DSL RNN & Transformers Assignment :

Reading Materials

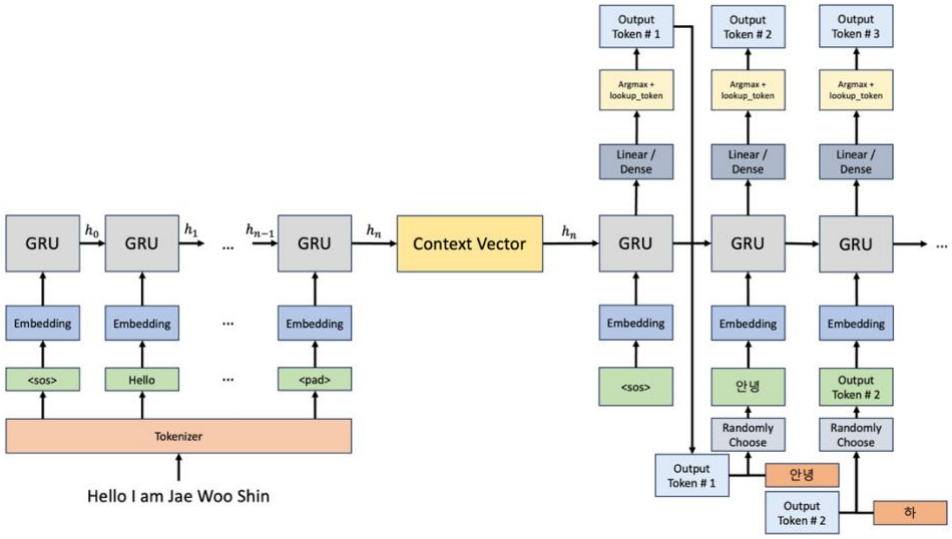
작성자 : 10기 신재우

1. 과제 설명

진행해야할 문제들은 모두 Seq2Seq 번역 모델과 관련되어 있으며, 다른 파일은 모델링 OT 때 잠시 소개드렸던 Gemini LLM 을 "어떻게 하면 더 활용할 수 있을까?" 에 대한 내용입니다. 후자의 파일의 경우 실행만 시켜서 제출하시면 되며, 전자의 경우 모든 문제들 (코드 빈칸) 들을 채워주셔서 제출하시면 됩니다. **제출하실 때에 데이터 셋은 포함하지 않으신 채 돌려진 상태의 코드 파일들만 포함된 zip 파일로 제출하시길 바랍니다.**

이번 과제에서는 Accuracy 구하는 부분이 없습니다. 정답과 똑같는지 아닌지 확실한 CV 와는 다르게 NLP 에서는 정답과 틀렸다고 해서 다 틀렸다고 보기가 어렵기 때문에 헷갈림 방지를 위해서 제외 시켰습니다. Wandb 역시 효율적으로 볼 수는 있으나 Loss 만 있기에 포함을 안 시켰습니다. 하이퍼 파라미터들의 경우 조정이 가능하긴 하지만 그것이 이번 과제의 핵심이 아니라고 생각해서 값들을 미리 주었습니다. Attention 이 안 들어간 상태에서의 모델에서는 결과가 안 좋은 것이 당연합니다! 참고를 위해서 모델 Architecture 를 다음과 같이 만들었습니다.

마지막 Linear / Dense Layer 이후에 활성화를 Softmax 를 통해서 시켜줘야 합니다!



2. Tokenizer (토크나이저) 란?

전에 했던 CNN 과제에서는 Embedding 과정을 크게 신경을 안 써도 됐습니다. 그에 대한 이유는 이미지 형태의 데이터를 다루었기 때문입니다. 즉 데이터가 수치화가 되어 있어서 머신러닝 혹은 딥러닝 기법들을 바로 사용해도 큰 문제 없이 모델이 잘 돌아갑니다.

하지만 Computer Vision 이외의 Task 에서는 Embedding 과정이 중요합니다. 이것은 특히나 NLP Task 에서는 더더욱 중요하며, 그에 대한 이유는 영어 문장을 예시로 갖고 오면서 설명하겠습니다.

Hello, my name is Jae Woo Shin.

위의 문장을 모델에 넣어서 번역을 시키고 싶다면 숫자 (벡터) 로 바꿔서 표현을 해야 합니다. 이것을 하기 위해서는 각 단어를 나누는 방법도 있을 것이며 문장부호 (마침표, 쉼표 등) 들도 따로 구분을 해주는 방법도 있습니다. 이렇게 나누는 것을 Tokenizing (토크나이징) 이라고 불리며 나눈 것들을 토큰 (Token) 들이라고 부릅니다. 위의 예시를 spacy 에서의 Tokenizer (토크나이저) 를 활용해서 풀면 다음과 같이 나눌 수가 있습니다.

[“Hello”], [“,”], [“my”], [“name”], [“is”], [“Jae”], [“Woo”], [“Shin”]

각 언어의 특성에 따라서 토크나이징 방법이 다를 수도 있습니다. 예로 한국어 문장을 갖고오겠습니다.

안녕하세요 저는 신재우에요.

위의 문장을 영어처럼 각 단어별로 나눌 수도 있지만 언어의 특성상 “안녕하세요” 를 하나의 토큰으로 바라보면 “안녕” 이란 단어가 가진 의미가 무색해지게 됩니다. 이는 한국어 언어의 특성 때문에 단어별로 나누기보다는 명사, 동사 등으로 품사별로 나누어서 토크나이징 하는 것이 더 효과적이라고 볼 수가 있습니다. 위의 문장에 적용시키면 다음과 같이 나오게 됩니다.

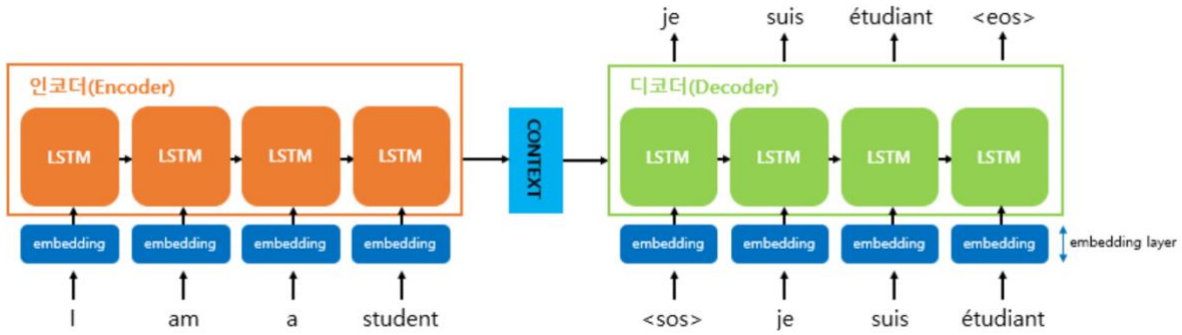
[“안녕”], [“하”], [“세요”], [“저”], [“는”], [“신”], [“재우”], [“에”], [“요”], [“.”]

다양한 토크나이저들이 있으며, 과제에서 사용될 것들은 Spacy 와 Kkma 입니다.

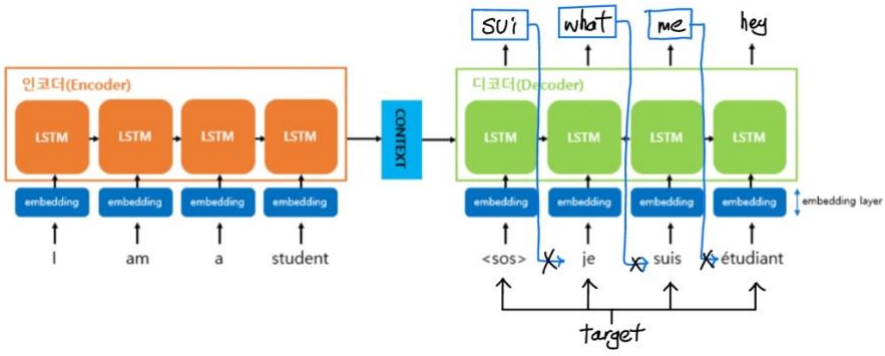
참고 :
https://wikidocs.net/64779, https://soy3on.tistory.com/170,
https://velog.io/@dayday/%EC%9E%90%EC%97%B0%EC%96%B4%EC%B2%98%EB%A6%AC-Tokenizing

3. Teacher Forcing

Seq2Seq 모델은 토큰들을 순차적으로 받으면서 다음 토큰을 예측하는 방식으로 작동이 됩니다. 즉 예측한 토큰이 다음 Time Stamp 의 Input 에 사용이 됩니다. 그렇다는 뜻은 예측한 토큰이 틀리면 모델 성능에 영향이 클 수도 있다는 뜻입니다. 이것을 구제적으로 살피기 위해서 모델 Architecture 를 갖고 오겠습니다.



위의 예시처럼 "je" 란 토큰을 디코더에서 예측했지만, 만약에 "je" 가 아닌 엉뚱한 토큰을 예측했다면 그 뒤로 나오는 토큰들도 엉뚱하게 예측이 됩니다. 이것을 방지하기 위해서 Training 과정에서는 Transformers 세션 때 배웠던 Teacher Forcing 기법이 적용됩니다.



위의 그림과 같이 모델의 Output 을 다음 토큰으로 넣는 것이 아닌, 타겟 즉 정답에서의 토큰 자체를 다음 토큰으로 넣는 것입니다. 당연히 이것을 항상 적용시키는 것이 아닌 어떠한 확률 때만 적용하게 되며 그 확률 역시 하이퍼 파라미터로 저희가 직접 설정해주면 됩니다. 이것을 항상 적용시키게 된다면 학습이 빠른 대신에 Bias 에 더 Expose 가 되기에 적절하게 세팅을 해주면 됩니다.

4. Vocabulary

2번 칸에서는 토크나이징에 관한 얘기를 했습니다. 그렇다면 Vocabulary 는 무엇일까요? 저희는 토크나이징을 시키고 나서 데이터 셋 내에서 사용된 모든 토큰들을 가지고 있게 됩니다. 해당 토큰들을 가지고 다시 새로운 단어들에 예측하며 진행을 시켜야 하지만 그러기 위해서는 토큰들을 숫자로 표현을 해야 합니다. 다시 말해, 각각의 토큰들을 숫자로 표현해야 하며 데이터 셋에 있는 토큰들도 숫자로 변형시켜서 표현해야 합니다.

이해를 돕기 위해 2번 칸의 예시를 다시 갖고 오겠습니다.

[“Hello”], [“,”], [“my”], [“name”], [“is”], [“Jae”], [“Woo”], [“Shin”],

문장을 토큰들로 변형은 시켰지만 이것 자체를 모델에는 넣을 수 없으므로 Vocabulary 를 구성했으며 Vocabulary 는 다음과 같이 구성이 된다고 치겠습니다.

[“Hello” : 1], [“,” : 2], [“my” : 3], [“name” : 4], [“is”: 5], [“Jae” : 6], [“Woo” : 7], [“Shin”: 8]

이것을 다시 토큰화 된 문장에 매칭을 시키게 된다면 다음과 같은 벡터 형태로 문장을 표현할 수가 있습니다.

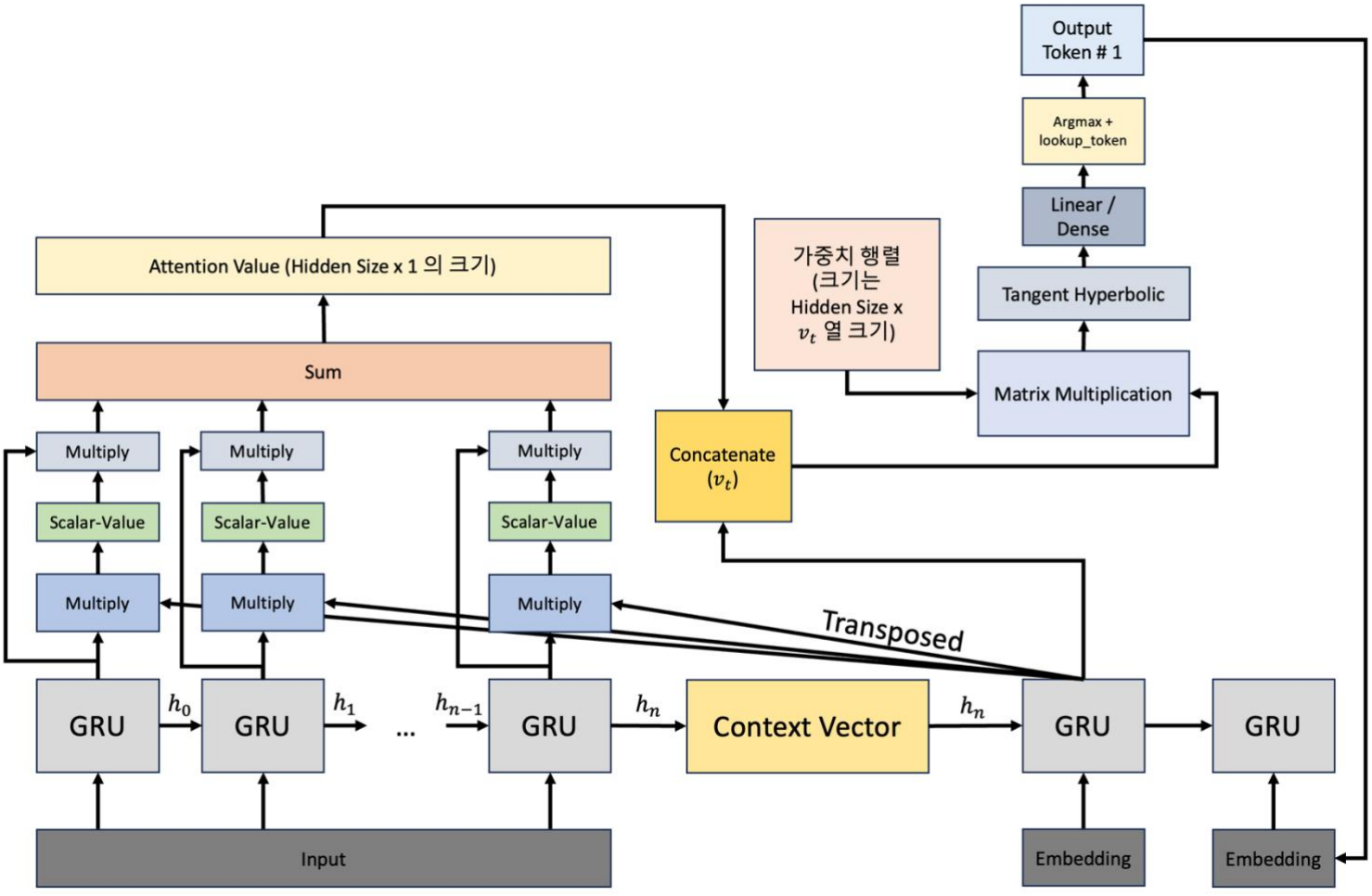
[1, 2, 3, 4, 5, 6, 7, 8]

위의 벡터를 텐서로 바꿔주면 모델에 바로 활용이 가능합니다. 과제에서는 Vocabulary 구성하는 단계는 주어졌으며 매칭 시키는 단계만 빈칸으로 되어 있습니다. 해당 칸에 관한 힌트를 드리자면 **vocab.lookup_indices()** 함수를 활용하면 됩니다. 이 함수가 해주는 것은 위에서 말했던 것을 그대로 해주는 것이며, **en_vocab.lookup_indices(토큰)** 의 형태로 활용이 가능합니다.

5. Attention

Attention 을 적용시키는 것 역시 과제에 포함이 되어 있습니다. 아래 그림과 Transformers 세션을 참조하셔서 Attention 구간이 구체적으로 어떻게 구현이 되어야 하는지 참조를 해주시길 바랍니다.

아래 그림은 오로지 한개의 Time Step, 즉 한개의 토큰에 관한 것만 하고 있으며, 이것을 Decoder 내에서 모든 토큰들에 한에서 진행을 해야 합니다. 코드 상에서는 힌트들을 살피면서 구현을 해주시길 바랍니다.



참고 자료 :

https://wikidocs.net/22893

https://luna-b.tistory.com/29