Sequential Recommendation Using Transformer Architecture

- SASRec, BERT4Rec -

세 줄 요약

- SASRec (Self-Attentive Sequential Recommendation)
  - item embedding, positional embedding 사용
  - Transformer의 decoder layer 사용 (encoder-decoder block 제외)
  - 모든 sequence에서 loss 계산

- BERT4Rec
  - item embedding, positional embedding 사용
  - Transformer의 encoder layer 사용
  - Sequence 중간중간 masking한 item에 대해서만 loss 계산 (Masked Language Model)

Encoder Layer, Decoder Layer

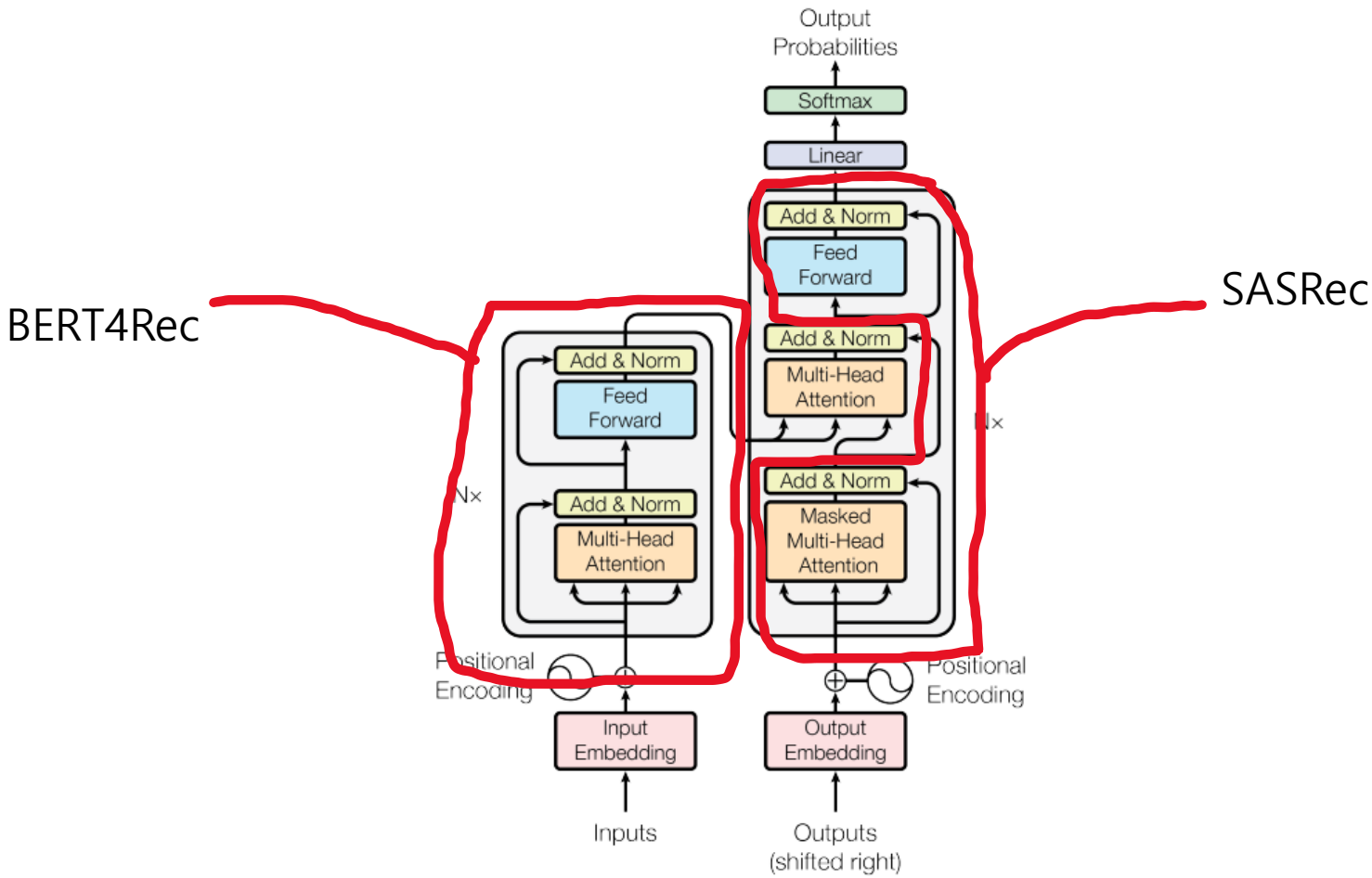
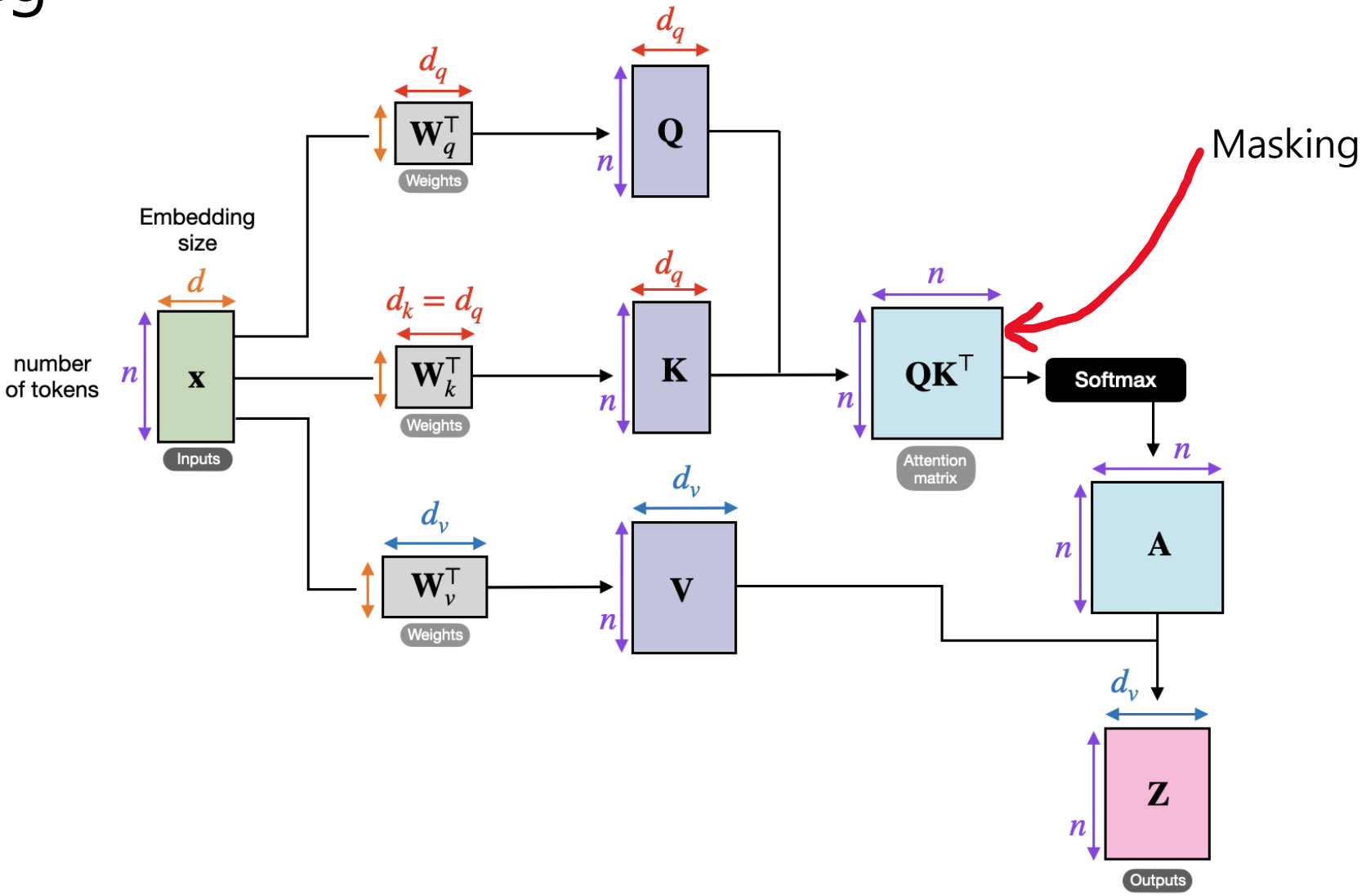


Figure 1: The Transformer - model architecture.

Masking

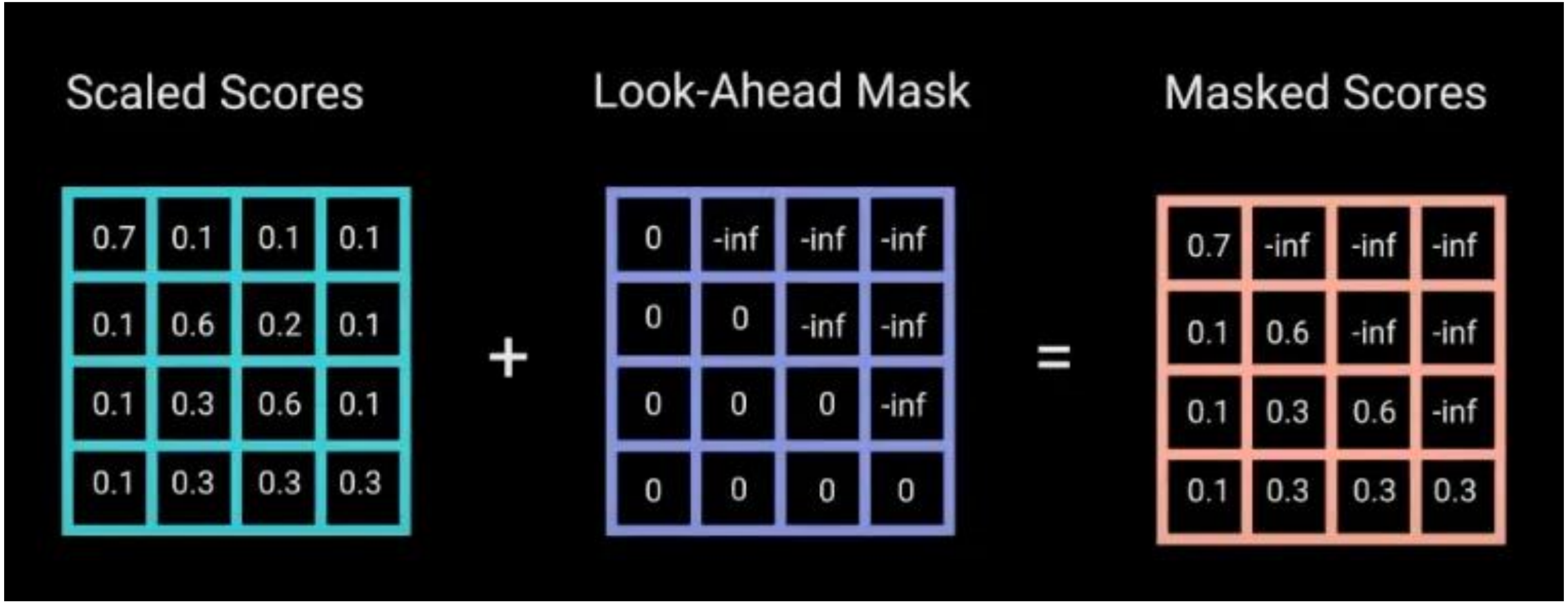

Masking
Embedding size
number of tokens
Inputs
Weights
Weights
Weights
Attention matrix
Softmax
Outputs

Masking

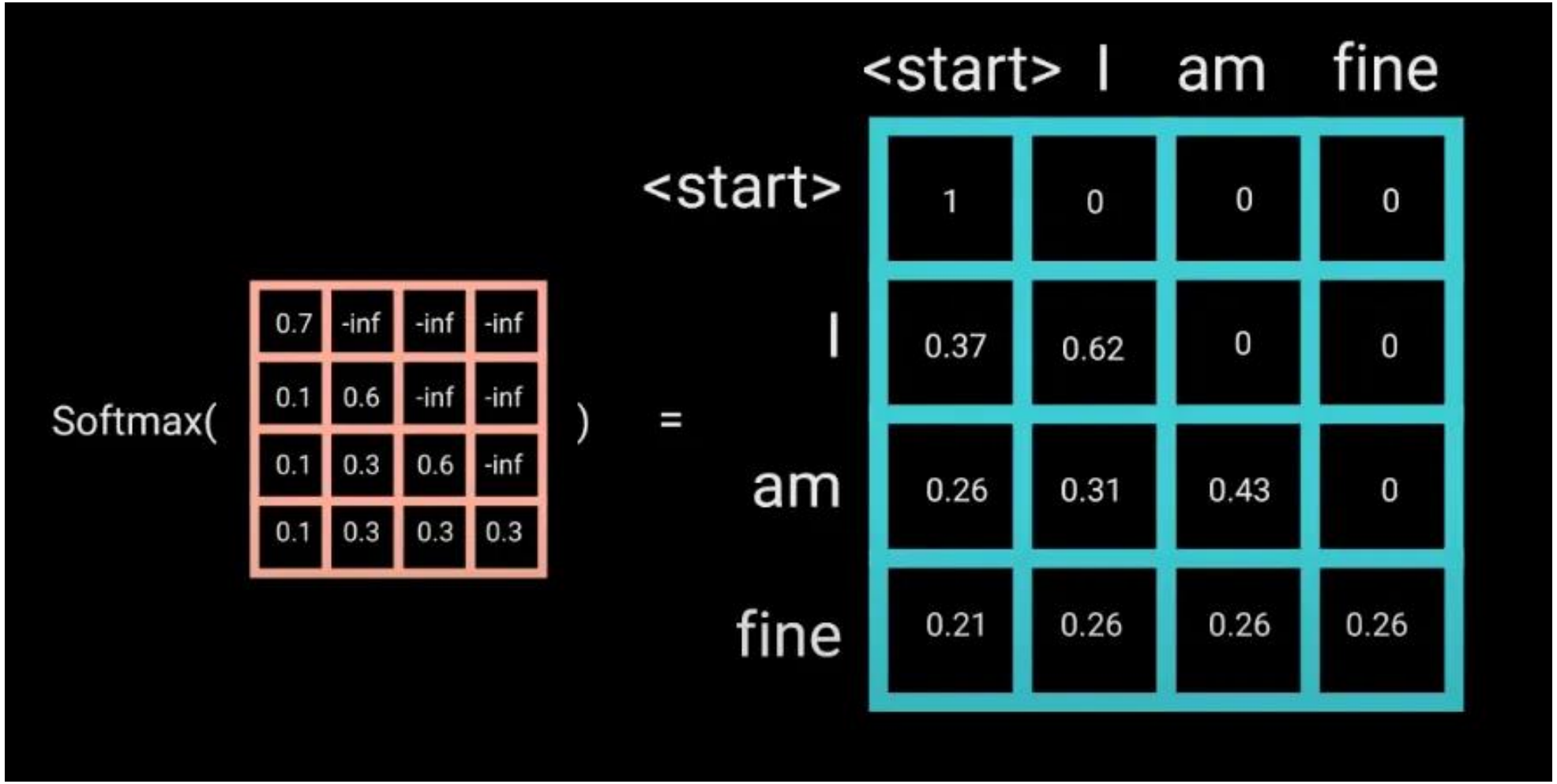
- Zero-padding mask
  - Sequence 길이가 최대 길이보다 짧을 때 길이를 맞추기 위해 사용
  - 앞에 하나 뒤에 하나 상관 없음

- Look-ahead mask
  - Transformer는 전체 sequence의 정보가 한 번에 입력
  - 뒤쪽의 정보를 사용해 앞쪽의 attention을 계산하는 것을 방지

Masking


Scaled Scores
Look-Ahead Mask
Masked Scores
0.7 0.1 0.1 0.1
0.1 0.6 0.2 0.1
0.1 0.3 0.6 0.1
0.1 0.3 0.3 0.3
+
0 -inf -inf -inf
0 0 -inf -inf
0 0 0 -inf
0 0 0 0
=
0.7 -inf -inf -inf
0.1 0.6 -inf -inf
0.1 0.3 0.6 -inf
0.1 0.3 0.3 0.3

Masking


Softmax(
0.7 -inf -inf -inf
0.1 0.6 -inf -inf
0.1 0.3 0.6 -inf
0.1 0.3 0.3 0.3
)
=
<start> I am fine
<start> 1 0 0 0
I 0.37 0.62 0 0
am 0.26 0.31 0.43 0
fine 0.21 0.26 0.26 0.26

SASRec - Unidirectional Model

- Interaction sequence의 특성 고려
  - 따라서 단방향 모델(left-to-right unidirectional model)
  - Look-ahead mask 사용

Causality: Due to the nature of sequences, the model should consider only the first t items when predicting the $(t+1)$-st item. However, the t-th output of the self-attention layer ($\mathbf{S}_t$) contains embeddings of subsequent items, which makes the model ill-posed. Hence, we modify the attention by forbidding all links between $\mathbf{Q}_i$ and $\mathbf{K}_j$ $(j > i)$.

BERT4Rec - Bidirectional Model

- User-Item interaction sequence의 특성 고려
  - 시계열이나 언어 데이터는 순서가 정해져 있음
    - Ex) 나는 배고프다 (O) / 배고프다 나는 (X)
  - 반면 아이템 구매 이력에서는 순서가 정해져 있다고 볼 수 없음
    - Ex) 모니터 – 그래픽 카드
  - 양방향 모델(bidirectional model)인 BERT
  - Look-ahead mask 사용 X

BERT4Rec - Bidirectional Model

Another limitation is that previous unidirectional models are originally introduced for sequential data with natural order, *e.g.*, text and time series data. They often assume a rigidly ordered sequence over data which is not always true for user behaviors in real-world applications. In fact, the choices of items in a user's historical interactions may not follow a rigid order assumption [18, 54] due to various unobservable external factors [5]. In such a situation, it is crucial to incorporate context from both directions in user behavior sequence modeling.

모델 구조 – 전처리

- Train/Valid/Test split (leave-one-out)
  - 유저 sequence의 마지막 데이터 -> Test
  - 마지막에서 두번째 데이터 -> Valid
  - 나머지 데이터 -> Train
  - Sequnece 길이가 최대 길이보다 길면 절삭, 짧으면 padding 추가

모델 구조 – 전처리 (BERT4Rec)

- Masked Language Model (빈칸 채우기)
  - Ex) BERT4Rec is used for sequential recommendation.
    -> BERT4Rec is [mask] for [mask] recommendation.
  - 중간중간 masking된 문장이 주어졌을 때 mask에 올 단어 예측

- Masking 규칙
  - 80%는 평범하게 mask, 10%는 랜덤 단어로 대체, 10%는 원래 단어
  - 원래의 BERT는 fine-tuning 용도
  - Fine-tuning단계에서는 mask 토큰이 없기 때문

모델 구조 – 전처리 (BERT4Rec)

- 정답 레이블 필요
  - Ex) 유저 sequence : [4, 7, 3, 6, 1, 9, 2]
  - Train, Valid, Test
    [4, 7, 3, 6, 1], [9], [2]
  - Mask, zero-padding 추가
    [0, 0, 0, 4, mask, 3, 6, mask]
  - 정답 레이블
    [0, 0, 0, 0, 7, 0, 0, 1]

- 일부 sequence는 마지막 item에만 masking
  - Valid, inference때 input과 동일하도록
  - 미션 코드에는 없음

모델 구조 – Embedding Layer

- Item Embedding
  - 아이템의 index에 따라서 d차원의 벡터로 임베딩
  - SASRec은 n_items+1개 (zero-padding)
  - BERT4Rec은 n_items+2개 (zero-padding, mask token)

- Positional Embedding
  - Sequence의 step에 따라서 d차원의 벡터로 임베딩
  - Item + positional로 최종 임베딩

- (batch_size, max_len) -> (batch_size, max_len, d)

모델 구조 – Transformer Layer

- Multi-head Attention
  - Mask argument 넣어서 self-attention
    - SASRec은 zero-padding, look-ahead mask 둘 다 사용
    - BERT4Rec은 zero-padding mask만 사용
  - Dropout, residual connection, layer normalization 적용

- (batch_size, max_len, d) -> (batch_size, max_len, d)

모델 구조 – Transformer Layer

- Position-wise Feed Forward Network
  - Linear -> Activation -> Linear로 이루어짐
    - SASRec은 activation으로 RELU 사용
    - BERT4Rec은 GELU 사용
  - Dropout, residual connection, layer normalization 적용

- (batch_size, max_len, d) -> (batch_size, max_len, d)

모델 구조 – Output Layer

- Shared Item Embedding
  - 두 모델 다 item embedding을 output layer에 다시 사용
  - 학습 파라미터 개수 감소하는 효과
  - h_t : transformer layer를 통과한 t 시점의 output vector
  - E : item embedding matrix

모델 구조 – Output Layer

- SASRec
  - 간단히 내적해 prediction 생성
  - $Softmax(h_t E^T)$

- Bert4Rec
  - 2개의 linear layer 통과
  - $GELU(h_t W_p + b_p)E^T + b_o$
  - 미션 코드에서는 E^T 사용하지 않고 $h_t W_p + b_p$ 에서 바로 predict

- (batch_size, max_len, d) -> (batch_size, max_len, n_items+1)
  (E^T 를 사용하는 BERT4Rec의 경우엔 n_items+2)

Loss Function

- SASRec
  - 모든 positive item에 대해 negative item 하나씩 sampling
  - Positive item의 스코어는 최대화, negative item의 스코어는 최소화하도록 binary cross entropy loss 사용
  - Ex) positive score, negative score = 0.7, 0.2
  - Criterion = nn.BCEWithLogitsLoss()
  - Loss = criterion(0.7, 1) + criterion(0.2, 0)

- BERT4Rec
  - Multi-class classification에 적합한 negative log-likelihood 사용
  - Masking한 시점에 대해서만 loss 계산
  - Criterion = nn.CrossEntropyLoss(ignore_index=0)

Evaluation Metric

- 마지막 시점 모든 아이템의 Score(logit) 계산
  - BERT4Rec은 중간 mask 없이 모든 train 데이터 input으로 활용
    - Paper에는 특별히 언급 없는데 미션 코드에서 이렇게 구현
  - 마지막 시점에 예측할 위치를 나타내는 mask token 추가
  - Ex) [4, 7, 3, 6, 1] -> [0, 0, 4, 7, 3, 6, 1, mask]

- Negative sampling
  - 하나의 valid item에 대해 100개의 negative item random sampling
  - Valid item 1개 + Negative sample 100개의 score 계산
  - Valid item의 score 순위가 k 이하인지 확인해 NDCG@k, HIT@k 계산

Inference For Top k Recommendation

- 유저별 candidate 생성
  - Interaction 없었던 모든 item 대상
- 마지막 시점 모든 아이템의 Score(logit) 계산
  - Valid data도 사용
- Candidate 중 가장 score가 높은 k개 선택

실험 가능한 요소들

- 임베딩 방식
  - Side information(장르, 개봉연도, 인기도 등) 활용 여부
  - 유저 임베딩 활용 여부
  - Positional 임베딩 활용 여부

- Valid data split
  - 한 개 말고 여러 개?
  - 여러 개 사용한다면 위치는?

실험 가능한 요소들

- Inference
  - Candidate 생성 방식
  - 마지막 시청이력이 2007년인 유저한테 2015년 개봉 영화를 추천해도 될까?
  - 유저의 특성을 고려해 candidate 만들 수 있을까?
- Negative Sampling
- Data Augmentaion
  - 길이가 긴 sequence 버리지 않고 train에 사용
  - (BERT4Rec) Masking 위치에 따라 data 엄청나게 생성 가능