지를 결정함

#GRU

Reset 게이트와 output 게이트로 LSTM을 단순화시킴

속도도 빠르고 성능도 뛰어남

뉴욕대학교 조경현 교수가 처음 함께 고안한 모델

#Seq2seq

기존의 RNN은 마지막 값으로 결과값을 얻는 반면, seq2seq의 인코더 디코더 모델은 인코더로 입력 문장을 하나의 벡터로 만들고, 디코더로 벡터를 target language로 변환함

입력 seq에서 인코더로 생성된 vector를 context vector라고 부름

하지만 이 역시 문장이 길면 길어질수록 하나의 제한된 벡터에 표현하는데 제한이 생김

RNN의 기본적 문제에 공착하게 됨

이런 인코더 디코더 문제를 해결하기 위해 attention모델이 나옴

LSTM에서는 keep, forget을 나눈 것처럼, 어디에 주목할 것인지를 score로 계산하는 것이 attention

인코딩의 어느 부분에 집중하게 될 지를 결정

모든 인코딩의 정보를 디코딩에서 사용하자 -> 인코더와 디코더 사이에 attention layer를 생성

모든 인코더의 히든 스테이트를 제공

인코더- 디코더 모델은 번역 모델에서 많이 사용되는 모델

기계번역을 할 때, source와 target이 어떻게 연결되었는지 align하는 것이 중요 (어텐션으로)

#attention

각각의 히든스테이트에서 어떤 입력값이 어떤 출력값과 연결되는지 관계 설정

Decoder의 히든 스테이트와 인코더의 히든스테이트를 비교하여 반영

Dot-product를 한다는 것은 두 값이 얼마나 비슷한지를 구하는 것

각각의 스코어를 확률값으로 바꾸는 것이 소프트맥스 함수를 취함

다음 어텐션 레이어에서 확률값은 원래의 값과 곱함

어텐션의 가장 주목되는 값을 디코더와 연결시킴

Dot-product값이 유사도를 구한 것임

#Attention의 예시

Bahdanau - 어텐션 모델의 선구자

Luong - 인코더 레이어를 두개 쌓고 진행

앞으로도 가고, 뒤로고 감

인코딩의 레이어를 몇 개씩 쌓는지는 사람마다 다름

GNMT - 2016년 8개의 LSTM를 거쳐서 어텐션 값을 디코딩을 함

마지막 출력 층에서 ResNet (residual net)을 거치지 않고 출력 값으로 전달

히든 레이어를 여러 개를 거치는데, 어떨 때는 거치지 않고 아웃풋을 얻는 것이 더 나음

8개에서 나온 값

#summary

벡터값 하나로 만들지 말고, 각 스테이트의 값을 사용하자

**멀티해드 어텐션에서 포지셔널 정보를 기억하는 것과 포지셔널 인코딩에서 포지셔널 정보를 기억하는 것의 차이?**

**Value가 무엇인지?**