





(인공지능과빅데이터) 연습문제_12


1. RNN 계열 인공지능망인 LSTM으로 seq2seq 모델을 구현하였을 때, 입력 시퀀스의 함축적인 정보가 담긴 최종 출력 히든 상태를 부르는 이름으로 옳은 것은?
 1. 어텐션 벡터 (Attention Vector)
 2. 컨텍스트 벡터 (Context Vector) ✓
 3. 셀 상태 (Cell State)
 4. 게이트 벡터 (Gate Vector)
 5. 임베딩 벡터 (Embedding Vector)
2. RNN 계열 Seq2Seq 모델에서 입력 시퀀스가 길어질 때 발생하는 정보 손실 및 성능 저하 문제를 보완하기 위해 고안된 방법론으로 가장 적절한 것은?
 1. 백프로파게이션 스루 타임 (Backpropagation Through Time)
 2. 렐루 활성화 함수 (ReLU Activation Function)
 3. 어텐션 메커니즘 (Attention Mechanism) ✓
 4. 드롭아웃 (Dropout)
 5. 배치 정규화 (Batch Normalization)
3. 어텐션 메커니즘에서 사용되는 K (Key), V (Value), Q (Query)에 대한 설명으로 옳은 것은?
 1. Q(Query)는 디코더의 현재 상태를 나타내며, K(Key)와 V(Value)는 인코더의 모든 은닉 상태를 의미한다.
 2. K(Key)는 디코더가 찾고자 하는 정보의 "질문"을 나타내며, V(Value)는 인코더의 각 은닉 상태가 가지는 "중요도"를 나타낸다.
 3. Q(Query)는 인코더의 모든 은닉 상태를 나타내며, K(Key)는 디코더의 현재 상태에 해당한다.
 4. V(Value)는 K와 Q를 통해 계산된 어텐션 스코어를 나타내며, 최종 컨텍스트 벡터 계산에 사용된다.
 5. Q(Query)는 어텐션 계산 시 대상이 되는 정보의 "질문" 역할을 하며, K(Key)는 이 질문에 매칭될 수 있는 "키"들을, V(Value)는 각 "키"에 해당하는 "값"들을 나타낸다. ✓
4. 어텐션 메커니즘에서 사용되는 주요 개념인 어텐션 스코어(Attention Score)와 어텐션 값(Attention Value)에 대한 설명으로 옳은 것은?
 1. 어텐션 스코어는 디코더의 현재 상태와 인코더의 모든 은닉 상태를 직접 더하여 계산된 값이며, 어텐션 값은 이 스코어들의 평균을 의미한다.
 2. 어텐션 스코어는 인코더와 디코더 간의 유사도를 측정하여 각 입력 요소의 중요도를 나타내는 정규화되지 않은 값이며, 어텐션 값은 이 스코어들

을 소프트맥스 함수를 통해 확률 분포로 변환한 가중치를 의미한다.


3. 어텐션 스코어는 인코더의 각 입력 토큰에 대한 중요도를 나타내는 고정된 값이며, 어텐션 값은 디코더의 출력 예측에 사용되는 최종 임베딩 벡터이다.
 4. 어텐션 스코어는 디코더의 현재 쿼리(Query)와 인코더의 각 키(Key) 간의 유사도를 측정하여 얻는 값이며, 어텐션 값은 이 스코어를 바탕으로 계산된 가중 평균을 통해 얻어진 컨텍스트 벡터이다. 
 5. 어텐션 값은 어텐션 스코어들을 직접 곱하여 얻어진 결과이며, 이는 모델의 다음 단어 예측에 그대로 사용된다.
5. 루엡(Luong)의 닷 프로덕트(Dot-Product) 어텐션 기법에서 어텐션 값(Attention Value)을 계산하는 과정에 대한 설명으로 가장 적절한 것은?
1. 디코더의 현재 은닉 상태와 인코더의 모든 은닉 상태 간의 내적(dot product)을 통해 어텐션 스코어를 계산하고, 이 스코어들을 소프트맥스(softmax) 함수에 통과시켜 가중치를 얻은 다음, 이 가중치와 인코더의 은닉 상태들을 가중 합(weighted sum)하여 어텐션 값을 얻는다. 
 2. 디코더의 현재 은닉 상태와 인코더의 모든 은닉 상태를 단순히 연결(concatenate)한 후, 이를 MLP(다층 퍼셉트론)에 통과시켜 어텐션 스코어를 계산하고, 그 스코어들을 모두 더하여 어텐션 값을 얻는다.
 3. 인코더의 최종 은닉 상태만을 쿼리(Query)로 사용하여 디코더의 모든 은닉 상태와 내적을 수행한 후, 가장 높은 스코어를 가진 하나의 은닉 상태를 어텐션 값으로 선택한다.
 4. 어텐션 스코어 계산 시 소프트맥스 함수를 전혀 사용하지 않으며, 각 어텐션 스코어를 그대로 인코더의 은닉 상태와 곱하여 어텐션 값을 계산한다.
 5. 어텐션 스코어는 고정된 크기의 벡터로 미리 정의되어 있으며, 이 벡터와 디코더의 현재 은닉 상태를 내적하여 어텐션 값을 얻는다.
6. 바다나우(Bahdanau)의 컨캣(Concatenative/Additive) 어텐션 기법에서 어텐션 값(Attention Value)을 계산하는 과정에 대한 설명으로 가장 적절한 것은?
1. 디코더의 **이전 시점 은닉 상태**와 인코더의 모든 은닉 상태를 각각 내적(dot product)한 후 합산하여 어텐션 스코어를 계산하고, 이를 소프트맥스(softmax) 함수에 통과시켜 가중치를 얻는다.
 2. 디코더의 이전 시점 은닉 상태와 인코더의 각 은닉 상태를 연결(concatenate)한 후, 이를 피드포워드 신경망(Feed-Forward Network)에 통과시켜 어텐션 스코어를 계산한다. 이 스코어들을 소프트맥스(softmax) 함수에 통과시켜 가중치를 얻은 다음, 이 가중치와 인코더의 은닉 상태들을 가중 합(weighted sum)하여 어텐션 값을 얻는다. 
 3. 인코더의 최종 은닉 상태와 디코더의 현재 은닉 상태를 직접 더하여 어텐션 스코어를 얻고, 이를 어텐션 값으로 사용한다. 소프트맥스 함수는 이 과정에서 사용되지 않는다.

4. 어텐션 스코어는 디코더의 현재 예측 단어만을 기반으로 계산되며, 어텐션 값은 이 단어의 임베딩 벡터와 동일하다.
 5. 가장 높은 유사도를 가진 인코더의 은닉 상태 하나를 선택하여, 이를 어텐션 값으로 간주한다. 이때 소프트맥스 함수는 선택 과정에 아무런 영향을 주지 않는다.
7. 인공신경망 기반의 시퀀스-투-시퀀스(Seq2Seq) 모델에서 어텐션(Attention) 메커니즘이 도입되면서 얻게 된 가장 중요한 이점은 무엇인가?
1. 모델의 학습 시간을 획기적으로 단축시켜 대규모 데이터셋에서도 빠른 학습이 가능해졌다.
 2. 입력 시퀀스의 길이에 관계없이 항상 단일 고정 길이의 컨텍스트 벡터만을 사용하여 정보 손실을 최소화하였다.
 3. RNN 계열 모델에서 발생하는 기울기 소실(Vanishing Gradient) 문제를 완전히 해결하여 매우 깊은 시퀀스 모델의 학습을 가능하게 했다.
 4. 입력 시퀀스의 모든 부분을 동일한 중요도로 고려하여, 특정 정보에 대한 편향 없이 전체 정보를 균등하게 활용할 수 있게 되었다.
 5. 입력 시퀀스가 길어질 때 발생하는 정보 손실 문제(bottleneck)를 완화하고, 디코더가 매 시점 출력 단어를 생성할 때 입력 시퀀스의 관련성 높은 부분에 집중할 수 있도록 하여 성능을 크게 향상시켰다. 
8. 어텐션 메커니즘을 기반으로 하여 등장한 트랜스포머(Transformer) 모델과 그 내부에서 사용되는 주요 어텐션 기법들에 대한 설명으로 가장 적절하지 않은 것은?
1. 트랜스포머는 스케일드 닷 프로덕트 어텐션(Scaled Dot-Product Attention)이라는 단일 핵심 어텐션 알고리즘을 기반으로 하며, 나머지 형태들은 이의 응용 또는 확장이다.
 2. 인코더 블록의 셀프 어텐션(Self-Attention)은 입력 시퀀스 내의 각 단어가 시퀀스 내의 다른 모든 단어들과의 관계를 파악하는 데 사용된다.
 3. 디코더 블록의 마스킹된 셀프 어텐션(Masked Self-Attention)은 현재 시점의 단어가 미래 시점의 단어들을 참조하지 못하도록 하여 순차적인 예측이 가능하게 한다.
 4. 디코더 블록의 인코더-디코더 어텐션(Encoder-Decoder Attention)은 디코더의 현재 상태(쿼리)가 인코더의 최종 출력(키, 값)으로부터 관련 정보를 추출하는 데 사용된다.
 5. 멀티 헤드 어텐션(Multi-Head Attention)은 완전히 서로 다른 3가지의 독립적인 어텐션 계산 알고리즘을 병렬로 사용하여 모델의 표현력을 높인다. 
9. 트랜스포머(Transformer) 모델에서 사용되는 멀티 헤드 어텐션(Multi-Head Attention)에 대한 설명으로 가장 적절한 것은?
1. 단일 어텐션 메커니즘을 여러 번 순차적으로 실행하여, 각 단계에서 계산된 어텐션 결과를 다음 단계로 전달하는 방식이다.
 2. 입력 시퀀스의 길이와 관계없이 항상 고정된 개수의 헤드를 사용하여, 계산 효율성을 높이는 데 주력한다.
 3. 각 어텐션 헤드는 Q, K, V를 동일한 선형 변환 없이 직접 사용하여 병


렬 계산의 복잡성을 줄인다.

4. 여러 개의 어텐션 서브 유닛(헤드)을 병렬로 실행하여, 입력 시퀀스의 다양한 관점과 관계를 동시에 학습하고, 이를 통해 모델의 표현력을 높이는 방식이다. 
5. 주로 디코더 블록에서만 사용되며, 인코더 블록의 셀프 어텐션에는 적용되지 않는다.


10. 트랜스포머로부터 파생된 대표적인 대규모 언어 모델인 BERT와 GPT에 대한 설명 및 비교로 가장 적절하지 않은 것은?

1. BERT(Bidirectional Encoder Representations from Transformers)는 인코더(Encoder)만으로 구성된 트랜스포머 구조를 가지며, GPT(Generative Pre-trained Transformer)는 디코더(Decoder)만으로 구성된 트랜스포머 구조를 가진다.
2. BERT는 사전 학습 시 문장의 중간을 마스킹하여 양방향 문맥을 학습하는데 중점을 두며, GPT는 이전 단어들을 기반으로 다음 단어를 예측하는 단방향(자기 회귀적) 방식으로 학습한다.
3. BERT는 주로 자연어 이해(NLU) 태스크(예: 문장 분류, 질의응답)에 강점을 보이며, GPT는 주로 자연어 생성(NLG) 태스크(예: 텍스트 생성, 요약)에 강점을 보인다.
4. 두 모델 모두 방대한 양의 비지도 학습 데이터로 사전 학습된 후, 특정 태스크에 파인튜닝되거나 프롬프트(Prompt) 기반으로 활용될 수 있다.
5. BERT와 GPT 모두 텍스트 생성(Text Generation)에 최적화된 모델이며, 주로 자기 회귀적(Auto-regressive) 방식으로 다음 단어를 예측하도록 사전 학습된다. 

11. PyTorch에서 모델의 가중치를 저장하는 가장 일반적이고 권장되는 방법은 무엇입니까?

1. `torch.save(model.state_dict(), 'model_weights.pth')` 
2. `torch.save(model, 'full_model.pth')`
3. `model.save_weights('model_weights.h5')`
4. `torch.dump(model.state_dict(), 'model_weights.pkl')`
5. `model.checkpoint('model_weights.pt')`

12. PyTorch에서 이전에 `torch.save(model.state_dict(), 'model_weights.pth')` 로 저장된 모델의 가중치를 불러와서 모델에 로드하는 올바른 방법은 무엇입니까?

```
# 1)   
model = MyModel()  
model.load_state_dict(torch.load('model_weights.pth'))  
# 2)  
model = torch.load('model_weights.pth')  
# 3)  
model = MyModel()  
torch.load(model.state_dict(), 'model_weights.pth')  
# 4)
```

```
model = MyModel()  
model.load_weights('model_weights.pth')  
# 5)  
model = MyModel()  
model.state_dict().load_from_file('model_weights.pth')
```