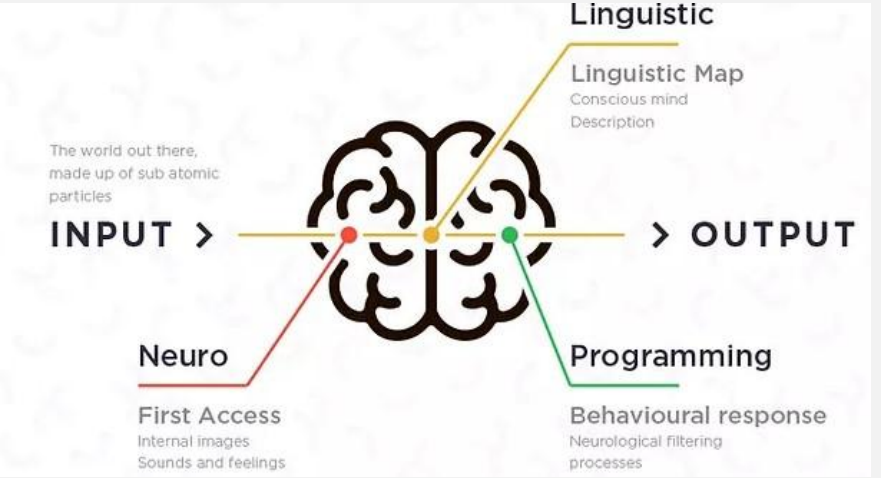
자연어처리 개념과 언어모델

머신러닝의 기초 4조

자연어 처리 Natural Language Processing

NLP

- 기계가 인간의 언어를 처리하고 이해하여 반복적인 작업을 자동으로 수행할 수 있도록 도움
- 전달자가 메시지 전달하면 수신자가 메시지를 해독

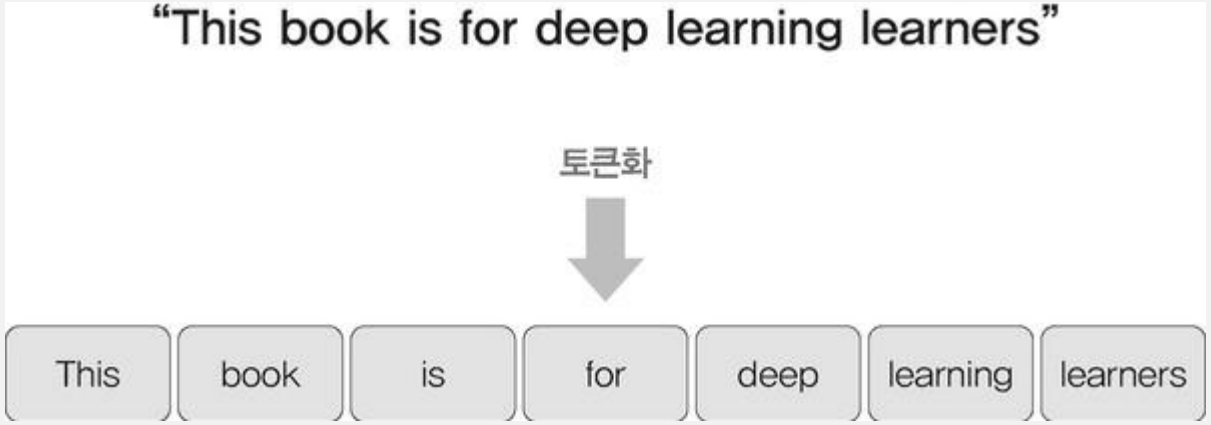


다양한 처리 기술

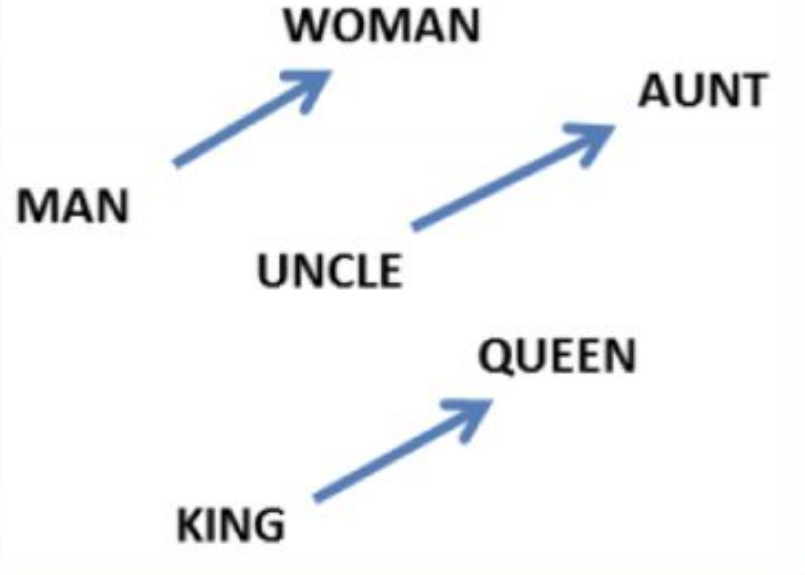
- Symbolic Approach: 규칙 또는 지식 기반 접근법
- Statistical Approach: 통계 또는 확률 기반 접근법
  a) TF - Term Frequency = 문서에 단어가 나타나는 빈도를 단어 수로 나눈 값
  b) DF - Document Frequency = 특정 용어를 포함하는 문서의 수

자연어 처리 단계

1) 전처리 - 불필요한 정보, 반복적으로 사용된 문자 등을 처리
2) Tokenizing
   a) “자연어를 어떻게 살펴볼까?”
   b) 텍스트 데이터에 대한 정보를 단위별로 나누는 것
3) Lexical Analysis - 어휘 분석, 형태소 분석, 상호 참조
4) Syntactic Analysis - 구문 분석
5) Semantic Analysis - 의미 분석



Word2Vec - 임베딩



- Word2vec (word to vector) 알고리즘

  자연어(특히, 단어)의 의미를 벡터 공간에 임베딩

  => 중심 단어의 주변 단어들을 이용해 그 의미를 추론하여 파악

- Dense representation으로, 한정된 차원으로 표현이 가능하며

  의미 관계를 유추하고 비지도 학습으로 단어의 의미 학습이 가능

- 장점
  a) 단어 간의 유사도 측정과 관계 파악에 용이
  b) 벡터 연산을 통한 추론이 가능 (예시: 한국 - 서울 + 도쿄 = 일본)

- 단점
  a) 단어의 subword information 무시 (예시: 서울 vs 서울시 vs 고양시)
  b) OOV(Out of vocabulary)에서 적용 불가능

FastText - 임베딩

<N-gram>

- word2vec 알고리즘의 단점을 보완할 수 있는 임베딩 알고리즘인 **FastText 알고리즘** 등장
- word2vec 알고리즘과 유사하지만 **단어를 n-gram으로 나누어 학습**
- **n-gram vector**를 형성
  a) 입력 단어가 vocabulary(단어사전)에 있을 경우 해당 단어의 word vector를 return
  b) OOV(Out of Vocabulary)일 경우, 입력 단어의 모든 n-gram vector 합산의 평균을 return (n-gram: 2-5 일 경우 bigram, trigram, 4-gram, 5-gram 합산의 평균을 return)
- 오탈자와 OOV에 대해서 n-gram vector이 return 되기 때문에 word2vec의 단점을 보완한 형태의 임베딩 모델임

임베딩 정리

- 다른 NLP 모델의 input으로 사용 가능
- 토픽 키워드 추출 가능
- word embedding 방식의 한계점
  a) 동형어, 다의어에 대해 embedding 성능 저하
  b) 주변 단어를 통한 학습이기 때문에 문맥 고려 불가능

언어모델

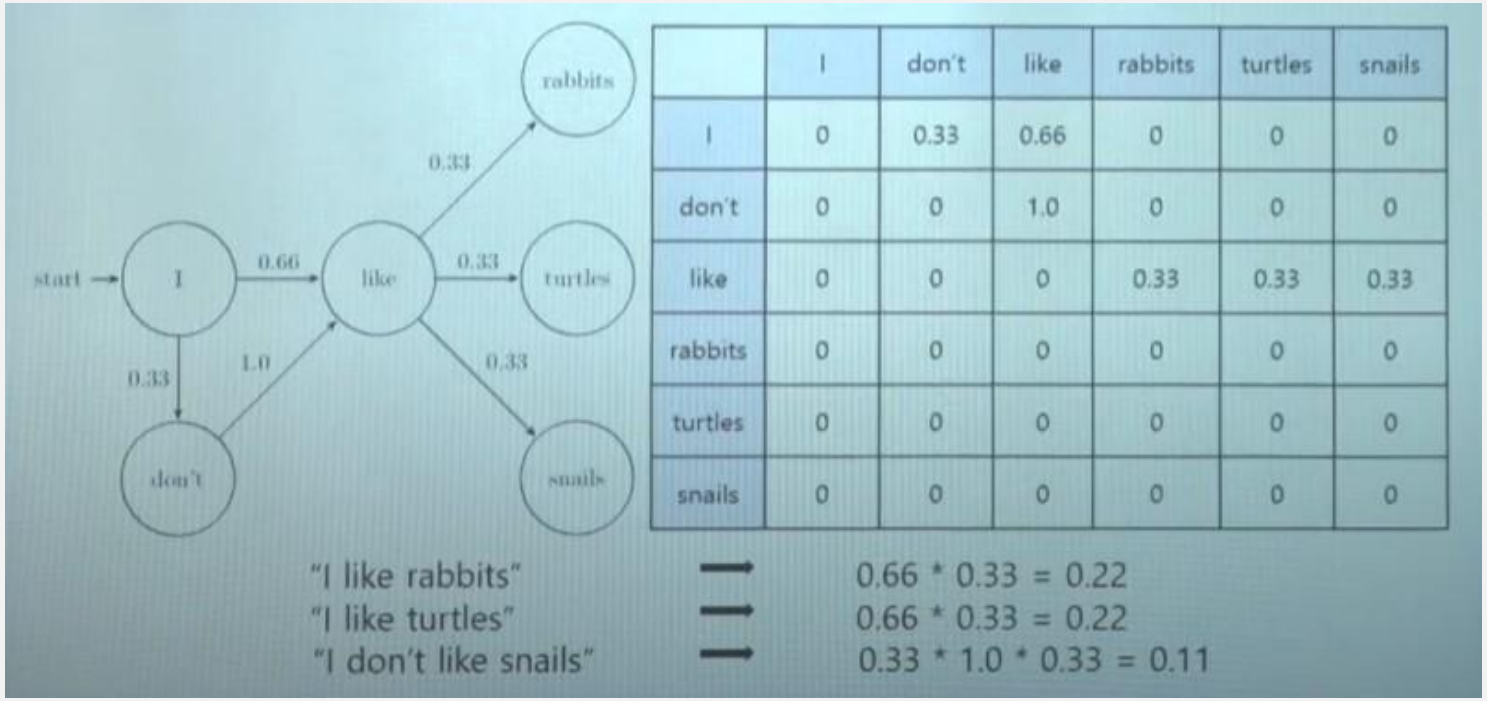
- ‘자연어’의 법칙을 컴퓨터로 구현한 모델
- 주어진 단어들로부터 다음에 등장할 단어의 확률을 예측하는 방식으로 학습

  (이전 data로 미래 data 예측)
- 좋은 언어모델은 언어의 특성이 잘 반영되고, 문맥을 잘 계산하는 모델

Markov 확률 기반 언어모델

Markov Chain Model

- 초기 언어 모델은 다음에 나올 단어나 문장을 확률과 n-gram 기반으로 계산
- 해당 확률을 최대로 하도록 네트워크를 학습



	I	don't	like	rabbits	turtles	snails
I	0	0.33	0.66	0	0	0
don't	0	0	1.0	0	0	0
like	0	0	0	0.33	0.33	0.33
rabbits	0	0	0	0	0	0
turtles	0	0	0	0	0	0
snails	0	0	0	0	0	0

RNN 기반 언어모델

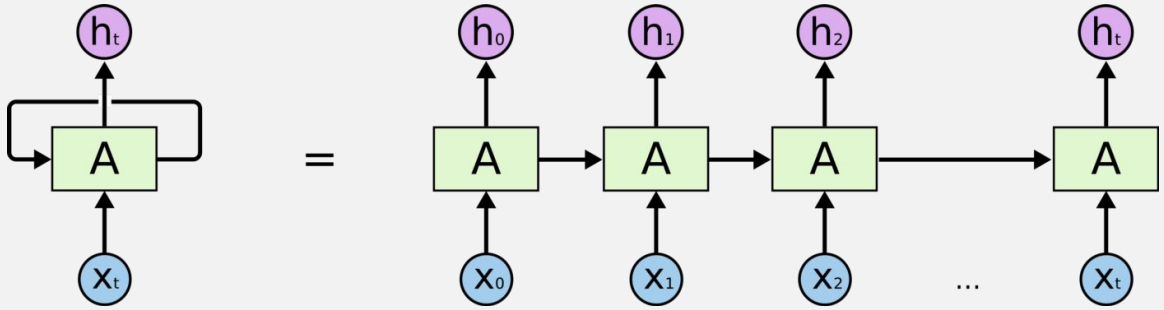
- RNN(순환 신경망)이란 유닛 간의 연결이 **순환적 구조**를 가지고 있는 인공 신경망의 한 종류

- 이전 state 정보가 다음 state를 예측하는데 사용되므로 시계열 데이터 처리에 특화되어 있음

- 이러한 RNN의 특성을 이용하여 문장 전체의 문맥을 고려한 최종 출력을 return

- vector -> context vector 문장이 가진 context vector를 output으로 얻을 수 있음

- output layer에 Classification layer만 추가하면 기계 독해, 긍부정분류 등이 가능



- RNN model application

Seq2Seq (Sequence to Sequence)

RNN 모델을 이용하여 Encoder layer에서 Context Vector를 추출하고 Decoder layer에서 Context vector를 입력층으로 하여 출력을 예측하는 방식으로 번역 등에 사용할 수 있음

RNN 기반 언어모델

RNN의 구조적 문제점

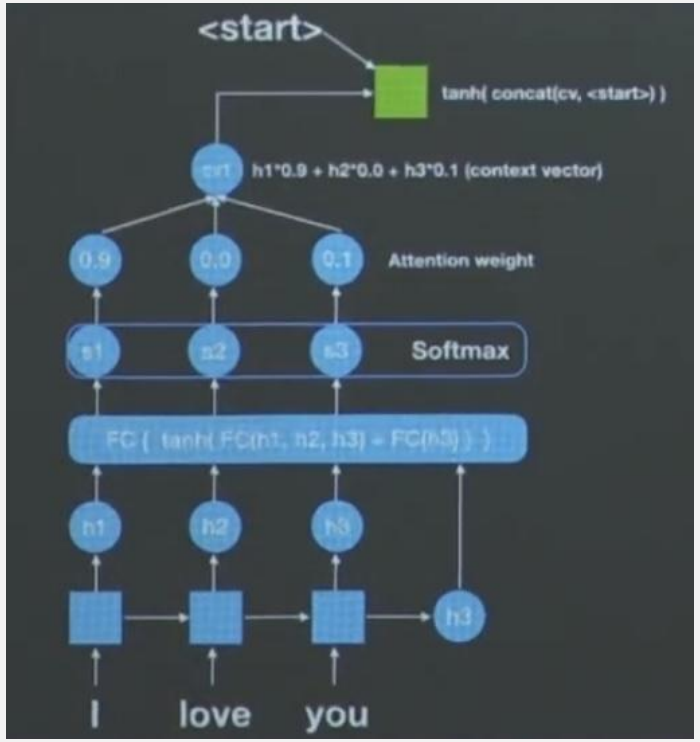
a) 입력 sequence의 길이가 매우 긴 경우, 처음에 나온 token의 정보 희석

b) 고정된 context vector 사이즈로 인해 긴 sequence에 대한 정보 함축 어려움

c) 모든 token이 영향을 미치기 때문에, 중요하지 않은 token도 영향을 줌

Attention 모델

RNN의 한계점을 보완하여 만들어진 모델로, 중요도를 고려하여 정보를 처리



Attention weight와 hidden state를 곱하여 Context vector (Decoder의 input)

⬆

Softmax를 취해서 0과 1 사이의 값 (Attention weight) 으로 변환

⬆

layer의 output을 각 셀의 score로 결정

⬆

셀의 각 output을 입력으로 하는 Feed forward Fully connected layer

⬆

RNN 모델에서 시작

Attention 모델

장점

a) input의 attention weight로 중요도 확인 가능

b) 기존의 Seq2Seq의 encoder, decoder 성능을 향상

단점

a) 여전히 RNN을 거치기 때문에 연산 속도가 느림

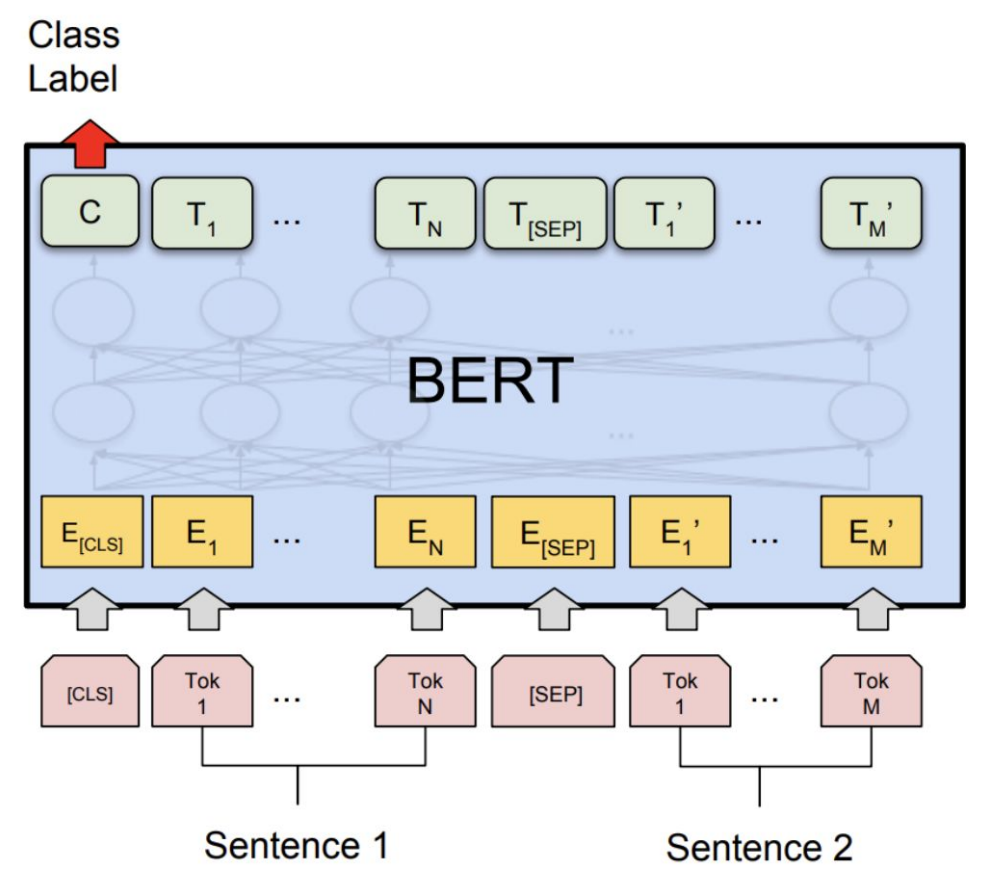
- 단점을 보완하기 위해 RNN을 제거한 모델이 **“self-attention”** 모델
- 기존의 attention 모델은 decoder의 적절한 output을 위한 weight를 찾고자 했다면 self-attention 모델은 weight가 input 값을 가장 잘 표현하도록 학습
- self-attention 여러 개를 동시에 수행 -> **Multi-head Self Attention**

Transformer Model

Transformer Model

- 순차적 데이터의 관계를 추적하여 문맥과 의미를 학습하는 신경망
- 이 모델은 attention 또는 self-attention 수학적 기술 세트를 적용함
- 순차 입력 데이터를 처리하도록 설계되어 있음
- 번역, 테스트 요약과 같은 작업에 응용

BERT 모델 구조



Input sentence를 예측하도록 학습

⬆

Contextual representation of token

⬆

Transformer layer

⬆

Input embedding layer

⬆

Input: 2개의 sentence

BERT 학습 데이터

WordPiece Tokenizing

- 입력 문장을 tokenizing 하고 입력 문장을 token sequence를 만들어 학습에 사용
- 2개에 token sequence가 학습에 사용
- BPE (Byte Pair Encoding) 알고리즘을 사용하여 빈도수에 따라 의미 있는 패턴으로 잘라서 토큰화

WordPiece Tokenizing 순서

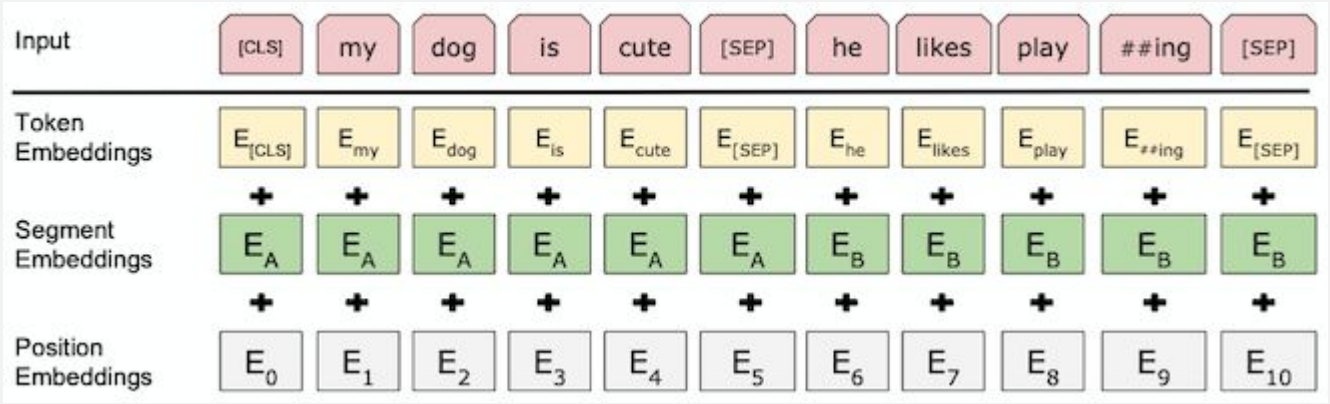
1) Tokenize
   a) 예시: “경찰청 철장살은 외청살이고 검찰청 철장살은 쌍철창살이다.”
   b) 같은 글자라도 서로 차이가 있다고 가정
   c) BERT의 경우 뒷단어에 “##”을 넣어서 구별
      예시: 경찰청 -> 경 ##찰 ##청
2) Building Vocab
   a) Vocab의 생성은 정해진 iteration을 모두 수행 후 tokenized sentence를 토큰화해서 생성
3) Vocab 기준으로 Bi-gram pairs 통해 가장 빈도수가 높은 패턴 기준으로 학습
4) 가장 좋은 짝으로 (best pair) 단어들 정리
5) Vocab 후보를 계속 업데이트하며 단어 생성

BERT 모델 - Masking 기법

1) 토큰화된 input sentence의 앞에 [CLS]와 뒤에 [SEP]라는 토큰을 붙임

2) input 토큰 중 15% 확률로 무작위 선택

3) 선택된 토큰을 80%로 [MASK] 토큰 (Masking), 10%로 Randomly replacing, 10%로 Unchanging

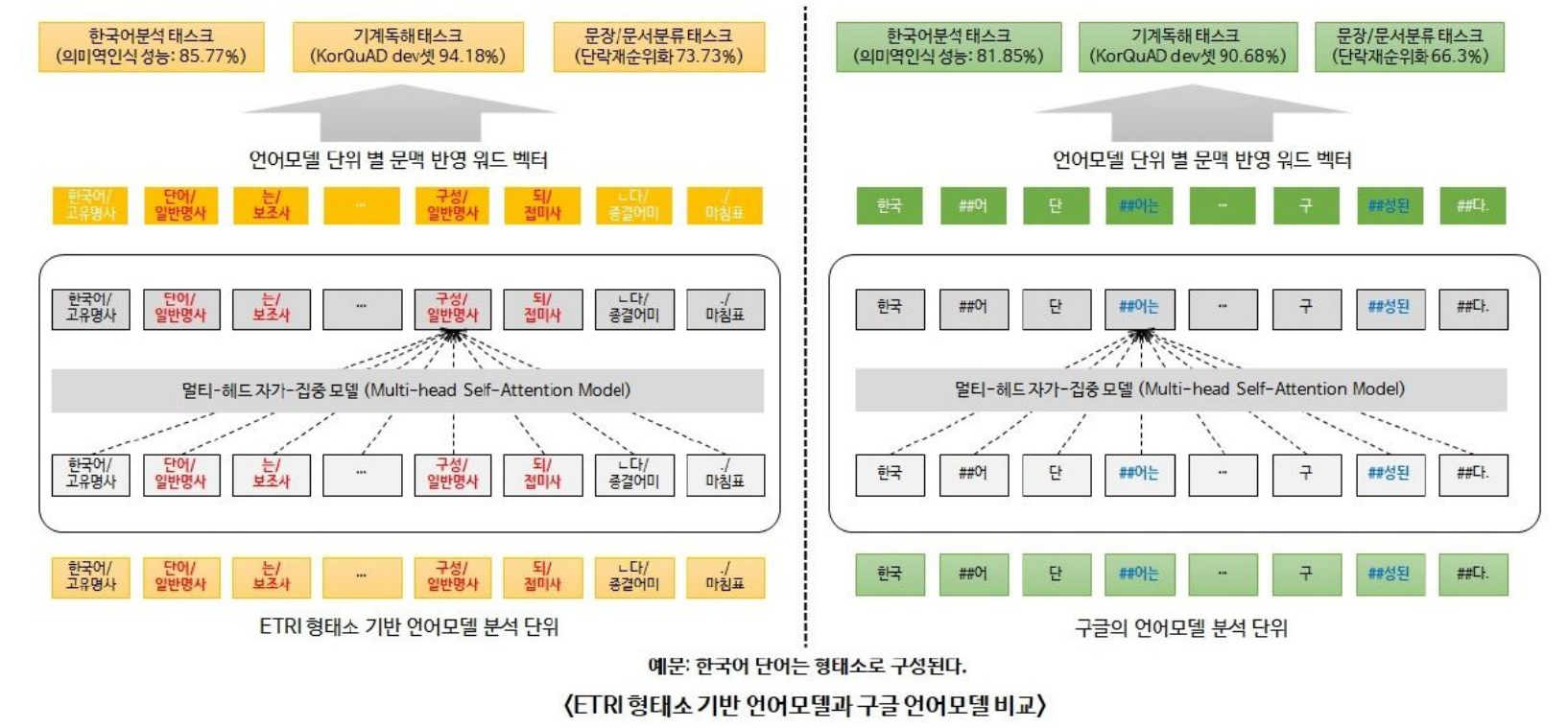


- WordPiece를 이용해 token embedding
- 2개의 input sentence 구분
- Segment embedding을 통해 2개로 나눈 후 문장 내의 순서 학습

KorBERT

- 한국어에 사용된 BERT는 일반 BERT와 다름
- 첫 입력 토큰 자체를 줄 때 형태소 분석을 한 다음에 줌
- 형태소 태그를 다 붙인 다음에 WordPiece 알고리즘 타게 됨
- 한 단어가 여러 의미를 가질 수 있기 때문에 KorBERT 사용하여 각각 다른 단어로 학습

  예시: “이” = E, 2, 이것, 무엇’이’, 이빨 등



예문: 한국어 단어는 형태소로 구성된다.

〈ETRI 형태소 기반 언어모델과 구글 언어모델 비교〉