

3주차 1조

팀원: 강용진, 조현진, 조선빈






```
lookup.KeyValue  
f.constant(['em  
=tf.constant([0  
ce = tf.lookup.StaticV  
init,  
num_oov_buckets=5)
```

```
lookup.StaticVocabular  
initializer,  
num_oov_buckets,  
lookup_key_dtype=None  
name=None,  
experimental_is_sparse
```

강용진

RAG: Retrieval-Augmented Generation

- 이번 competition의 경우 제출 과정에서 인터넷 사용을 할 수 없어 **Pre-trained model**을 이용해야 함
- **Pre-trained language model**은 **task-specific**하고 정보들이 **parameterize** 되어있음
- 그로 인해 **knowledge-intensive**한 (외부지식을 요하는) 이번 **competition**과 같은 주제에 취약할 수 있음
- 이런 문제점을 보완하기 위해 많은 유저들은 **Pre-trained model**과 함께 **RAG**를 이용하였음

	Platypus2-70B with Wikipedia RAG Updated 1h ago Score: 0.836 · 46 comments · Kaggle - LLM Science Exam +7	▲ 111 Gold ...
	Uni-TianYan-70B with Wikipedia RAG Notebook copied with edits from simjeg · Updated 7d ago Score: 0.812 · 17 comments · Kaggle - LLM Science Exam +9	▲ 34 Silver ...
	Xwin-LM-70B-V0-1 with Wikipedia RAG Notebook copied with edits from CPMP · Updated 4d ago Score: 0.825 · 4 comments · Kaggle - LLM Science Exam +9	▲ 29 Silver ...
	How To Train Open Book Model - Part 1 Updated 25d ago 62 comments · Kaggle - LLM Science Exam +2	▲ 225 Gold ...
	orca_mini_v3_70b with Wikipedia RAG Notebook copied with edits from CPMP · Updated 7d ago Score: 0.81 · 0 comments · Kaggle - LLM Science Exam +9	▲ 7 Bronze ...

RAG: Retrieval-Augmented Generation

- RAG: Retrieval을 통해, Pre-trained model바깥의 **외부 정보**도 활용하여 답변을 생성할 수 있게 함.
- 이번 competition에서는 보통 Wikipedia 데이터가 사용되었고, 이는 주어진 문제 상황을 open-book에 가깝게 만들어줌
- 구체적인 작동 프로세스
 1. 참고할 정보들(예시: Wikipedia)을 정리, 저장해둔다.
 2. 질문 텍스트가 입력되면, 저장해둔 정보들 중 가장 유사한 것들을 찾는다.
 3. prompt를 작성해 LLM에게 전달한다.
 4. LLM이 응답을 생성한다.
- 이를 통해 이번 competition과 같이 도메인 지식이 많이 필요한 질문에 대해서도 더 나은 결과를 도출하게 할 수 있음
- 논문 링크 : <https://arxiv.org/abs/2005.11401>

조현진

tokenizer → NLP에서 주로 단어 단위로 나누는 것

데이터 처리를 용이하게 하기 위해서 사용

padding → 가변적 길이를 가지는 문장을 같은 길이로 맞춰주기 위해 사용

길이가 부족한 문장은 0으로 채움

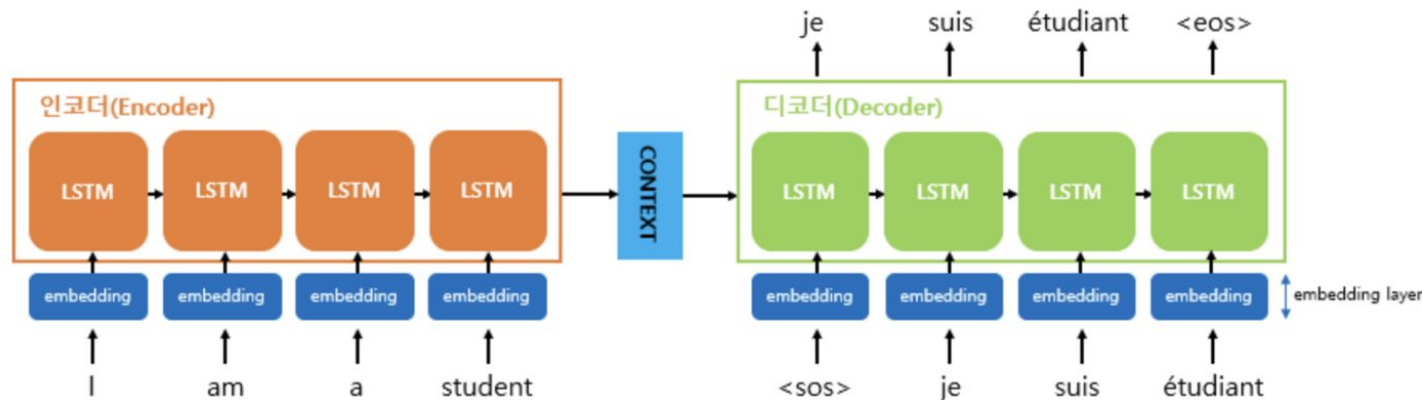
pre-padding → 앞에서 부터 0채워

post-padding → 뒤에서부터 0채워

주로 pre-padding이 고성능임 why? → 주로 사용하는 학습모델인 LSTM, GRU와 같은 recurrent model은 입력데이터의 순차적인 특성을 모델링함 즉, 마지막 단어가 입력으로 들어갈 때는 앞단어들의 시퀀스 모델링이 반영된 가장 중요한 상태인데 pre-padding의 경우 마지막에 0이 들어가는게 아님

8. ``collections.abc`` (Abstract Base Classes): 추상 베이스 클래스를 포함한 컨테이너 및 반복 가능한 객체에 대한 추상화를 제공하는 모듈입니다.
9. ``faiss`` (Facebook AI Similarity Search): 벡터 검색 및 유사성 검색을 위한 라이브러리입니다. 대규모 벡터 데이터를 빠르게 검색하는 데 사용됩니다.
10. ``SentenceTransformer``: 문장 임베딩을 생성하기 위한 라이브러리로, 자연어 처리 작업에서 문장의 의미를 잘 나타내는 벡터 표현을 만들 수 있습니다.

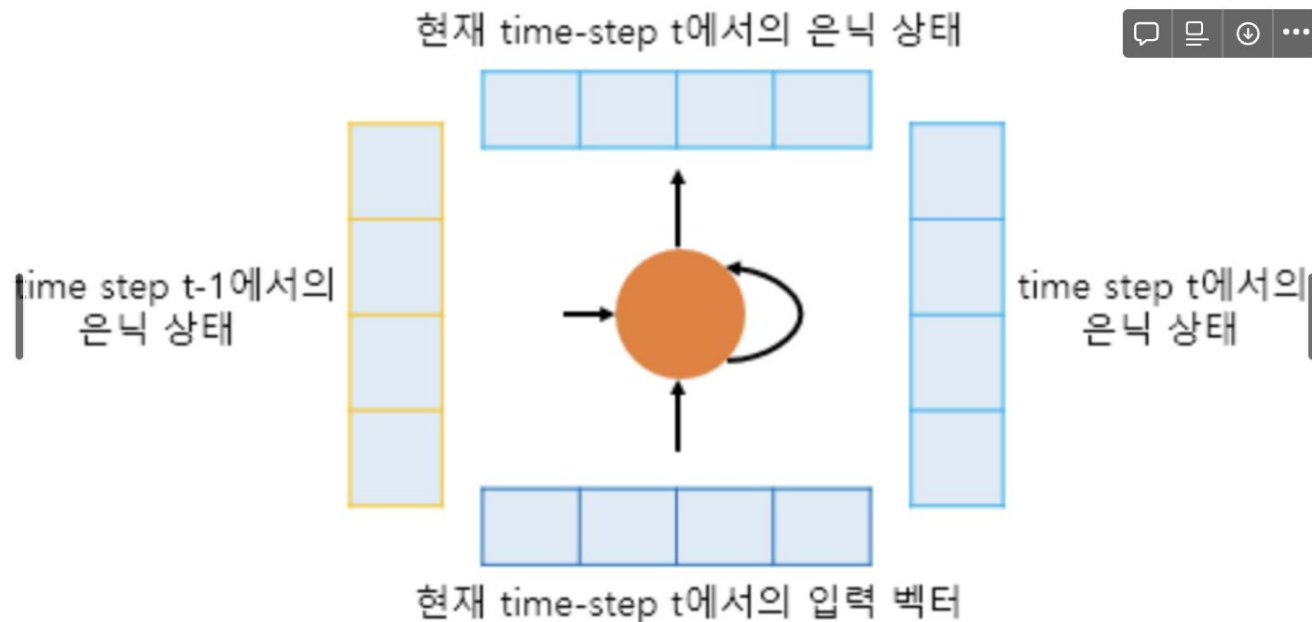
seq2seq



테스트 시 인코더에서 나온 **conext vector**가 디코더의 입력 값으로 들어가고 **이러를** 기반으로 **<sos>**입력 들어가고 그게 다음 **LSTM으로** 가서 **je**가 나오고 **je**가 다음 **LSTM으로** 들어가고... 근데 훈련 할 때는 좀 다른

훈련 시 컨텍스트 벡터와 실제 정답인 상황인 **<sos> je suis étudiant**를 입력되고 **je suis étudiant <eos>**가 나와야 된다고 정답을 알려주면서 훈련함. → **teacher forcing**

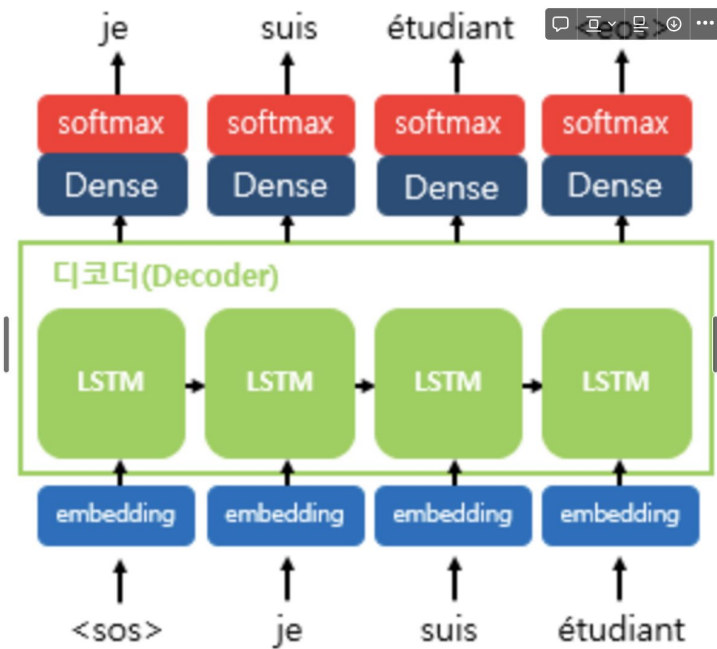
word embedding → 텍스트를 숫자 벡터로 **전환해줌**



RNN의 셀 확대한 거임

t-1일때의 은닉 상태(현시점까지의 상태를 요약한 벡터) + t에서의 context vector 받고 t에서의 은닉 상태 만들어서 넘김

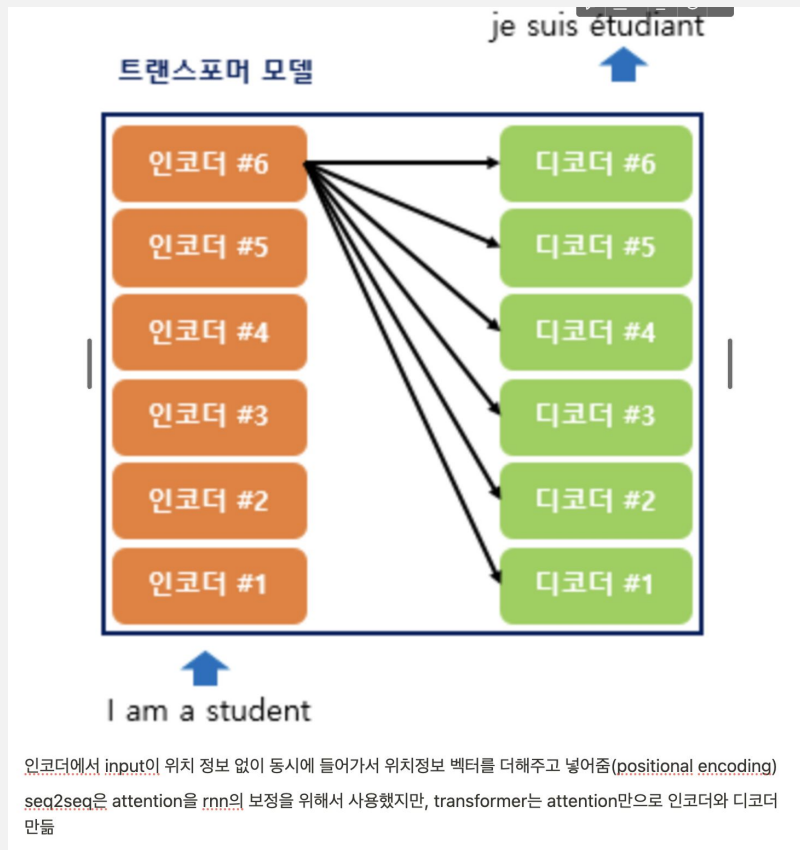
디코더에서는 인코더의 마지막 은닉상태(context vector) + <sos>입력을 받아서 예측 시작함



디코더에서 softmax 함수를 통해 뒤에 나올 수 있는 여러 단어 중 확률로 하나를 고를 (je) 그리고 이 단어가 다음 RNN 셀의 입력 값으로 들어감

→ 근데 인코더에서 디코더로 넘어가는 벡터 만들 때 정보가 손실 될 수 있음 → 보안을 위해 attention 기법 사용

Transformer



bert-model

pre-training된 벡터 값을 사용하고 동음이의어 다의어 구분 가능해짐

sentence-bert model

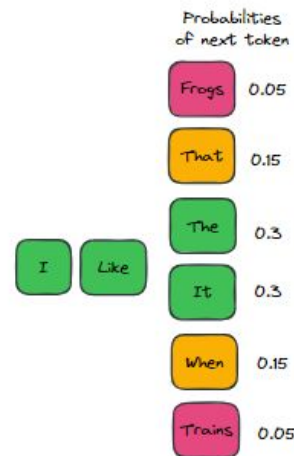
bert model 에서 더 개선된 부분임

조선빈

About LLM

LLM setting

- Temperature : unpredictability of a language model's output
 - ↑ : outputs become more creative and less predictable
 - amplifies the likelihood of less probable tokens
 - reduces the likelihood of more probable tokens
 - ↓ : outputs become less creative and more predictable
 - more conservative
 - predictable results
- Top P : the randomness of a language model's output
 - probability threshold
 - tokens whose combined likelihood surpasses the limit
- tokens' probabilities are summed to Top P
 - randomly pick one among these options



Example) Top P = 0.9

- tree : 50%
- roof : 25%
- wall : 15%
- window : 7%
- carpet : 3%

→ $50 + 25 + 15 > 90$

options = [tree, roof, wall]

About LLM

LLM setting

- Maximum Length : total # of tokens allowed to generate
- Stop Sequences : when to cease output generation
- Frequency Penalty : how frequently tokens appear
→ proportionally
- Presence Penalty : occur or not
→ flatly

Code Review : Ranked Predictions with BERT

BERT 모델 (Bidirectional Encoder Representations from Transformers):

- 트랜스포머 모델 아키텍처 중 하나로, 언어 이해 및 자연어 처리 작업을 위한 사전 훈련된 모델
- 양방향 문맥을 고려하여 문장을 이해 + 다양한 NLP 작업에 전이학습(Transfer Learning)으로 활용
- bert-base-cased
 - i. BERT 모델의 사전 훈련된 버전 중 하나
 - ii. 대문자와 소문자를 구분

Hugging Face Transformers 라이브러리:

- 자연어 처리를 위한 트랜스포머 모델을 쉽게 사용할 수 있도록 제공되는 라이브러리
- AutoTokenizer 및 AutoModelForMultipleChoice와 같은 클래스 사용

Trainer 및 TrainingArguments:

- Trainer: Hugging Face Transformers 라이브러리에서 제공하는 훈련 및 평가를 간단하게 수행할 수 있는 클래스
- TrainingArguments : 훈련 설정 및 하이퍼파라미터를 정의
- 모델 훈련 및 평가를 단순화하고 모델 체크포인트 저장 및 성능 추적을 관리

Code Review : Ranked Predictions with BERT

DataCollatorForMultipleChoice:

- 다중 선택 문제(Multiple Choice)를 위한 데이터 콜레이터
- 다중 선택 문제에서 각 선택지가 모델에게 입력될 때 데이터를 패딩하여 일관된 길이의 입력을 생성

학습 및 예측 과정:

- 코드는 주어진 학습 데이터셋을 사용하여 **BERT** 모델을 훈련
- **Trainer**를 통해 모델을 훈련하고 예측
- 예측된 결과는 **predictions_to_map_output** 함수로 가장 높은 확률을 가진 상위 3개의 선택지로 변환

데이터 전처리:

- 주어진 데이터셋은 학습용 및 테스트용 **CSV** 파일에서 로드
- 학습 데이터셋과 테스트 데이터셋은 **datasets** 패키지를 사용하여 불러오고 처리
- 문제와 선택지를 결합하여 문맥을 생성하고, 중요한 키워드를 추출하여 문맥에 추가