Week #12

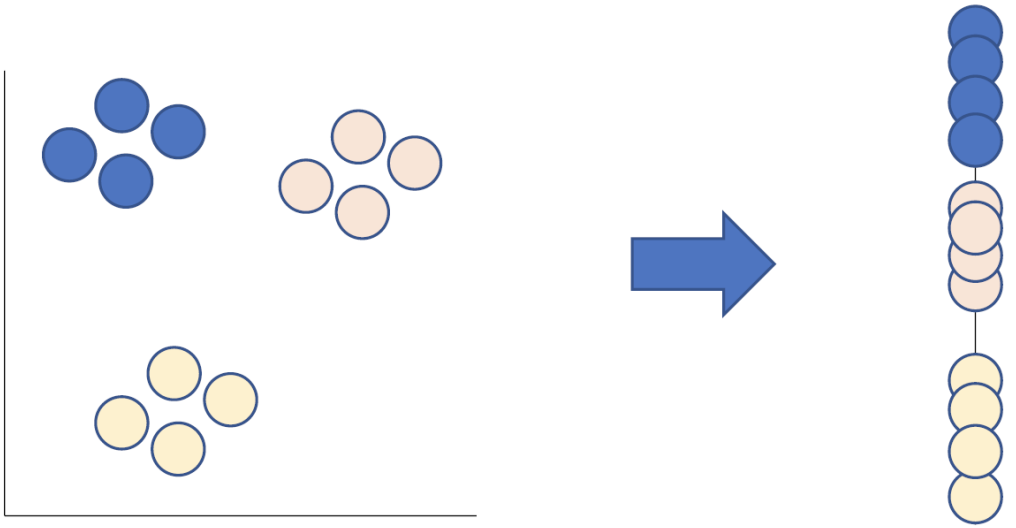
t-SNE & 자연어 처리: Pre-trained model 시대

May 26, 2022

Sources: Source: 파이썬 딥러닝 파이토치 (이경택,방성수, 안상준 지음), 정보문화사. Materials on the Internet including the Wikipedia/YouTube This book was modified and provided by Hyun-Chul Kim, Ph.D.

t-SNE (t-Stochastic Neighbor Embedding)

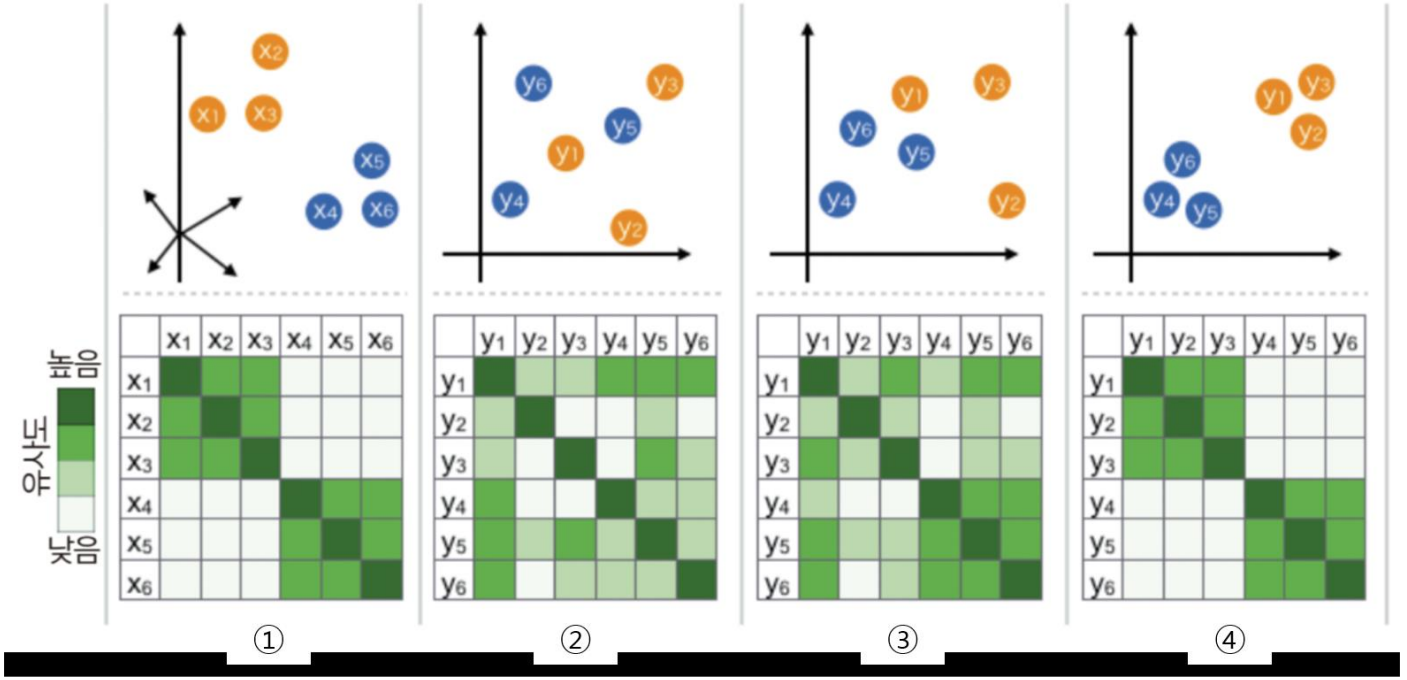
- t-SNE
  - 높은 차원의 복잡한 데이터를 2차원에 차원 축소하는 방법입니다. 낮은 차원 공간의 시각화에 주로 사용하며 차원 축소할 때는 비슷한 구조끼리 데이터를 정리한 상태이므로 데이터 구조를 이해하는 데 도움을 줍니다.
  - t-SNE는 **비선형적인 방법**의 차원 축소 방법이고 특히 고차원의 데이터 셋을 시각화하는 것에 성능이 좋습니다.



https://gaussian37.github.io/ml-concept-t_sne/

t-SNE

- t-SNE 학습과정
  - 아래 그림에서 x_i는 기존 데이터에 해당하며 고차원에 분포되어 있고 y_i는 t-SNE를 통하여 저차원으로 매핑된 데이터로 볼 수 있습니다. 예시에서 기존 데이터는 3차원이고 저차원은 2차원으로 사용되었습니다



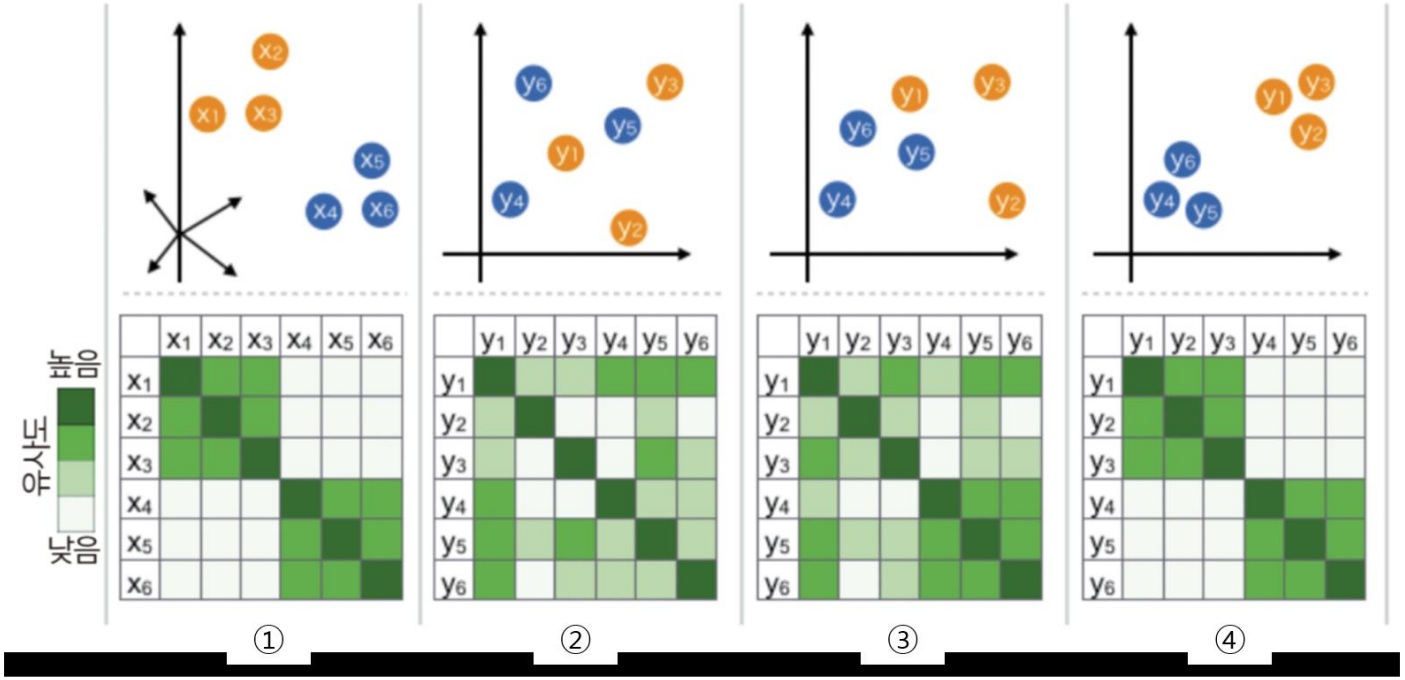
https://gaussian37.github.io/ml-concept-t_sne/

t-SNE

- t-SNE 학습과정

① 모든 i,j 쌍에 대하여 x_i,x_j의 유사도를 가우시안 분포를 이용하여 나타냅니다.

② x_i 와 같은 개수의 점 y_i를 낮은 차원 공간에 무작위로 배치하고, 모든 i,j쌍에 관하여 y_i,y_j의 유사도를 t-SNE를 이용하여 나타냅니다



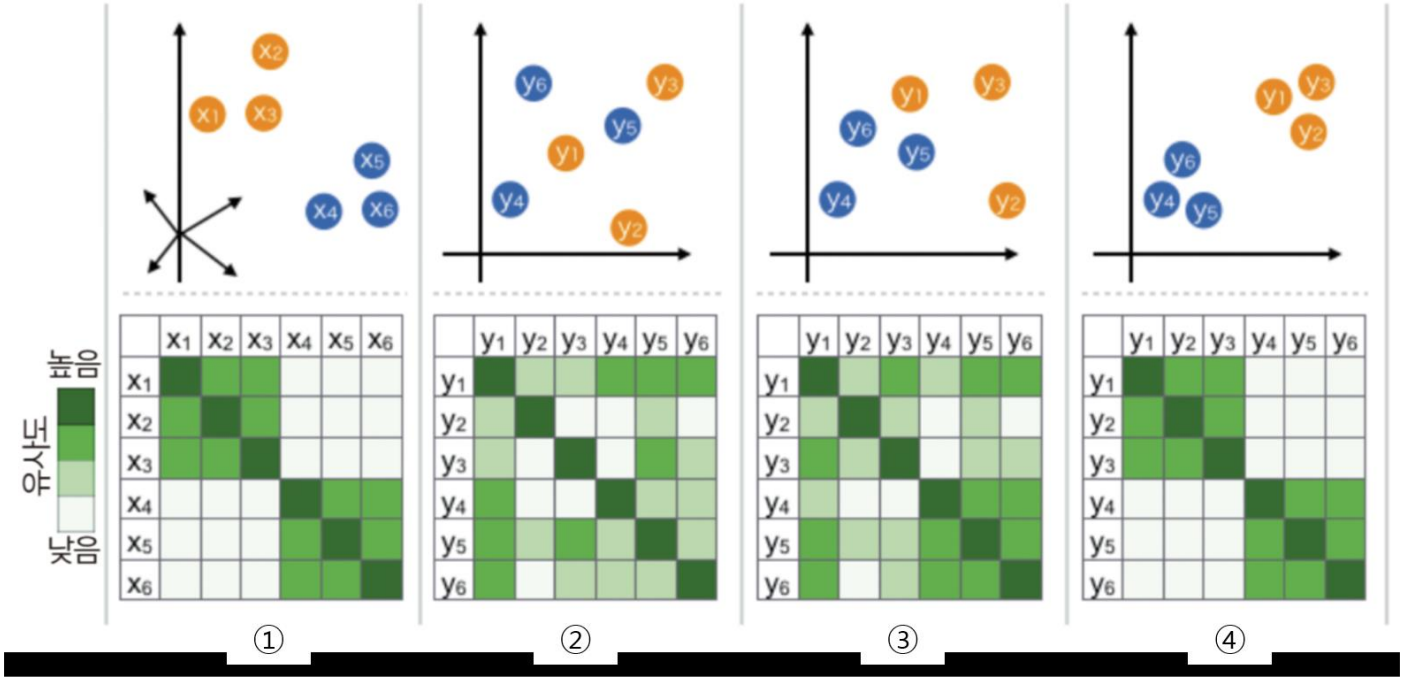
https://gaussian37.github.io/ml-concept-t_sne/

t-SNE

- t-SNE 학습과정

  ③ 앞의 ①, ②에서 정의한 유사도 분포가 가능하면 같아지도록 데이터 포인트 y_i를 갱신합니다.

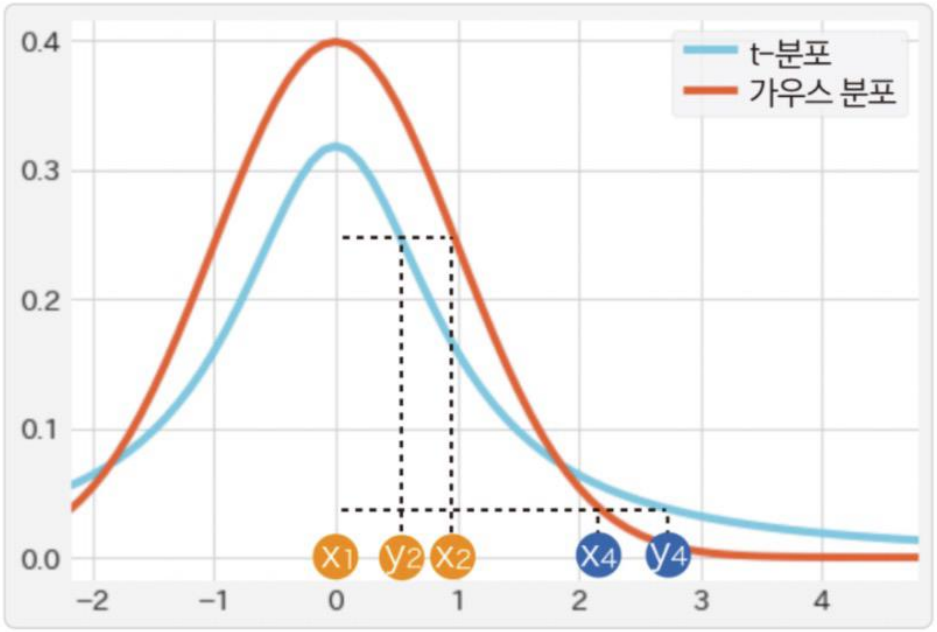
  ④ 수렴 조건까지 과정 ③을 반복합니다.



https://gaussian37.github.io/ml-concept-t_sne/

t-SNE

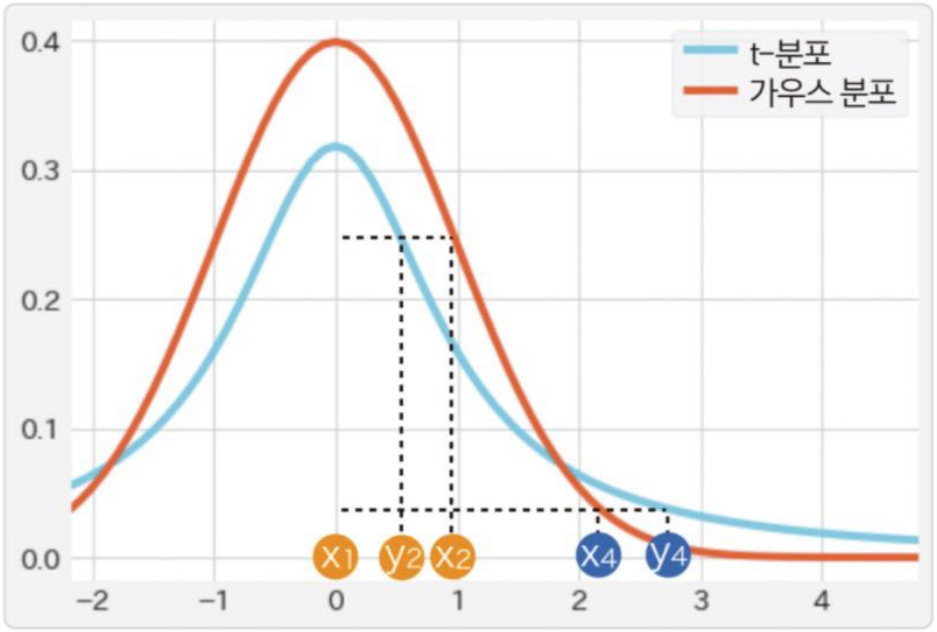
- t-SNE 학습과정
  - 이전 알고리즘에서 ①, ②의 유사도는 데이터 포인트들이 얼마나 비슷한 지 나타냅니다. 단순히 데이터 사이의 거리를 이용하는 것이 아니라 확률 분포를 이용합니다.
  - 아래 그래프는 가로축으로 거리, 세로축으로 유사도를 설정하여 t-분포와 가우시안 분포를 비교한 것입니다.



https://gaussian37.github.io/ml-concept-t_sne/

t-SNE

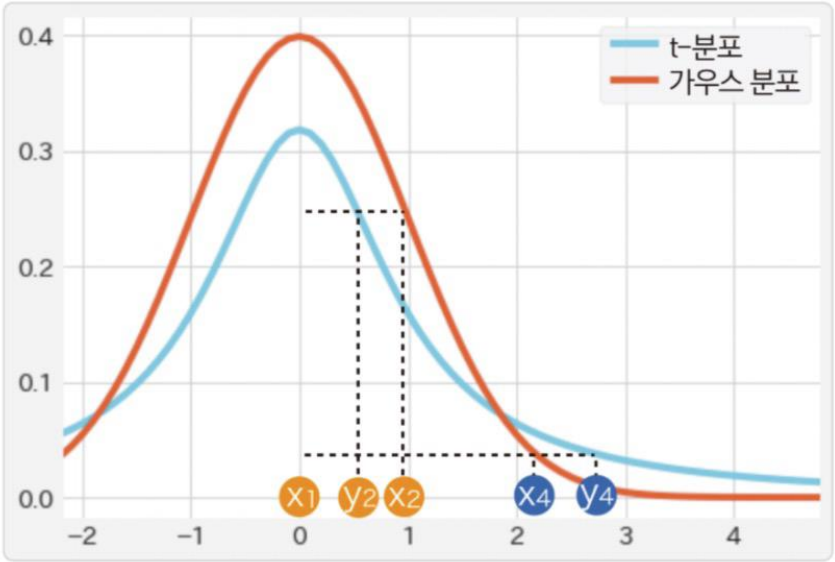
- t-SNE 학습과정
  - **데이터 사이의 거리가 가까울수록 유사도가 크고, 멀수록 유사도가 작아집니다.** 먼저 원본의 높은 차원 공간에서 정규 분포로 유사도를 계산하고 p_{ij}라는 분포로 나타냅니다. p_{ij}는 데이터 포인트 x_i, x_j 의 유사도를 나타냅니다.
  - 다음으로 x_i 에 대응하는 데이터 포인트 y_i 를 낮은 차원 공간에 무작위로 배치합니다. y_i에 관해서도 t-분포로 유사도를 나타내는 q_{ij}를 계산합니다



https://gaussian37.github.io/ml-concept-t_sne/

t-SNE

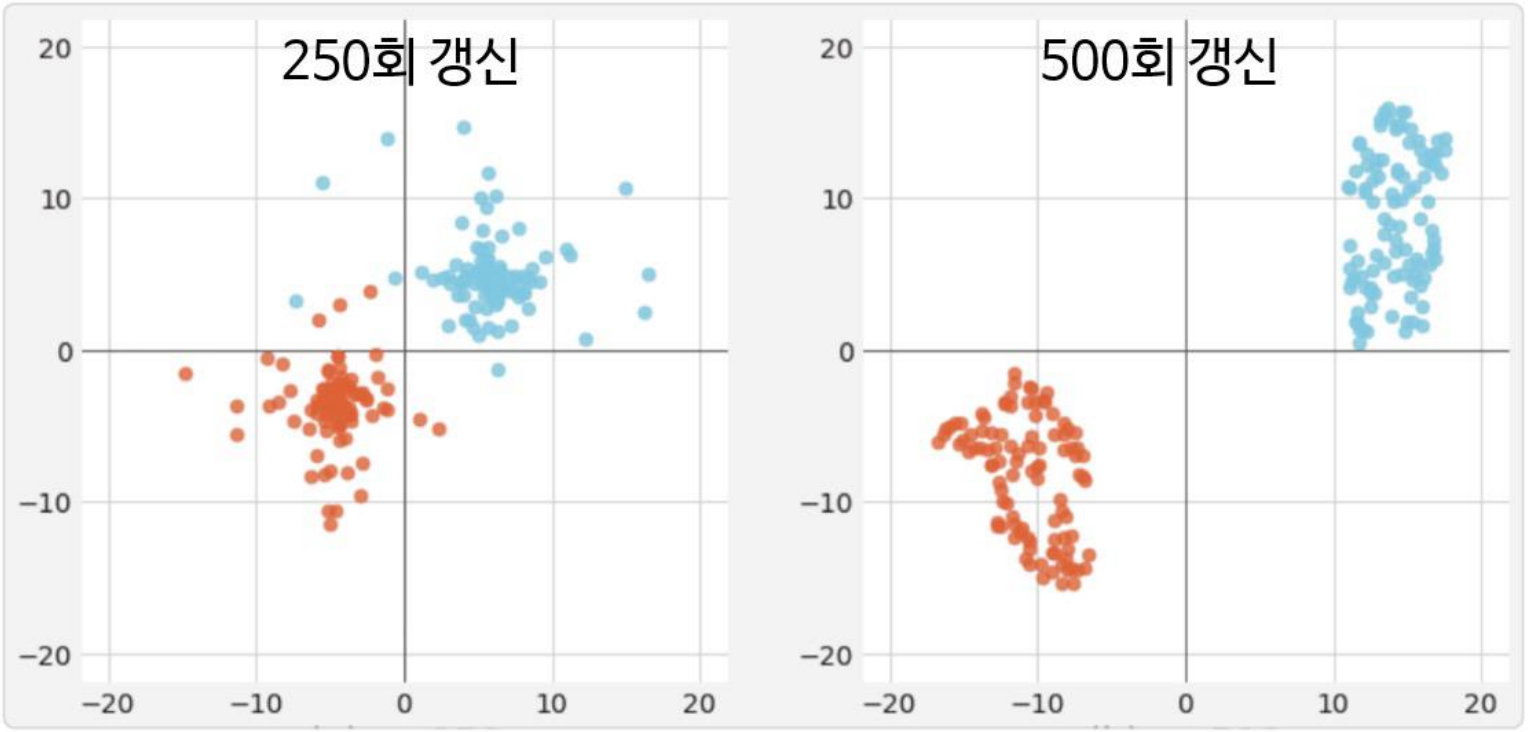
- t-SNE 학습과정
  - 여기서 p_{ij} 와 q_{ij} 를 계산하면 q_{ij} 를 p_{ij} 와 같은 분포가 되도록 데이터 포인트 y_i 를 갱신합니다. 이는 높은 차원 공간의 x_i 유사도 각각의 관계를 낮은 차원 공간의 y_i에서 재현하는 것입니다.
  - 이 때, 낮은 차원 공간에서 t-분포를 이용하므로, **유사도가 큰 상태**의 관계를 재현할 때는 낮은 차원 공간에서 **데이터 포인트를 더 가까이** 배치합니다. 반대로 **유사도가 작은 상태의 관계**를 재현할 때에는 **낮은 차원 공간에서 데이터 포인트를 더 멀리 배치**합니다.



https://gaussian37.github.io/ml-concept-t_sne/

t-SNE

- t-SNE 학습과정
  - 아래 그림은 t-SNE를 적용하였을 때, 데이터 포인트 y_i를 갱신하는 모습입니다. 왼쪽 그래프는 갱신 횟수가 250회이고 오른쪽 그래프는 갱신 횟수가 500회 입니다. **갱신 횟수가 늘수록 데이터 포인트의 차이를 명확하게 나타냅니다**.



https://gaussian37.github.io/ml-concept-t_sne/

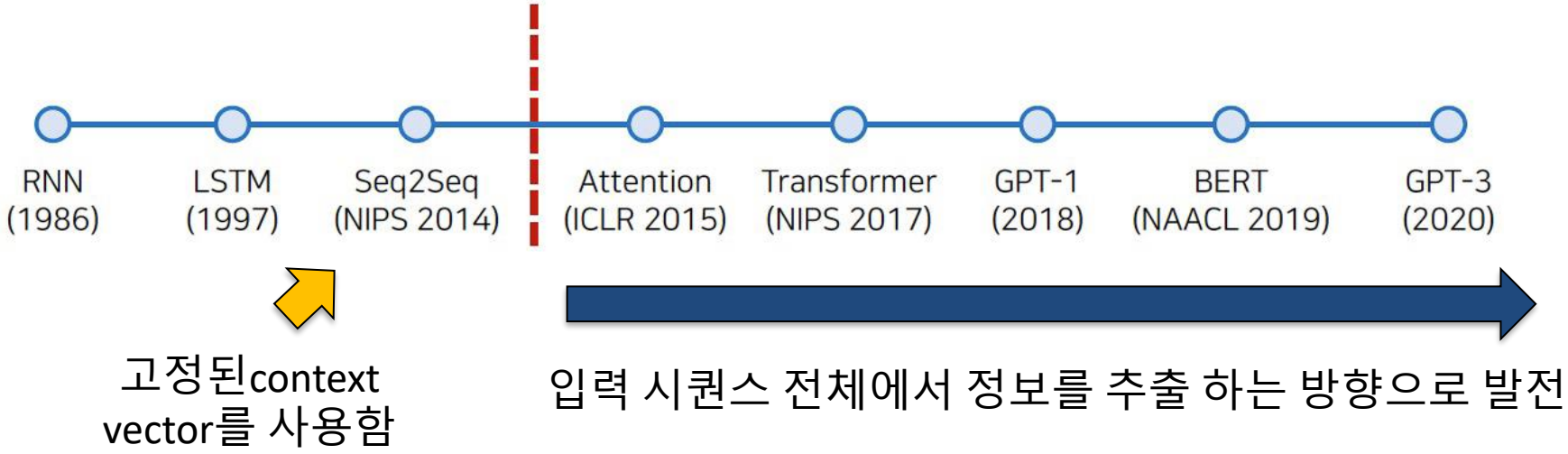
왜 t-분포를 사용?

- t-SNE
  - 낮은 차원에 임베딩 할 때, 정규 분포를 사용하지 않고 t-분포를 사용합니다. 그 이유는 앞에서 다루었듯이 t-분포가 heavy-tailed distribution임을 이용하기 위해서 입니다. 즉, t-분포는 **일반적인 정규분포보다 끝단의 값이 두터운 분포**를 가집니다.
  - t-SNE가 전제하는 확률 분포는 정규 분포이지만 정규 분포는 꼬리가 두텁지 않아서 *i*번째 개체에서 적당히 떨어져 있는 이웃 *j*와 아주 많이 떨어져 있는 이웃 k가 선택될 **확률이 크게 차이가 나지 않게** 됩니다.
  - 또한 **높은 차원 공간에서는 분포의 중심에서 먼 부분의 데이터 비중이 높기 때문에** 데이터 일부분의 정보를 고차원에서 유지하기가 어렵습니다.
  - 이러한 문제로 인하여 구분을 좀 더 잘하기 위해 정규 분포보다 **꼬리가 두터운 t분포**를 사용합니다.

https://www.youtube.com/watch?v=NEaUSP4YerM

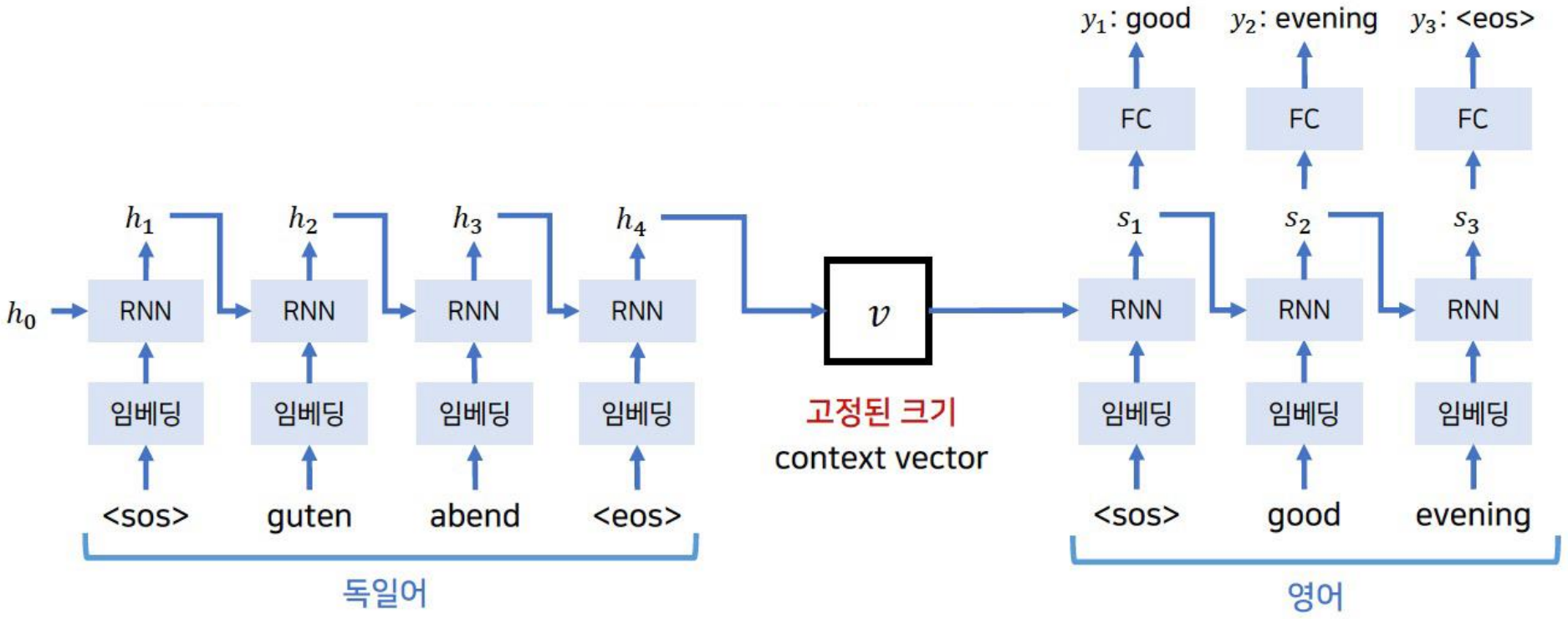
딥러닝 기반의 기계번역 발전 과정

- 2021년 기준으로 최신 고성능 모델들은 Transformer 아키텍처를 기반으로 하고 있습니다.
  - **GPT**: Transformer의 **디코더(Decoder)** 아키텍처를 활용
  - **BERT**: Transformer의 **인코더(Encoder)** 아키텍처를 활용



기존Seq2Seq 모델들의 한계점

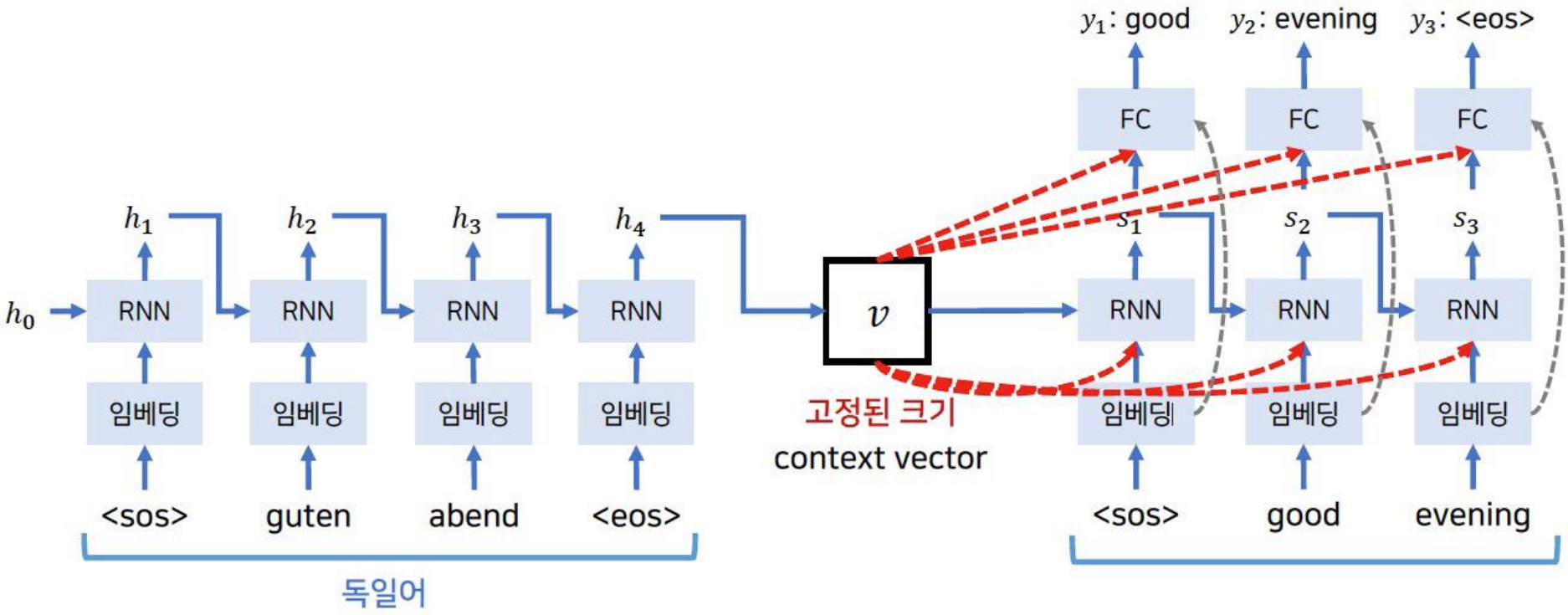
- **Context vector** v에 소스 문장의 정보를 압축 합니다.
  - **병목(bottleneck)이 발생**하여 성능 하락의 원인이 됩니다



Sequence to Sequence Learning with Neural Networks (NIPS 2014)

기존Seq2Seq 모델들의 한계점

- **디코더**가 context vector를 매번 참고 할 수도 있습니다.
  - 다만 여전히 소스 문장을 하나의 벡터에 압축해야 합니다



Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation (EMNLP 2014)

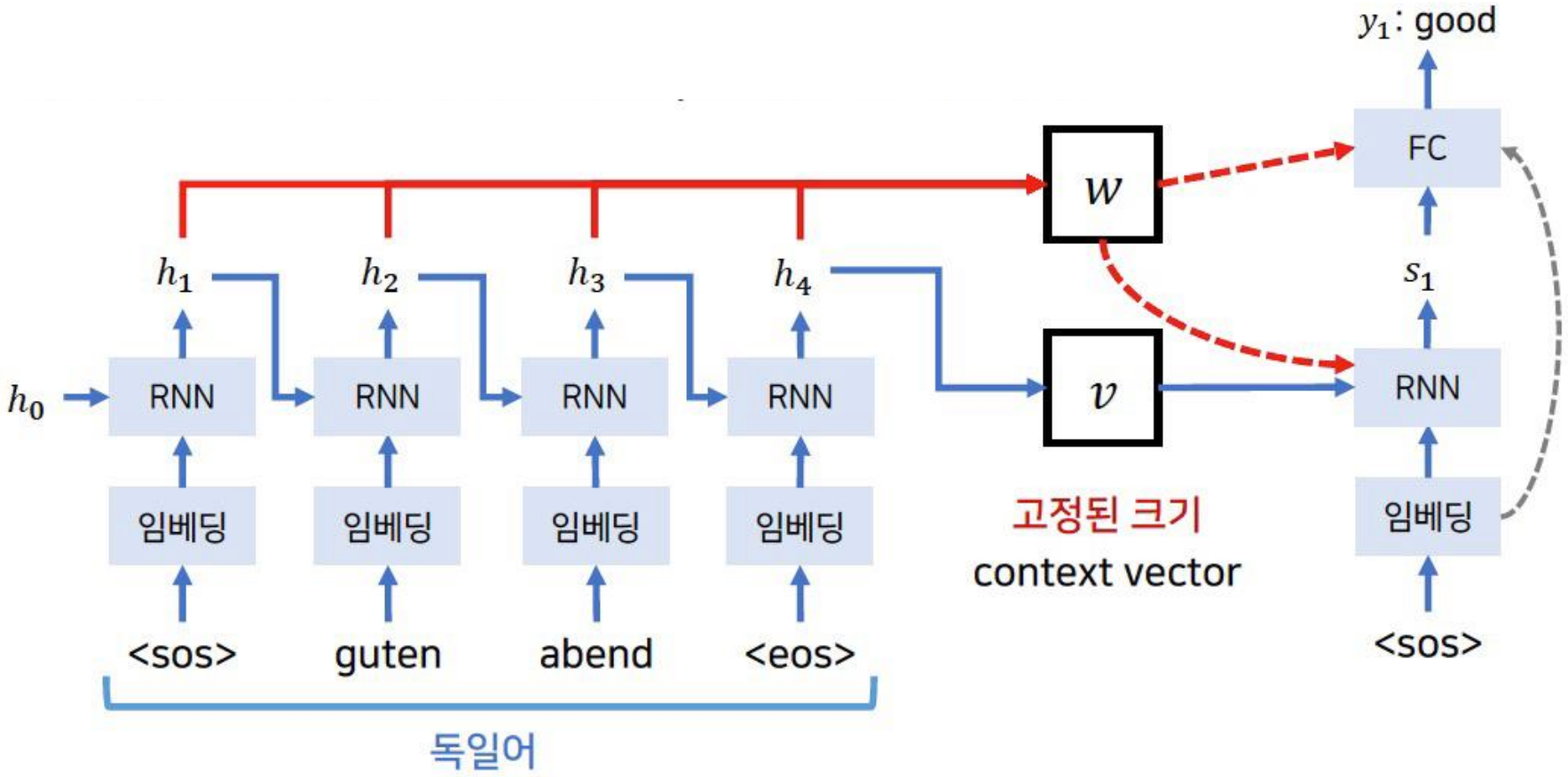
Seq2Seq with Attention

- [문제 상황]
  - 하나의 문맥 벡터가 소스 문장의 모든 정보를 가지고 있어야 하므로 성능이 저하 됩니다.

- [해결 방안]
  - 그렇다면 **매번 소스 문장에서의 출력 전부를 입력으로** 받으면 어떨까요?
    - 최신 GPU는 많은 메모리와 빠른 병렬처리를 지원합니다

Seq2Seq with Attention

- Seq2Seq모델에 **어텐션(attention)**매커니즘을 사용합니다.
  - 디코더는 **인코더의 모든 출력(outputs)을 참고**합니다



Pre-Trained Model의 시대

- Attention is all you need (The Transformer)
  - “Attention만 있으면 된다"는 제목 답게 이 논문에서 제시하고 있는 기계 번역 모델은 오직Attention 개념만 가지고 만들어져 있음.
  - Attention은 보통RNN류의 모델의 보조 정보 혹은 보조 장치로써의 좋은 성능을 보여옴
  - 하지만 이 모델은 RNN 계열의 Module을 아예 쓰지 않는 형태로 구조를 설계
  - 그럼에도 불구하고 기존 최고의 성능을 보였던 모델 결과 대비 좋은 성능을 보여줌

Pre-Trained Model의 시대

- Attention is all you need (The Transformer)
  - 모델의 큰 구조는 기계번역에서 자주 사용하는 Encoder-Decoder 형식
  - 번역하고자 하는 문장(Source Sentence)을 Input으로 입력하여 Encoder가 특정 벡터로 정보를 저장한 뒤, Decoder가 해당 정보를 통해 번역문을 만들어 내면 정답 번역 문장(Label)과 Loss 구해서 학습하는 구조.
  - 하지만 보통의 Encoder와 Decoder는 RNN류의 LSTM이나 GRU Module 을 사용 하고, Attention을 적용 하는 방식을 사용 했는데, The Transformer는 해당 부분을 RNN을 전혀 쓰지 않고 여러 개의 Module을 이어서 만듦
  - 모든 Token을 순서대로 입력 받는 RNN과 다르게 모든 Token을 한번에 받아 처리하는 방식을 사용하여서, 병렬처리가 가능하다는 장점을 가지고 있음.

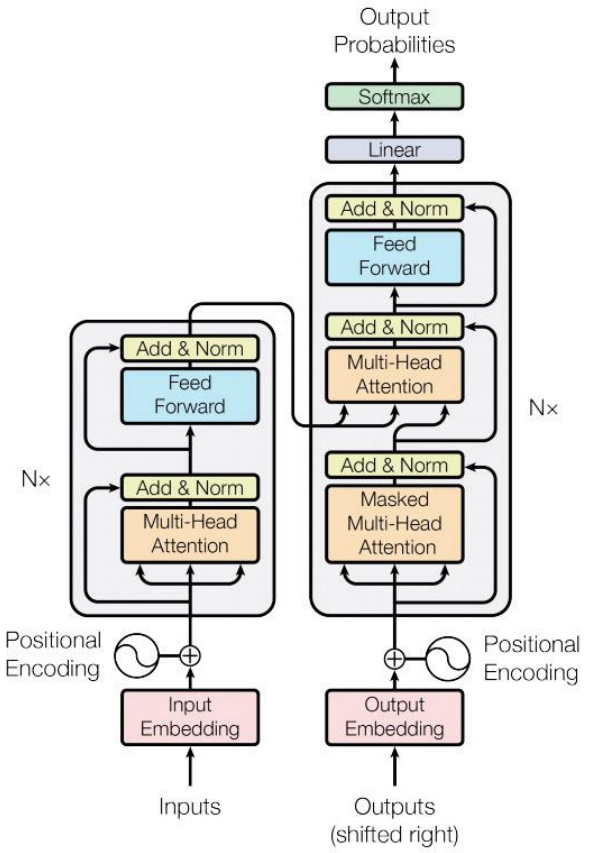
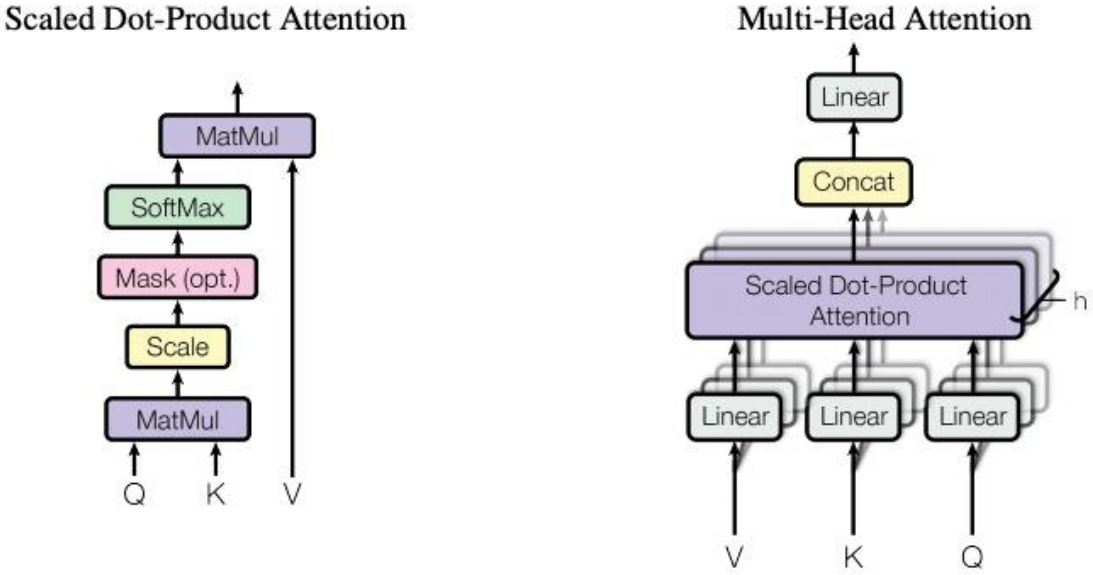


Figure 1: The Transformer - model architecture.

Pre-Trained Model의 시대

- Attention is all you need (The Transformer)
  - Multi-Head Attention이라는 Module 과 Feed Forward 변환을 ResNet에서 사용된 Shortcut으로 묶어 놓은 기본 Module을 사용.
  - 아래 그림에서 보듯 여러 개의 해당 Module들을 N번 쌓아둔 Encoder와 해당 정보를 받아 비슷한 구조의 Module들을 또 여러겹을 쌓아둔 Decoder로 구성되어 있음



Pre-Trained Model의 시대

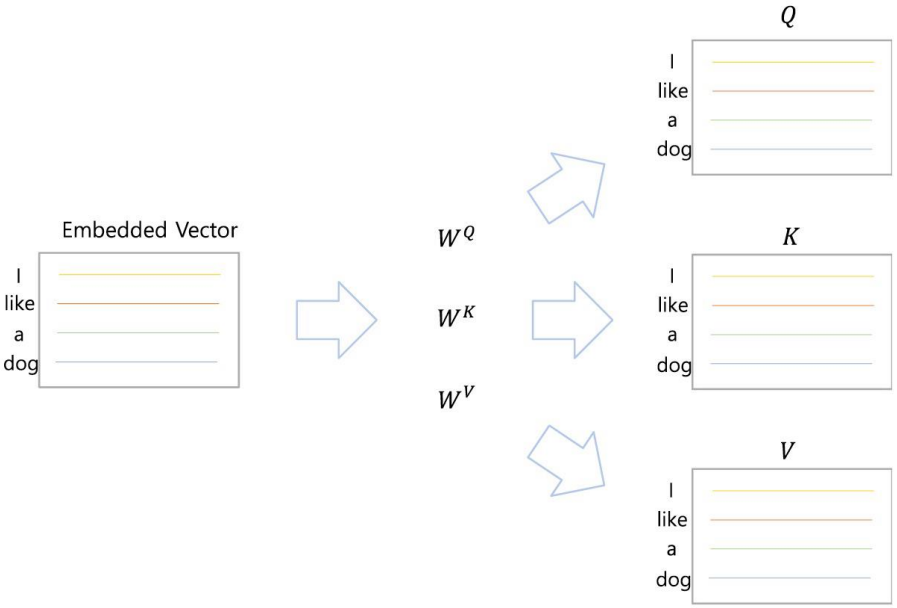
- Attention is all you need (The Transformer)
  - Multi-Head Attention Module은 이름에서 볼 수 있듯이 여러 개의 세부 Module Scaled Dot-Product Attention의 결과물을 모아서 하나의 Layer를 통과 시켜 정보를 전달. 그래서Multi-Head Attention 이해에 앞서 Scaled Dot-Product Attention에 대한 Module의 이해가 선행 되어야함.

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

  - Module의Input은 각각Query(Q), Key(K), Value(V)라고 표현.
  - Tokenized된 문장에 대해 기본 Embedding Vector와Position Encoding Vector를 합해 표현하고, 이에 Q, K, V에 대한 각각의 행렬을 곱하여Q, K, V 행렬로 변환시킴. 같은 문장이지만 다른 표현 방법으로 표현된 행렬들이라고 생각 할 수 있음

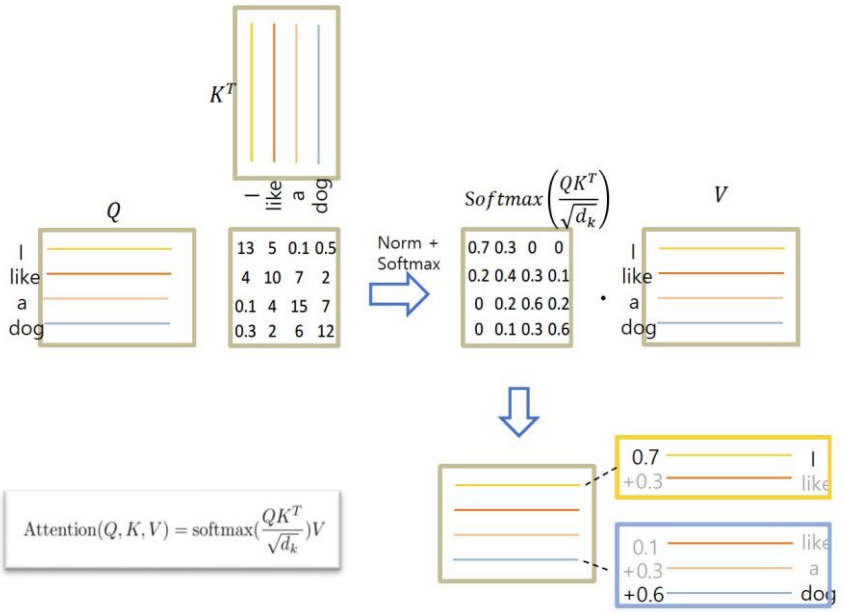
Pre-Trained Model의 시대

- Attention is all you need (The Transformer)
  - Q와K 행렬의 행렬곱 연산에 대한 해석. 행렬곱 연산을 한 결과 행렬의 요소들은 각 Token Vector의 Dot-Product라고 볼 수 있으며, 일반적으로 벡터 간 거리를 의미한다고 볼 수 있음.
  - 그리고 나누어 주는 벡터의 크기(dk) 값은 벡터크기에 대한 영향력을 줄여주는 요소라고 보면 되고, Row-By-Softmax를 취해줌으로서 나온 결과물은 각 Token Q Vector에 대한 다른 Token들의 K Vector간의 유사도를 나타내는 상대 점수라고 해석할 수 있음



Pre-Trained Model의 시대

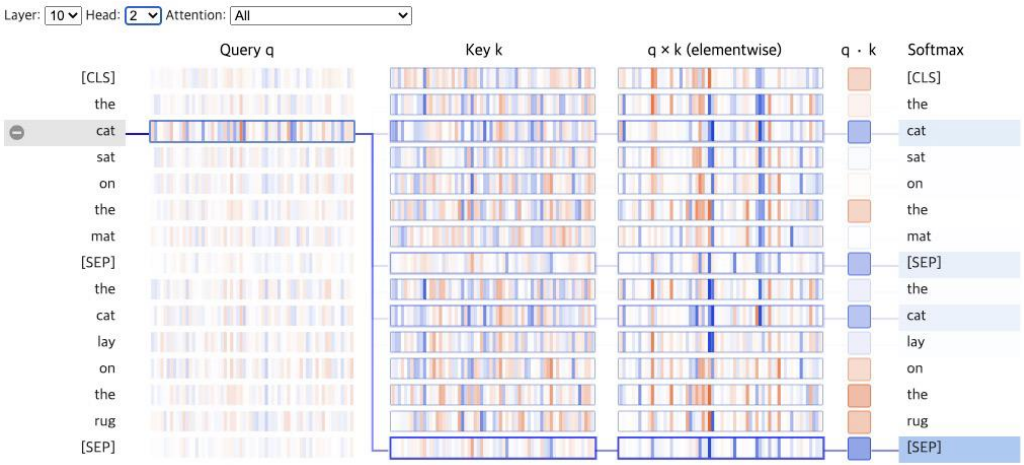
- Attention is all you need (The Transformer)
  - "I"라는 Token Q Vector에 대해서 "I" Token의 K Vector는 0.7 정도의 유사성을 "like"라는 Token에 대해서는 0.3 정도의 유사성을 가지고 있다고 볼 수 있음.
  - 그렇게 만들어진 행렬과 V 행렬을 다시 한번 행렬곱 연산을 함. 이 때, V 행렬에대해 Row측면에서 바라보면 각 Token Vector 입장 에서 볼 수 있게 됨.
  - 해당 연산은 Token Vector에 대한 상대 점수 Weighted Average라고 해석할 수 있음.
  - 우측 그림을 보면 위 설명했던 "I" vs ["I", "like"]간의 상대 점수 0.7, 0.3만큼의 Weighted Vector로 최종 결과가 나타나는 것을 확인 할 수 있음



Pre-Trained Model의 시대

- Attention is all you need (The Transformer)
  - Multi-Head Attention을 통과 하는 과정은 "다양한 Token별 관계에 대한 Weight Average 결과를 종합” 하는 것 이라고 볼 수 있음.
  - The Transformer의 Encoder는 이와 같은 Module을 여러 겹 이어 두어 정보의 재조합을 여러 번 작업하게끔 하였고, 끝에 나오는 Output을 Decoder에게 넘겨 주는 방식으로 모델이 구성 되어 있음
  - 해당 논문의 큰 성과는 기존 기계 번역 모델들에서 사용 하는 RNN류의 Module을 벗어 났다는 점과 Attention 정보만 가지고도 좋은 성능을 낼 수 있다는 점이라고 볼 수 있음

참고
- 모델에 대한 시각화가 잘 되어 있는 사이트 (https://github.com/jessevig/bertviz)
- github 의 Neuron View (Colab) 를 통해 학습된 Transformer 모델의 핵심인 Scaled Dot-product Attention Module을 더 잘 이해 할 수 있음.
- 해당 사이트를 통해 Token 별Q, K 표현의 관계가 어떻게 변하는지, 추가로 Layer와 Head 별 차이도 눈으로확인



Pre-Trained Model의 시대

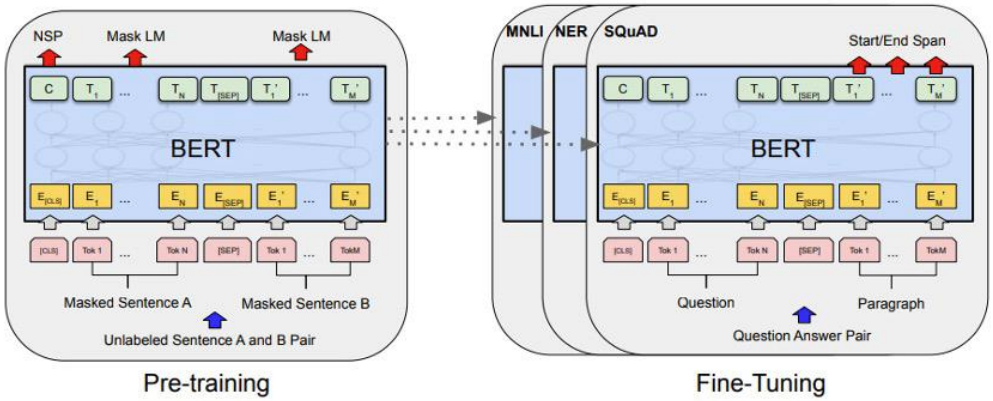
- Pre-training of Deep Bidirectional Transformers for Language Understanding (BERT)
  - NLP에서의 Pre-trained 모델을 학습하고, 이를Fine-tuning하는 모델의 가능성과 동시에 높은 성능을 보여줌.
  - 논문 발표할 시의 BERT는11개 Task에서 state-of-the-art 의 성능을 달성하였고, 기존 성능 대비 높은 향상을 보여 주면서 많은 관심을 받게 됨

Pre-Trained Model의 시대

- BERT 모델 학습
  - 일반문서에서 Feature를 학습하는 Unsupervised Pre-training 과정과 Pre-trained 모델을 가지고 각각의 특정Task에서 한번 더 학습을 시키는 Fine-tuning 과정을 거침.
  - Unsupervised Pre-training 과정에서는 일반 문장들만 Input으로 사용하고 특정한 답이 없기 때문에 Task과 정답Label을 만들어줘야 함.
  - BERT는 문장에서 랜덤으로 몇 개의 Token을 가리고 주변 문맥으로 해당 Token을 맞추는 Task인 Masked LanguageModel(MLM)과 연이은 문장Pair인지 의도적으로 랜덤으로 매칭 시킨Pair 문장 인지를 구분하는Task인 Next Sentence Prediction (NSP) 두 가지에 대한 학습을 진행 시킴.
  - Pre-training 과정에서는 단어간 관계와 문장 단위의 이해를 중점으로 학습을 시키고, Fine-tuning에서는 주어진 다양한 Task에 대한 새로운 데이터를 다시 Input으로 받아서 학습 시켜Task별 최종 모델을 구성

Pre-Trained Model의 시대

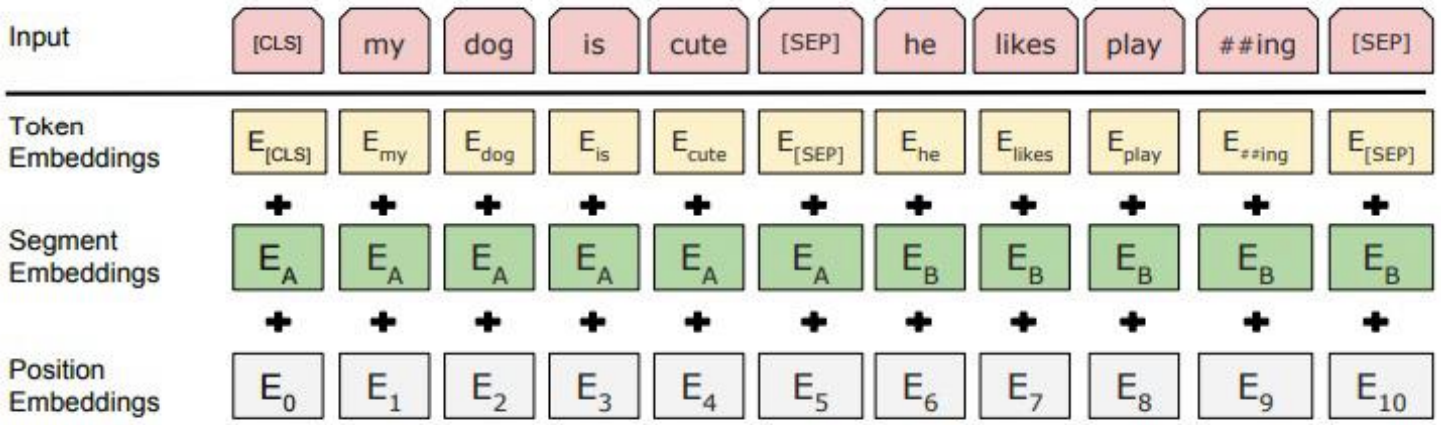
- 모델 구조
  - 기본적으로는 BERT는 The Transformer를 사용.
  - 하지만, 번역을위한 모델이었던 Transformer와는 달리 BERT는 일반적인 Language Model이기 때문에 Transformer의 Encoder 부분만 떼어 와서 모델을 구성.
  - 모델 구조적으로는 전에 설명한 The Transformer의 내용과 다르지 않음
  - Transformer의 Token Embedding과 Position Embedding은 그대로 사용하고, 두 개의 문장을 이어서 Input을 받기 때문에 문장을 구분 할 수 있는 Segment Embedding을 추가함.
  - 그리고 한 가지 [CLS], [SEP] 라는 특별한 Token을 추가 하는데, [SEP]는 문장의 끝을 알리는 문장 구분의 목적으로, [CLS]는 문장의 시작을 알리는 용도로 추가. 그리고[CLS] Token은 학습 하는 동안 문장 전체의 정보를 담는 목적으로 사용.



먼저 Token Embedding에서는 두 가지 특수 토큰(CLS, SEP)을 사용하여 문장을 구별하게 되는데요. Special Classification token(CLS)은 모든 문장의 가장 첫 번째(문장의 시작) 토큰으로 삽입됩니다. 이 토큰은 Classification task에서는 사용되지만, 그렇지 않을 경우엔 무시됩니다. 또, Special Separator token(SEP)을 사용하여 첫 번째 문장과 두 번째 문장을 구별합니다. 여기에 segment Embedding을 더해서 앞뒤 문장을 더욱 쉽게 구별할 수 있도록 도와줍니다. 이 토큰은 각 문장의 끝에 삽입됩니다.

Pre-Trained Model의 시대

- BERT
  - Output은 Token 개수만큼의 벡터들로 구성 되어 있고, 이 값들은 각 Token들마다 해당 문장에서 연관성 있는 Token들끼리의 정보를 잘 혼합해 놓은 것.
  - 그리고 연구진들은 각 Token의 Output 으로는 Mask가 되어 있는 Token을 예측 하는MLM Task를 해결하는데 사용 하였고, [CLS] Token의Output 값을 이용해서는 Sentence끼리의 관계를 맞추는 NSP Task를 해결하는데 사용



https://hwiyong.tistory.com/392

Pre-Trained Model의 시대

- BERT
  - 앞서 설명한 모델과 데이터를 통해 Pre-Training이된 모델의 결과물은 새로운 형태의 Embedding이라고 볼 수 있음.
  - 서로 다른Task마다 디테일한 구성은 다르지만 Pre-Trained Model Output들에 간단한 FC만 추가하여 조금 더 학습시키는 Fine-tuning을 거치면 각 Task마다 좋은 성능을 내는 최종 모델을 얻을 수 있었기 때문.
  - 이런 점이 NLP에서 자주 사용하였던 Pre-Trained Embedding과 유사 하였지만, 기존의 word2vec 형태의Fixed Embedding과는 다르게 BERT는 문장의 문맥에 따라 Embedding Vector가 달라지는 특성 때문에 Contextual Embedding이라고 표현하고 있음.
  - 이 덕분에 서로 다른 문맥에서 사용 되는 동음 이의어 문제도 해결 할 수 있었음

Pre-Trained Model의 시대

- BERT의 연구성과
  - BERT 모델은 모델의 성능이 SOTA를 여러 개 갱신했다는 점과 몇 개의 Task에서는 인간의 능력을 넘어섰다는 점에서도 큰 성과를 냈다고 볼 수 있음
  - 하지만 BERT의 진짜 성과는 NLP 영역에서의 Pre-Training & Finetuning 형태의 학습 방법을 성공적으로 보여줬다는 점.
  - Computer Vision 영역에서만 적용 되는 것이 아니라 NLP에서도 잘 적용된다는 모습을 보여 줬을 뿐 아니라 이 이후 활발한 연구를 이끌어 냄.
  - BERT 이후에 BERT보다 당연히 좋은 성능의 모델들은 많이 나왔지만, 대부분의 모델은 BERT의 핵심적인 요소들을 품고 있음.
  - BERT는 최신NLP 연구를 위해서는 반드시 알고 넘어가야하는 요소.