딥러닝 케라스 모델링 및 앙상블 구축

KDT 5기 김지혜

- Language Model 이란?
- GPT와 BERT의 차이점
- koGPT2를 활용한 챗봇 구현
- koBERT를 활용한 감성 기반 음악 추천

I. Language Model 이란?

기본적으로는 문장의 일부를 보고 다음 단어를 예측하는 것을 할 수 있는 머신 러닝 모델이다.

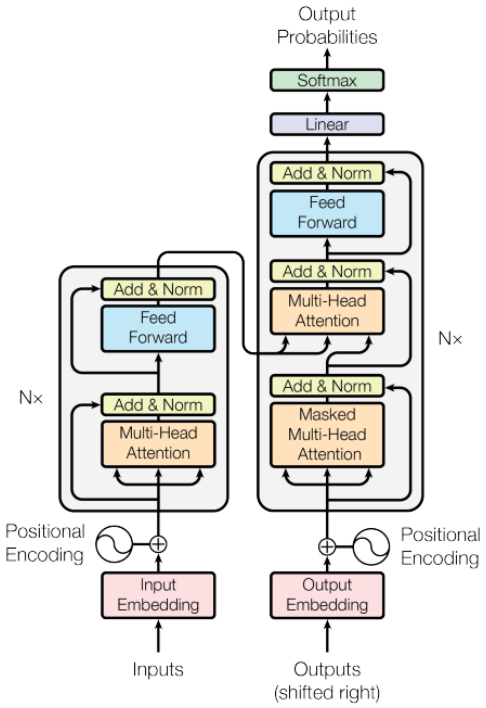


Figure 1: The Transformer - model architecture.

II. GPT와 BERT의 차이점

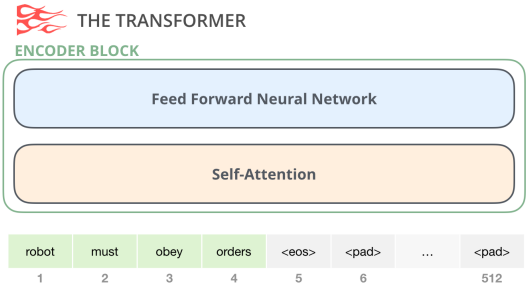
GPT-2는 transformer의 decoder 블럭으로 구성된다. 반대로 BERT는 transformer의 encoder 블럭을 사용한다.
하지만 이 둘 간의 주요한 차이는 GPT2가 다른 전통적인 language model들 처럼 한번에 하나의 token을 출력한다는 것이다.

이러한 모델들은 각 token이 생성된 후에 입력 시퀀스에 더해지는 방식으로 동작한다.
그러한 새 시퀀스는 다음 단계에서 모델의 입력으로 들어간다. 이것을 auto-regression이라고 부른다. 이것은 RNN을 엄청나게 효과적으로 만든 방법 중 하나이다.

GPT2와 TransformerXL, XLNet과 같은 후속 모델들은 본질적으로 auto-regression이다.
하지만 BERT는 그렇지 않다. 이것은 상충관계(trade off)가 있다. auto-regression 특성을 잃는 대신, BERT는 더 좋은 결과를 내기 위해 단어의 양쪽 방향으로부터의 context를 활용할 수 있는 능력을 얻었다.

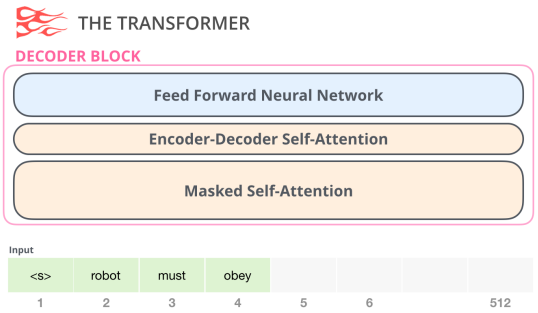
❖ Transformer block의 진화

Encoder block
- self-Attention
- Feed Forward Neural Network



최초의 transformer 논문에서 encoder block은 입력을 특정 최대 시퀀스 길이까지 가질 수 있다.
입력 시퀀스가 이 길이보다 짧으면 괜찮다.
시퀀스의 나머지 부분에 패딩을 덧붙일 수 있다.

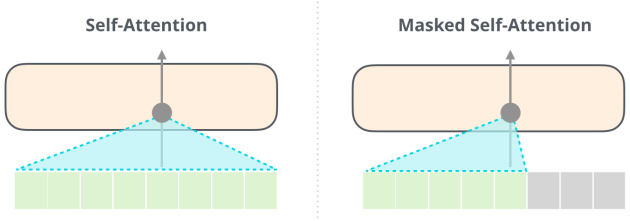
Decoder block
- Masked Self-Attention
- Encoder-Decoder Self-Attention
- Feed Forward Neural Network



encoder block의 아키텍처를 살짝 변형한 decoder block이 있다. encoder로 부터의 특정 segment에 attention을 줄 수 있도록 하는 레이어가 있다.

decoder의 self-attention 레이어에서의 주요 다른 특징은 token들을 masking하는 것이다.
BERT 처럼 word를 [mask]로 치환하는 것이 아니라 self-attention 계산시에 계산되어야 하는 위치의 오른쪽에 있는 token들로부터의 정보를 막는 방법으로 차단한다.

BERT가 사용하는 self-attention과 masked self-attention이 확연히 다르다는 것은 중요하다. 일반 self-attention block은 자신보다 오른쪽에 있는 token을 계산 과정에서 볼 수 있도록 한다. masked self-attention의 경우에는 이런 상황을 막는다.



The Decoder-Only Block
이 모델은 transformer의 encoder를 버렸다.
이 초창기의 transformer 기반의 language model은 6개의 transformer decoder block을 쌓아 구성했다.

이 decoder block들은 동일하다. 첫번째 block을 확대했기 때문에 self-attention 레이어가 masked 버전임을 알 수 있다. 현재는 이 모델이 특정 segment에서 최대 4000개 token까지 참조할 수 있다.

OpenAI의 GPT-2 모델은 이러한 decoder만으로 구성된 block들을 사용한다.

III. koGPT를 활용한 챗봇 구현

GPT_2는 1024개 token을 처리할 수 있다. 각 token은 각자의 경로를 따라서 모든 decoder block으로 흘러간다.

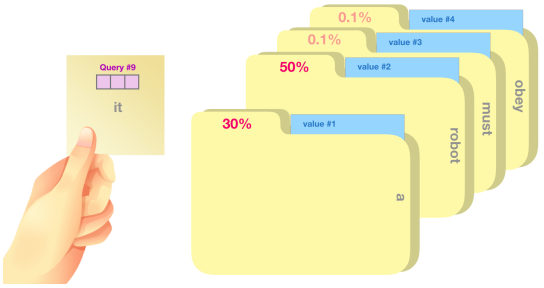
훈련된 GPT-2 모델을 돌려보는 가장 간단한 방법은 그것 자체의 램블링을 하도록 만드는 것이다.
rambling 방법에서 간단히 start token을 입력해서 단어들을 생성하기 시작하도록 만들 수 있다.

모델은 input token 하나를 가지고 있으므로 경로가 1개만 활성화 된다. token이 모든 레이어를 연속적으로 거쳐 처리되고 나면, 그 경로를 따라 vector가 생성된다. 그 vector에 모델의 어휘(vocab) 전체(모델이 알고 있는 모든 단어들)에 대해 점수가 매겨질 수 있다. 이 경우에 우리는 가장 높은 확률을 갖는 단어를 선택한다. GPT-2는 top-k라는 parameter를 가지고 있어서 우리는 모델이 top단어가 아닌 다른 단어를 샘플링하게 만들 수도 있다.

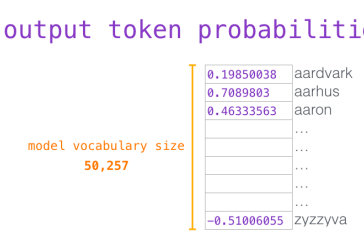
Self-Attention 프로세스
self-attention은 segment에서 각 token의 경로를 따라 처리된다. 중요한 요소들은 세가지 vector이다.

- Query: 다른 모든 word들과 score를 계산하는데 사용되는 현재 단어의 representation이다. 우리는 현재 처리 중인 token의 query 값만 고려한다.
- Key: Key vector는 segment에서 모든 word들에 대한 레이블과 같다. 관련 word를 검색할 때 매칭해보는 항목이다.
- Value: Value vector는 실제 word representation이다. 각 단어가 얼마나 관련이 있는지 score를 매기고 나면, 현재의 word를 표현하기 위한 합산한 값이다.

대략적으로 비유해보자면, 서류 캐비넷에서 어떤 서류를 찾는 것과 같다고 생각할 수 있다. Query는 찾고자 하는 주제를 적는 메모지이다. Key는 캐비넷 안의 서류 폴더들에 달린 레이블과 같다. 메모지와 tag를 매칭시키면, 폴더에서 내용물을 꺼내는네, 이 내용물이 바로 value vector이다. 단지 다른 점은 하나의 value만 찾는 것이 아니라 여러 폴더들에서 여러 value들의 혼합을 찾는다는 것이다.
Query vector를 각 key vector에 곱해서, 각 폴더 별 score 값을 만든다.



모델출력
모델의 최상위 block이 output vector를 생성할 때 모델은 그 vector와 embedding matrix를 곱한다. embedding matrix의 각 행은 모델 어휘 단어들의 embedding에 해당된다. 이 곱셈의 결과는 모델의 어휘에서 각 word에 대한 score로 해석된다.



가장 높은 score를 갖는 token을 선택할 수도 있다. 하지만 모델이 다른 word들도 고려한다면 더 좋은 결과를 얻을 수 있다. 더 좋은 전략은 전체 리스트에서 score를 어떤 word를 고르기 위한 확률값으로 사용하여 단어를 선택하는 것이다. 절충안은 top_k를 40으로 잡고 모델이 가장 높은 score를 갖는 40개의 word를 고려하도록 하는 것이다. 그렇게 해서 모델은 하나의 word를 출력하면서 한 iteration을 종료한다.
모델은 전체 컨텍스트가 생성될 때까지 혹은 EOS token이 생성될 때까지 iteration을 계속 수행한다.

챗봇
pre-train된 모델인 koGPT-2를 사용하여 일상대화가 가능한 챗봇을 구현한다. 허깅페이스 skt/kogpt2-base-v2를 사용하여 모델과 토크나이저를 선언한다.

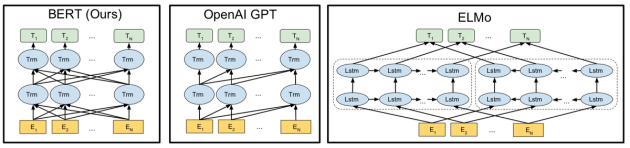
<u>데이터</u>
중립(0), 부정(1), 긍정(2)으로 라벨링된 챗봇데이터를 사용하여 모델을 훈련시킨다.

<u>tokenizer</u>
허깅페이스의 PreTrainedTokenizer인 GPT2Tokenizer를 사용한다.

- Tokenizing: 입력 문자열을 token id로 변환, token id를 다시 문자열로 변환의 기능
- 기존의 구조에 독립적으로 추가적인 token들을 추가하는 기능
- Special token들을 (mask, BOS, EOS) 관리하는 기능

파라미터
- bos_token: 문자의 시작을 나타내는 token
- eos_token: 문장의 끝을 나타내는 token
- unk_token: 모르는 단어를 나타내는 token
- pad_token: 동일한 batch 내에서 입력의 크기를 동일하게 하기 위해서 사용하는 token

IV. koBERT를 활용한 감성기반 음악추천



BERT는 화살표가 양방향으로 뻗어나가는 모습으로 이는 마스크드 언어 모델을 통한것이다.

BERT의 사전 훈련을 위해 입력 텍스트의 15% 단어를 랜덤으로 마스킹한다. 이후 신경망이 가려진 단어들을 예측하도록 한다. 쉽게 말해 빈칸 뚫고 맞추기 학습이다.

- 80% 단어들을 [MASK]로 변경
- 10% 단어들은 랜덤으로 단어 변경
- 10% 단어들은 동일하게 유지

이는 [MASK]만 사용할 경우 [MASK] 토큰이 파인튜닝 단계에선 나타나지 않기 때문에 사전 학습 단계와 파인튜닝 단계에서의 불일치 문제를 피하기 위함이다.

BERT는 총 3개의 임베딩 층을 사용한다.

- WordPiece Embedding

실질적인 입력이 되는 워드 임베딩
임베딩 벡터의 종류는 단어 집합의 크기로 30,553개

- Position Embedding

위치 정보를 학습하기 위한 임베딩
임베딩 벡터의 종류는 문장 최대 길이인 512개

- Segment Embedding

두 개의 문장을 구분하기 위한 임베딩
임베딩 벡터의 종류는 문장의 최대 개수인 2개

감정 기반 음악 추천
pre-train된 모델인 koBERT를 사용하여 감성분석을 하고 분석된 모델을 사용하여 음악 추천을 구현한다.

<u>데이터</u>
감정 말뭉치 데이터를 사용하여 5가지 감정(행복, 슬픔, 당황, 분노, 불안)으로만 전처리를 통해서 모델을 훈련시킨다.

<u>koBERT 학습모델</u>
다중분류할 클래스 수 만큼 num_classes 변수를 수정해주어야 한다. 이번 프로젝트에서는 5가지의 class를 분류하기 때문에 5로 입력해주었다.

모델과 옵티마이저(Adam), loss, scheduler를 선언해주고 처리해준 데이터를 통해 모델을 훈련시킨다.

reference

GPT-2
https://chloamme.github.io/2021/12/08/illustrated-gpt2-korean.html#h-%ED%8C%8C%ED%8A%B8-2-%EA%B7%B8%EB%A6%BC%EC%9C%BC%EB%A1%9C-%EC%84%A4%EB%AA%85%ED%95%98%EB%8A%94-self-attention-