

자연어 처리 개발 및 실습

자연어 처리 예제 실습

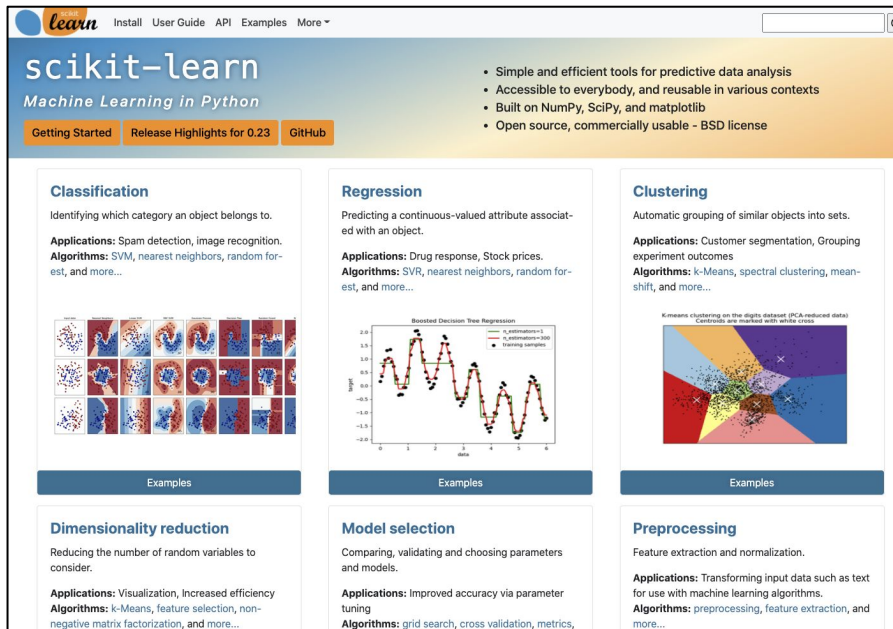
자연어 처리 관련 라이브러리

사이킷런 (scikit-learn)

- 특징 추출
- TfidfVectorizer

자연어 처리 관련 라이브러리

사이킷런 (scikit-learn)



The screenshot shows the scikit-learn website homepage. At the top, there's a navigation bar with links: Install, User Guide, API, Examples, and More. The main header features the scikit-learn logo and the tagline "Machine Learning in Python". Below this, there are buttons for "Getting Started", "Release Highlights for 0.23", and "GitHub". A list of bullet points highlights key features: "Simple and efficient tools for predictive data analysis", "Accessible to everybody, and reusable in various contexts", "Built on NumPy, SciPy, and matplotlib", and "Open source, commercially usable - BSD license". The main content area is divided into six sections, each with a title, description, applications, algorithms, and an "Examples" button. The sections are: Classification, Regression, Clustering, Dimensionality reduction, Model selection, and Preprocessing. Each section includes a small representative image or plot.

scikit-learn
Machine Learning in Python

Getting Started Release Highlights for 0.23 GitHub

- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification
Identifying which category an object belongs to.
Applications: Spam detection, image recognition.
Algorithms: SVM, nearest neighbors, random forest, and more...

Regression
Predicting a continuous-valued attribute associated with an object.
Applications: Drug response, Stock prices.
Algorithms: SVR, nearest neighbors, random forest, and more...

Clustering
Automatic grouping of similar objects into sets.
Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, and more...

Dimensionality reduction
Reducing the number of random variables to consider.
Applications: Visualization, Increased efficiency
Algorithms: k-Means, feature selection, non-negative matrix factorization, and more...

Model selection
Comparing, validating and choosing parameters and models.
Applications: Improved accuracy via parameter tuning
Algorithms: grid search, cross validation, metrics,

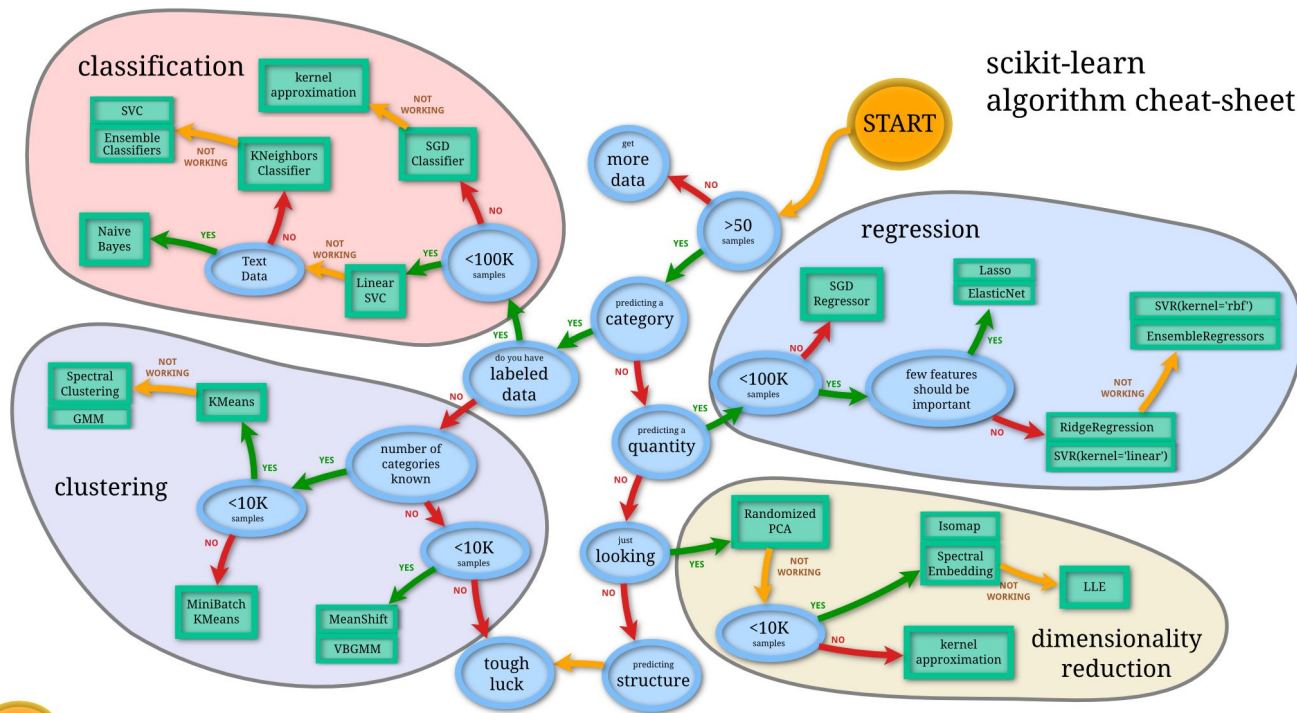
Preprocessing
Feature extraction and normalization.
Applications: Transforming input data such as text for use with machine learning algorithms.
Algorithms: preprocessing, feature extraction, and more...

<https://scikit-learn.org/stable/>

자연어 처리 관련 라이브러리

사이킷런 (scikit-learn)

scikit-learn
algorithm cheat-sheet



자연어 처리 관련 라이브러리

사이킷런 (scikit-learn) - 특징 추출

- CountVectorizer
- TfidfVectorizer

자연어 처리 관련 라이브러리

사이킷런 (scikit-learn) - 특징 추출 (CountVectorizer)

```
[1] 1 from sklearn.feature_extraction.text import CountVectorizer  
    2
```

```
[2] 1 text_data = ['나는 배가 고프다', '내일 점심 뭐먹지', '내일 공부 해야겠다', '점심 먹고 공부 해야지']  
    2  
    3 count_vectorizer = CountVectorizer()
```

```
[3] 1 count_vectorizer.fit(text_data)  
    2 print(count_vectorizer.vocabulary_)
```

☞ {'나는': 2, '배가': 6, '고프다': 0, '내일': 3, '점심': 7, '뭐먹지': 5, '공부': 1, '해야겠다': 8, '먹고': 4, '해야지': 9}

```
[4] 1 sentence = [text_data[0]] # ['나는 배가 고프다']  
    2 print(count_vectorizer.transform(sentence).toarray())
```

☞ [[1 0 1 0 0 0 1 0 0 0]]

자연어 처리 관련 라이브러리

TF-IDF란?

- TF (Term Frequency)
- IDF (Inverse Document Frequency)
- 다음 분석에서 주로 사용됨
 - 문서 간의 비슷한 정도를 구함
 - 특정 단어가 문서내의 얼마나 중요한지 척도를 계산
 - 문서 내 단어들에 척도를 계산해서 핵심어를 추출
 - 검색엔진에서 검색결과의 순위를 결정

자연어 처리 관련 라이브러리

TF-IDF란?

- TF (Term Frequency)

간단히 말해 **특정 단어의 빈도**를 뜻함

- DF (Document Frequency)

단어가 전체 문서 집합 내에서 얼마나 **공통적으로 많이 등장**하는지 나타냄

$$df(t, d) = \frac{|\{d \in D : t \in d\}|}{|D|} = \frac{\text{단어 } t \text{가 포함된 문서의 수}}{\text{전체문서의 수}}$$

DF가 크다는 것은, 모든 문서에 등장하는 흔한 단어라는 뜻으로 해석됨

- IDF (Inverse Document Frequency)

단어가 흔할 수록 중요도를 작게 계산하기 위해 **DF에 역수**를 취함

- TF-IDF

$$tfidf(t, d, D) = tf(t, d) \times idf(t, D)$$

자연어 처리 관련 라이브러리

사이킷런 (scikit-learn) - 특징 추출 (TfidfVectorizer)

```
1 from sklearn.feature_extraction.text import TfidfVectorizer
2
3 text_data = ['나는 배가 고프다', '내일 점심 뭐먹지', '내일 공부 해야겠다', '점심 먹고 공부 해야지']
4 tfidf_vectorizer = TfidfVectorizer()
5
6 tfidf_vectorizer.fit(text_data)
7 print(tfidf_vectorizer.vocabulary_)
```

{'나는': 2, '배가': 6, '고프다': 0, '내일': 3, '점심': 7, '뭐먹지': 5, '공부': 1, '해야겠다': 8, '먹고': 4, '해야지': 9}

```
1 sentence = [text_data[3]] # ['점심 먹고 공부 해야지']
2 print(tfidf_vectorizer.transform(text_data).toarray())
```

```
[[0.57735027 0.          0.57735027 0.          0.          0.
  0.57735027 0.          0.          0.          ]
 [0.          0.          0.          0.52640543 0.          0.66767854
  0.          0.52640543 0.          0.          ]
 [0.          0.52640543 0.          0.52640543 0.          0.
  0.          0.          0.66767854 0.          ]
 [0.          0.43779123 0.          0.          0.55528266 0.
  0.          0.43779123 0.          0.55528266]]
```

자연어 처리 관련 라이브러리

자연어 토크나이징 도구

- 영어 토크나이징 라이브러리
- 한글 토크나이징 라이브러리

자연어 처리 관련 라이브러리

영어 토큰나이징 라이브러리 - NLTK

```
[4] 1 from nltk.tokenize import word_tokenize
```

```
[5] 1 sentence = "Natural language processing (NLP) is a subfield of linguistics, computer science, and artificial inte
```

```
[7] 1 nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...  
[nltk_data]   Unzipping tokenizers/punkt.zip.  
True
```

단어 단위 토큰나이징

```
[8] 1 print(word_tokenize(sentence))
```

```
[nltk_data] ['Natural', 'language', 'processing', '(', 'NLP', ')', 'is', 'a', 'subfield', 'of', 'linguistics', ',', 'computer',
```

자연어 처리 관련 라이브러리

영어 토큰나이징 라이브러리 - Spacy

```
1 import spacy
```

```
1 nlp = spacy.load('en')  
2 sentence = "Natural language processing (NLP) is a subfield of linguistics, computer science, and artificial in  
3  
4 doc = nlp(sentence)
```

```
1 word_tokenized_sentence = [token.text for token in doc]  
2 sentence_tokenized_list = [sent.text for sent in doc.sents]  
3 print(word_tokenized_sentence)  
4 print(sentence_tokenized_list)
```

```
'Natural', 'language', 'processing', '(', 'NLP', ')', 'is', 'a', 'subfield', 'of', 'linguistics', ',', 'computer'  
'Natural language processing (NLP) is a subfield of linguistics, computer science, and artificial intelligence co
```

자연어 처리 관련 라이브러리

한글 토큰나이징 라이브러리

한글 토큰나이징에는 영어에 없는 형태소 분석, 음소 분리 같은 기능이 필요함

따라서 앞에서 소개한 라이브러리로는 분석이 불가능함

KoNLPy를 이용하면 형태소 분석, 구문 분석이 가능함

자연어 처리 관련 라이브러리

한글 토큰나이징 라이브러리 - KoNLPy

[Colab에 KoNLPy 설치하기](#)

KoNLPy에서 제공하는 형태소 분석기의 목록

- Hannanum
- Kkma
- Komoran
- Mecab
- Okt (Twitter)

자연어 처리 관련 라이브러리

한글 토큰나이징 라이브러리 - KoNLPy.Okt

- **okt.morphs()**
텍스트를 형태소 단위로 나눔
- **okt.nouns()**
텍스트에서 명사만 뽑아냄
- **okt.phrases()**
텍스트에서 어절을 뽑아냄
- **okt.pos()**
각 품사를 태깅하는 역할

자연어 처리 관련 라이브러리

한글 토큰나이징 라이브러리 - KoNLPy.Okt

```
1 from konlpy.tag import Okt
```

```
1 okt = Okt()
```

```
1 text = "안녕하세요 오늘은 날씨가 좋습니다. 집에 가고 싶네요."  
2  
3 print(okt.morphs(text))  
4 print(okt.morphs(text, stem=True))
```

```
['안녕하세요', '오늘', '은', '날씨', '가', '좋습니다', '.', '집', '에', '가고', '싶네요', '.']  
['안녕하다', '오늘', '은', '날씨', '가', '좋다', '.', '집', '에', '가다', '싶다', '.']
```

```
1 okt.nouns(text)
```

```
['오늘', '날씨', '집']
```

```
1 okt.phrases(text)
```

```
['오늘', '날씨']
```


자연어 처리 관련 라이브러리

한글 토큰나이징 라이브러리 - KoNLPy.Okt

```
1 okt.pos(text)
```

```
[('안녕하세요', 'Adjective'),  
( '오늘', 'Noun'),  
( '은', 'Josa'),  
( '날씨', 'Noun'),  
( '가', 'Josa'),  
( '좋습니다', 'Adjective'),  
( '.', 'Punctuation'),  
( '집', 'Noun'),  
( '에', 'Josa'),  
( '가고', 'Verb'),  
( '싶네요', 'Verb'),  
( '.', 'Punctuation')]
```

```
1 okt.pos(text, join=True)
```

```
[ '안녕하세요/Adjective',  
 '오늘/Noun',  
 '은/Josa',  
 '날씨/Noun',  
 '가/Josa',  
 '좋습니다/Adjective',  
 './Punctuation',  
 '집/Noun',  
 '에/Josa',  
 '가고/Verb',  
 '싶네요/Verb',  
 './Punctuation']
```

자연어 처리 관련 라이브러리

한글 토큰나이징 라이브러리 - KoNLPy 데이터

```
1 from konlpy.corpus import kolaw # 한국 법률 말뭉치. 'constitution.txt'
2 from konlpy.corpus import kobill # 대한민국 국회 의안 말뭉치. '1809890.txt' 부터 '1809899.txt' 까지 구성
```

```
1 kolaw.open('constitution.txt').read()
```

'대한민국헌법\n\n유구한 역사와 전통에 빛나는 우리 대한국민은 3·1운동으로 건립된 대한민국임시정부의 법통과 불의에 항거한 4·19민주이념을 계승 영역에 있어서 각인의 기회를 균등히 하고, 능력을 최고도로 발휘하게 하며, 자유와 권리에 따르는 책임과 의무를 완수하게 하여, 안으로는 국민생활의 처 국민투표에 의하여 개정한다.\n\n 제1장 총강\n 제1조 ① 대한민국은 민주공화국이다.\n②대한민국의 주권은 국민에게 있고, 모든 권력 통일을 지향하며, 자유민주적 기본질서에 입각한 평화적 통일 정책을 수립하고 이를 추진한다.\n 제5조 ① 대한민국은 국제평화의 유지에 노력하고 : 다.\n②외국인은 국제법과 조약이 정하는 바에 의하여 그 지위가 보장된다.\n 제7조 ① 공무원은 국민전체에 대한 봉사자이며, 국민에 대하여 책임

```
1 kobill.open('1809890.txt').read()
```

'지방공무원법 일부개정법률안\n\n(정의화의원 대표발의)\n\n 의 안\n 번 호\n\n9890\n\n발의연월일 : 2010. 11. 12. \n\n발 의 를 양육하기 위하여 육아휴직을 할 \n\n수 있는 자녀의 나이는 만 6세 이하로 되어 있어 초등학교 저학년인 \n\n자녀를 돌보기 위해서는 해당 부모 호\n\n지방공무원법 일부개정법률안\n\n지방공무원법 일부를 다음과 같이 개정한다.\n\n제63조제2항제4호 중 “만 6세 이하의 초등학교 취학 전 자 \n\n제63조(휴직) ① (생 략)\n\n제63조(휴직) ① (현행과 같음)\n\n ② 공무원이 다음 각 호의 어\n\n ② ----- \n\n-\n\n의 경우에는 대통령령으로 정\n\n-----...'

자연어 처리 관련 라이브러리

자연어 처리 개요 정리하기

- 단어 표현
- 텍스트 분류
- 텍스트 유사도
- 자연어 생성
- 기계 이해
- 데이터 이해하기

자연어 처리 관련 라이브러리

단어 표현

- 카운트 기반 기법
 - 특이값 분해
 - 잠재의미분석
 - Hyperspace Analogue to Language (HAL)
 - Hellinger PCA (Principal Component Analysis)
- 예측 방법
 - Word2vec
 - NNLM (Neural Network Language Model)
 - RNNLM (Recurrent Neural Network Language Model)

자연어 처리 관련 라이브러리

텍스트 분류

- 영어 텍스트 분류
- 한글 텍스트 분류

자연어 처리 관련 라이브러리

챗봇 만들기

- 데이터 소개
- 데이터 분석
- 분석 실습
- Transformer 모델

자연어 처리 관련 라이브러리

챗봇 만들기 - 데이터 소개

Chatbot_data.

Chatbot_data_for_Korean v1.0

Data description.

1. 챗봇 트레이닝용 문답 페어 11,876개
2. 일상상대반서 0, 이별(부정) 1, 사랑(긍정) 2로 레이블링

Quick peek.

ChatbotData

Q	A	label
12시 땡!	하루가 또 가네요.	0
1지망 학교 떨어졌어	위로해 드립니다.	0
3박4일 놀러가고 싶다	여행은 언제나 좋죠.	0
3박4일 정도 놀러가고 싶다	여행은 언제나 좋죠.	0
PPL 심하네	눈살이 찌푸러지죠.	0
SD카드 망가졌어	다시 새로 사는 게 마음 편해요.	0
SD카드 안돼	다시 새로 사는 게 마음 편해요.	0

관련 코드 : [Korean Language Model for Wellness Conversation](#)

- 이 곳에 저장된 데이터를 만들면서 누군가에게 위로가 되는 모델이 나오면 좋겠다고 생각했었는데 제 생각보다 더 잘 만든 모델이 있어서 링크 걸어 둡니다.
- 부족한 데이터지만 이곳에 저장된 데이터와 [AI 허브 정신건강 상담 데이터](#)를 토대로 만들었다고 합니다.

https://github.com/songys/Chatbot_data

자연어 처리 관련 라이브러리

챗봇 만들기 - 데이터 분석

“자연어 처리 공부는 매우 어렵다”



음절 : “자”, “연”, “어”, ...

어절 : “자연어”, “처리”, “공부는”, ..

형태소 : “자연어”, “처리”, “공부”, “는”, ...



길이에 대한 분석 진행

자연어 처리 관련 라이브러리

챗봇 만들기 - 데이터 분석

지난 시간에 배운 내용 (seq2seq model)과 한글 전처리 방법을 활용하여
챗봇 데이터 학습 해보기

자연어 처리 관련 라이브러리

Transformer 모델

Attention Is All You Need

Ashish Vaswani*
Google Brain
avaswani@google.com

Noam Shazeer*
Google Brain
noam@google.com

Niki Parmar*
Google Research
nikip@google.com

Jakob Uszkoreit*
Google Research
usz@google.com

Llion Jones*
Google Research
llion@google.com

Aidan N. Gomez* †
University of Toronto
aidan@cs.toronto.edu

Lukasz Kaiser*
Google Brain
lukaszkaizer@google.com

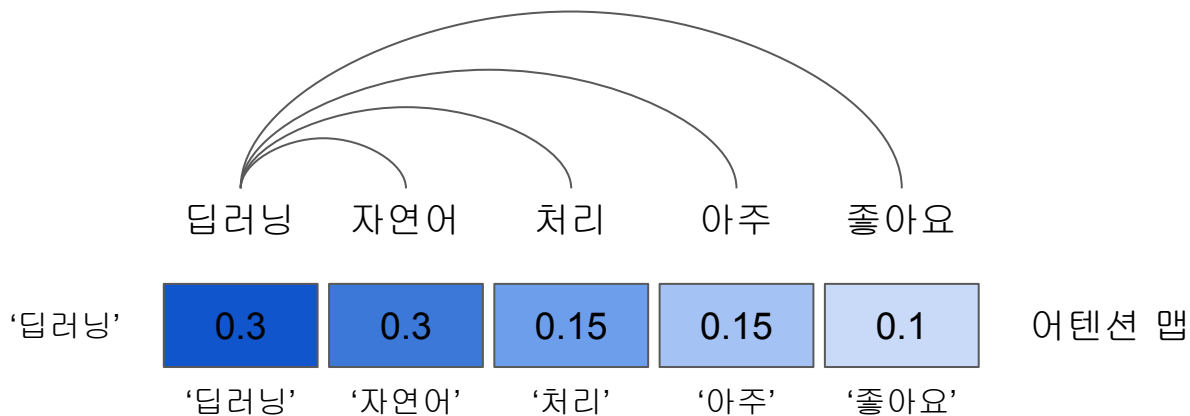
Illia Polosukhin* ‡
illia.polosukhin@gmail.com

2017년에 발표된 논문, RNN 기반으로 구성된 기존 모델과 다르게
단순히 어텐션 구조만으로 전체 모델을 만든 연구

자연어 처리 관련 라이브러리

Transformer 모델

사전 지식 - 셀프 어텐션 모델

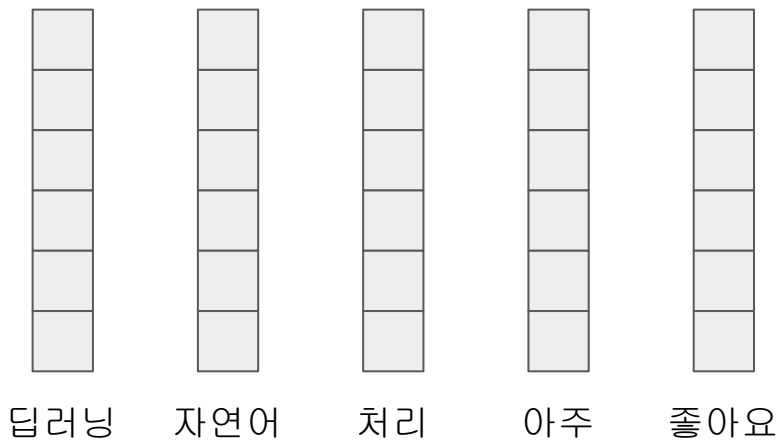


자연어 처리 관련 라이브러리

Transformer 모델

사전 지식 - 셀프 어텐션 모델

예시 문장에 대한 단어 벡터

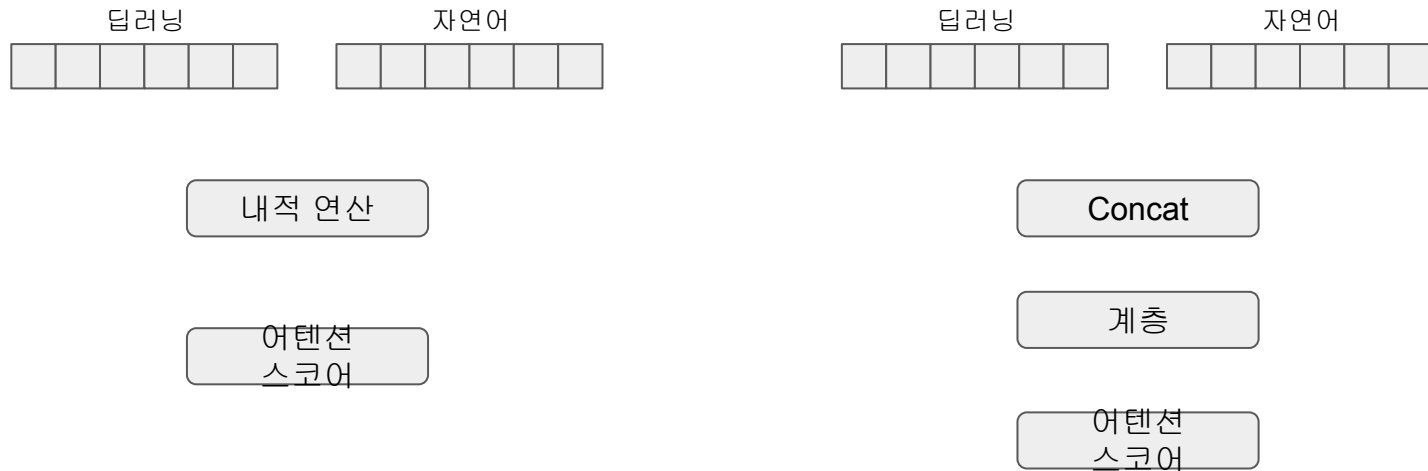


자연어 처리 관련 라이브러리

Transformer 모델

사전 지식 - 셀프 어텐션 모델

유사도 점수를 구하는 방법



자연어 처리 관련 라이브러리

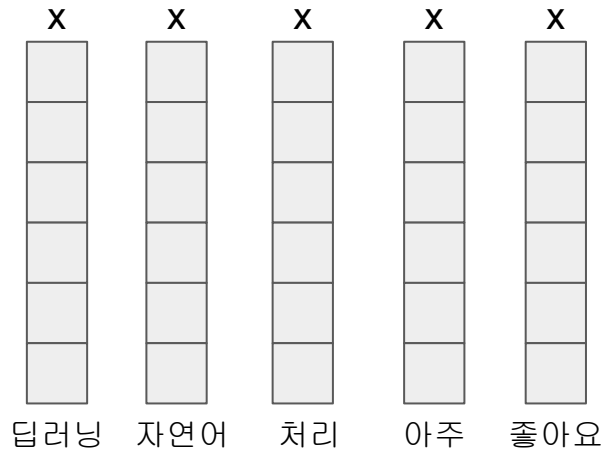
Transformer 모델

사전 지식 - 셀프 어텐션 모델

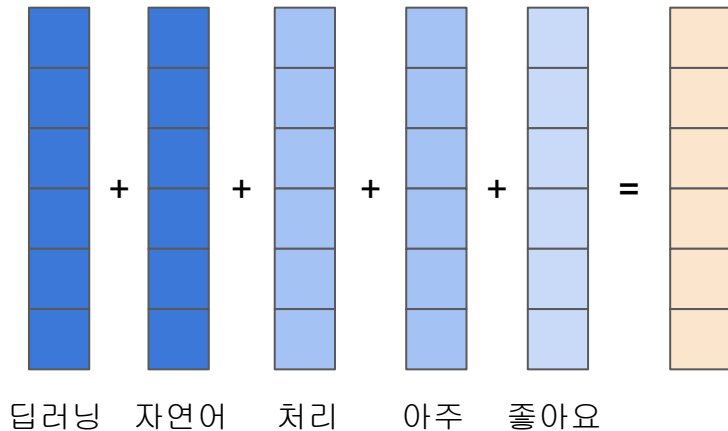
딥러닝	자연어	처리	아주	좋아요
25	25	23	22	21
Softmax				
0.3	0.3	0.15	0.15	0.1

softmax 함수로 어텐션
스코어를 확률 값으로 표현

어텐션 스코어와 각 단어
벡터와 상수곱 연산을 수행



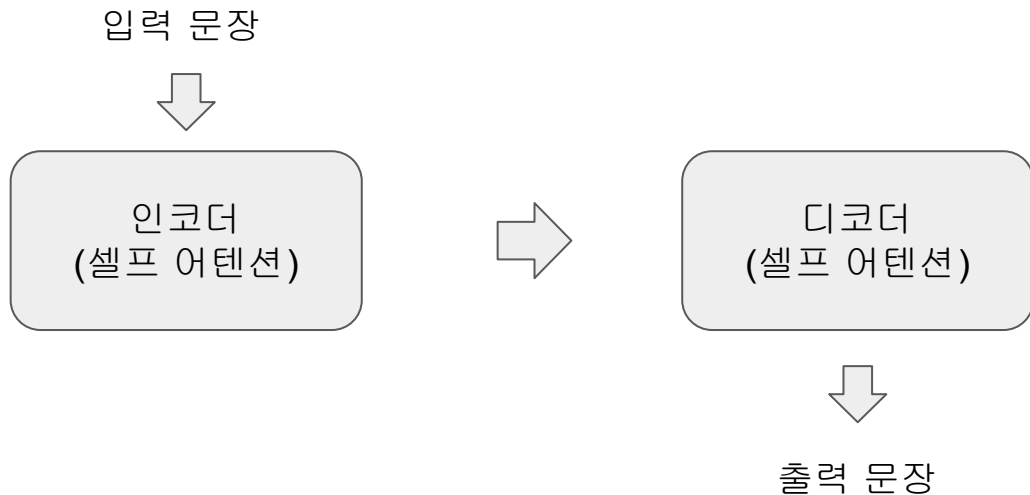
상수곱의 결과 단어
벡터들을 더해서 컨텍스트
벡터를 구함



문장에 대한 단어 벡터와 어텐션 맵과의 가중합

자연어 처리 관련 라이브러리

Transformer 모델



자연어 처리 관련 라이브러리

Transformer 모델

Transformer 모델 구현을 위해서는 아래 모듈 구현이 필요하다

- 포지션 인코딩 (Position Encoding)
- 멀티 헤드 어텐션 (Multi-head attention)
- 포지션-와이즈 피드 포워드 네트워크 (Position-wise feed forward network)
- 리지듀얼 커넥션 (Residual connection)

자연어 처리 관련 라이브러리

Transformer 모델

Transformer 모델 구현을 위해서는 아래 4가지 모듈이

- Positional Encoding
- Multi-head attention
- Position-wise feed forward network
- Residual connection

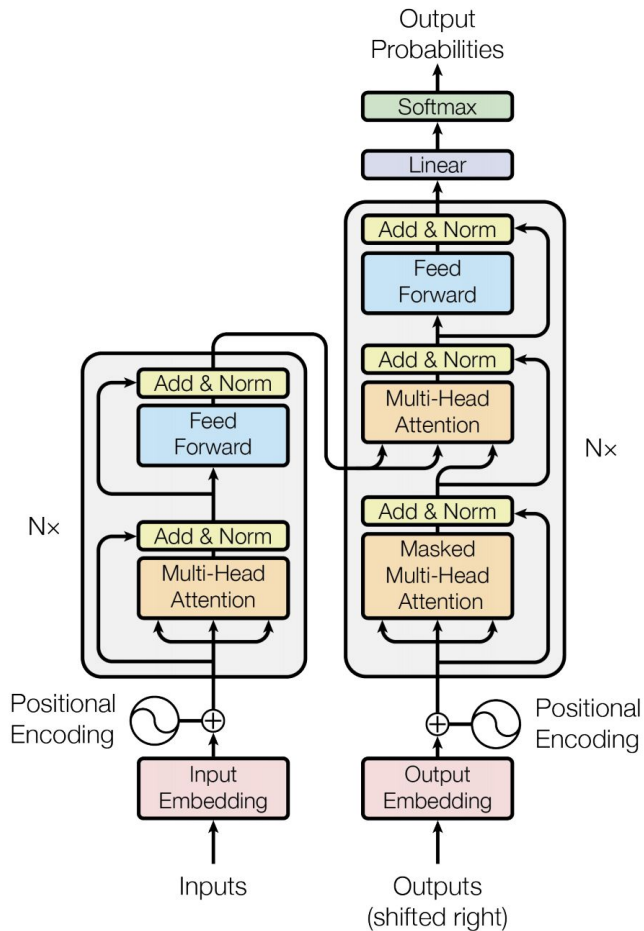


Figure 1: The Transformer - model architecture.

자연어 처리 관련 라이브러리

Transformer 모델 - Positional Encoding

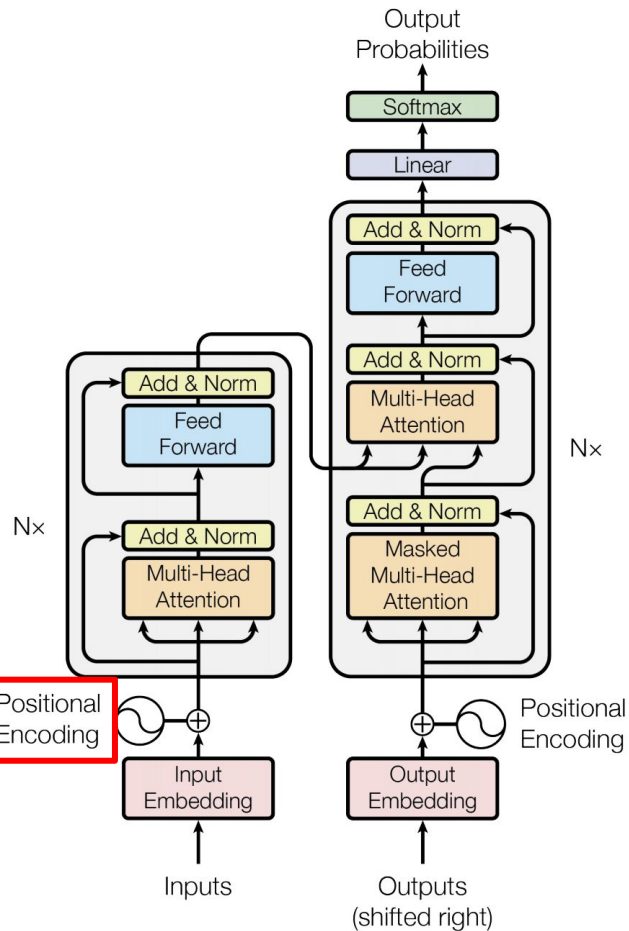
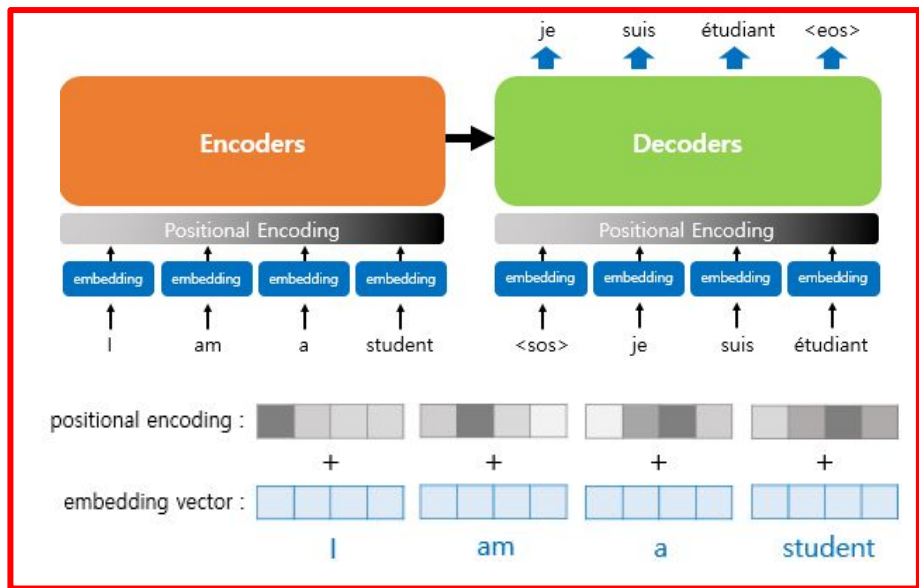


Figure 1: The Transformer - model architecture.

자연어 처리 관련 라이브러리

Transformer 모델 - 멀티 헤드 어텐션

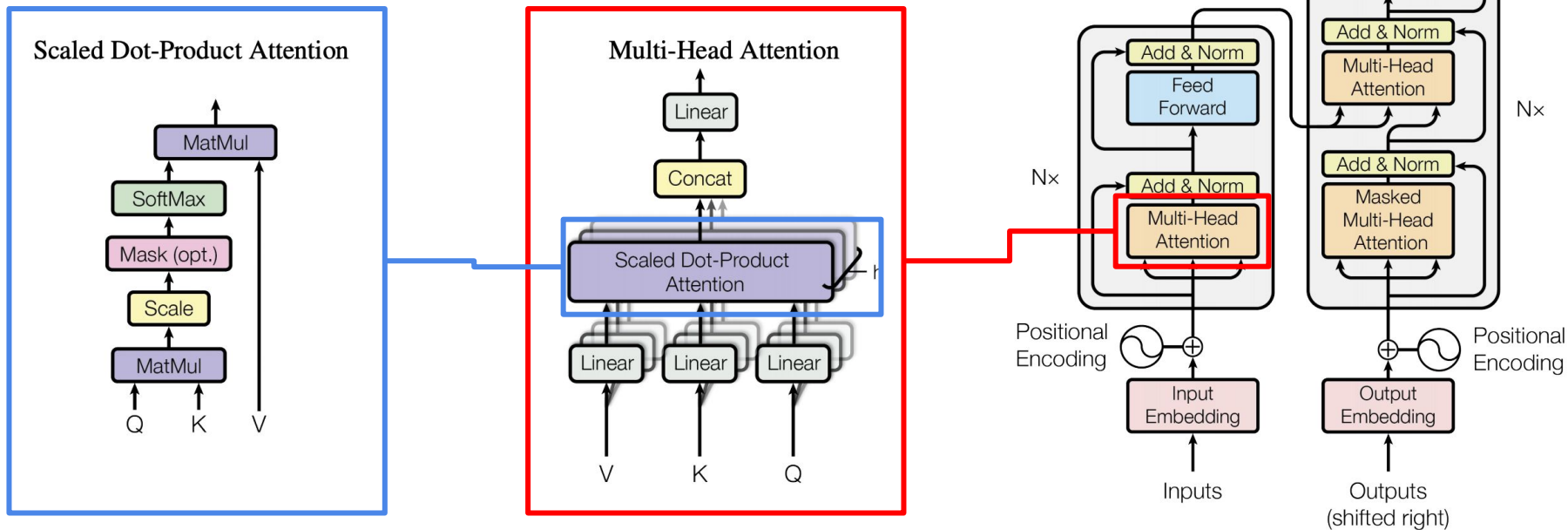
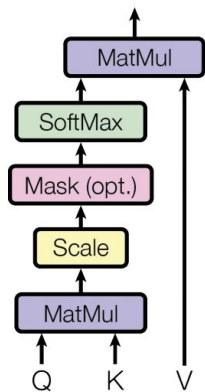


Figure 1: The Transformer - model architecture.

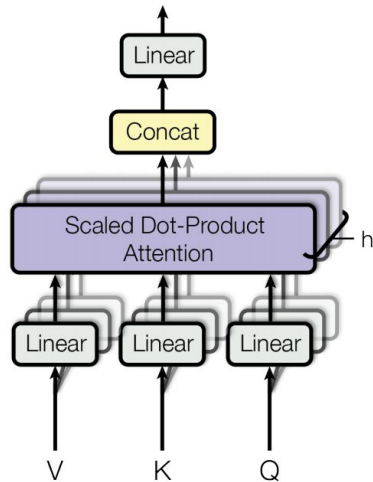
자연어 처리 관련 라이브러리

Transformer 모델 - 멀티 헤드 어텐션

Scaled Dot-Product Attention



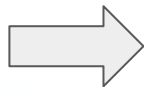
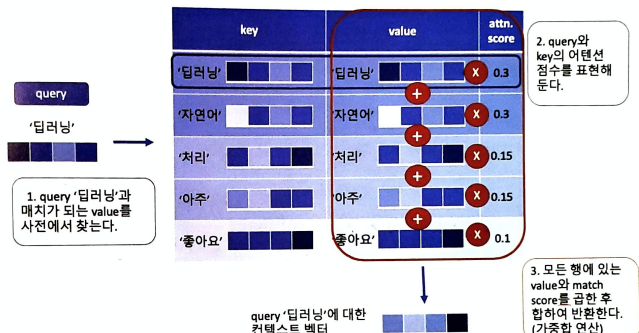
Multi-Head Attention



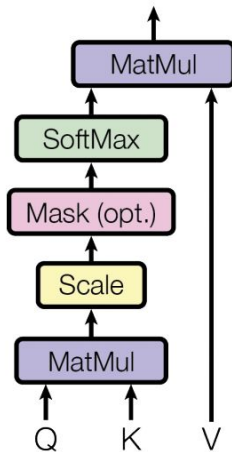
자연어 처리 관련 라이브러리

Transformer 모델 - 멀티 헤드 어텐션

셀프 어텐션



Scaled Dot-Product Attention



자연어 처리 관련 라이브러리

Transformer 모델 - 멀티 헤드 어텐션

```
def scaled_dot_product_attention(q, k, v, mask):
    """Calculate the attention weights.
    q, k, v must have matching leading dimensions.
    k, v must have matching penultimate dimension, i.e.: seq_len_k = seq_len_v.
    The mask has different shapes depending on its type(padding or look ahead)
    but it must be broadcastable for addition.

    Args:
        q: query shape == (... , seq_len_q, depth)
        k: key shape == (... , seq_len_k, depth)
        v: value shape == (... , seq_len_v, depth_v)
        mask: Float tensor with shape broadcastable
              to (... , seq_len_q, seq_len_k). Defaults to None.

    Returns:
        output, attention_weights
    """

    matmul_qk = tf.matmul(q, k, transpose_b=True) # (... , seq_len_q, seq_len_k)

    # scale matmul_qk
    dk = tf.cast(tf.shape(k)[-1], tf.float32)
    scaled_attention_logits = matmul_qk / tf.math.sqrt(dk)

    # add the mask to the scaled tensor.
    if mask is not None:
        scaled_attention_logits += (mask * -1e9)

    # softmax is normalized on the last axis (seq_len_k) so that the scores
    # add up to 1.
    attention_weights = tf.nn.softmax(scaled_attention_logits, axis=-1) # (... , seq_len_q, seq_len_k)

    output = tf.matmul(attention_weights, v) # (... , seq_len_q, depth_v)

    return output, attention_weights
```

자연어 처리 관련 라이브러리

Transformer 모델 - Feed Forward

```
def point_wise_feed_forward_network(**kwargs):  
    return tf.keras.Sequential([  
        tf.keras.layers.Dense(kwargs['dff'], activation='relu'),  
        tf.keras.layers.Dense(kwargs['d_model']) # (batch_size,  
        1)  
    ])
```

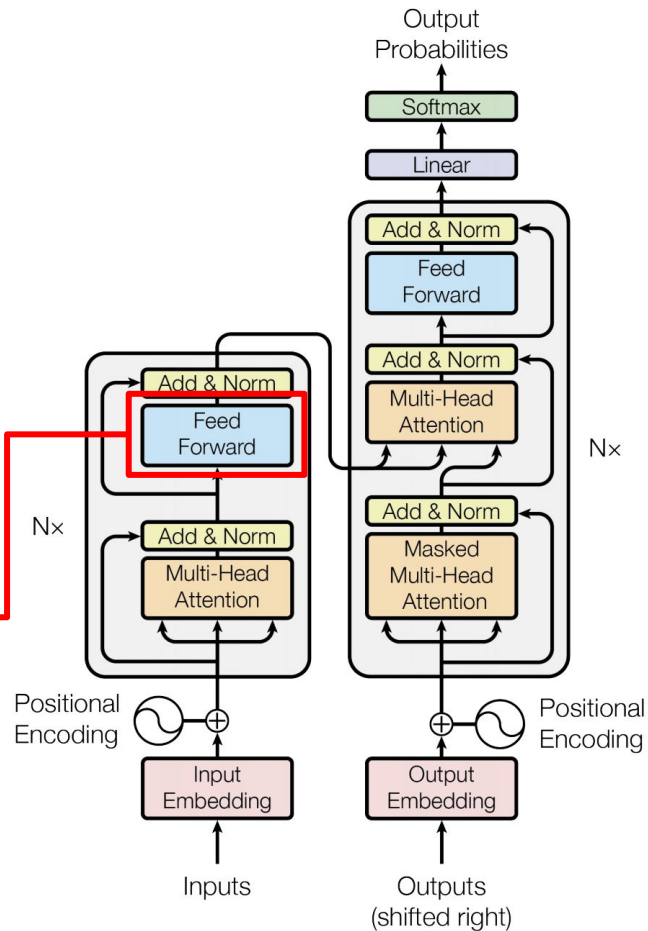
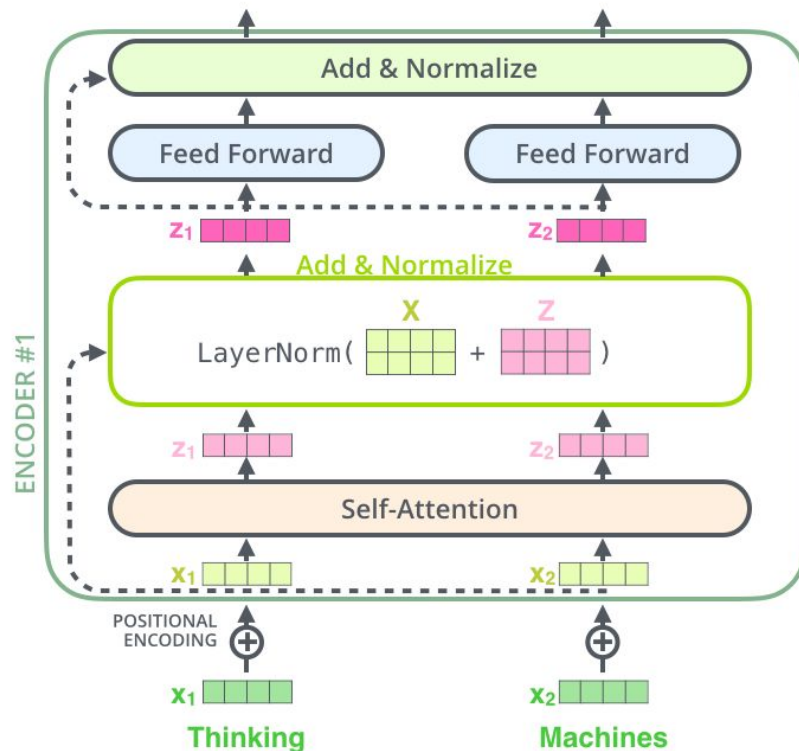
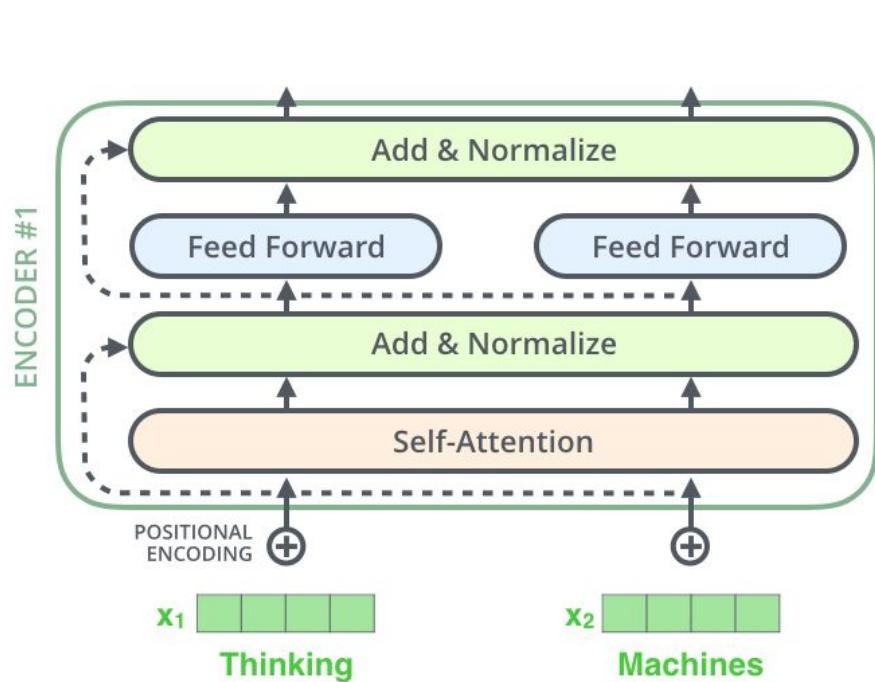


Figure 1: The Transformer - model architecture.

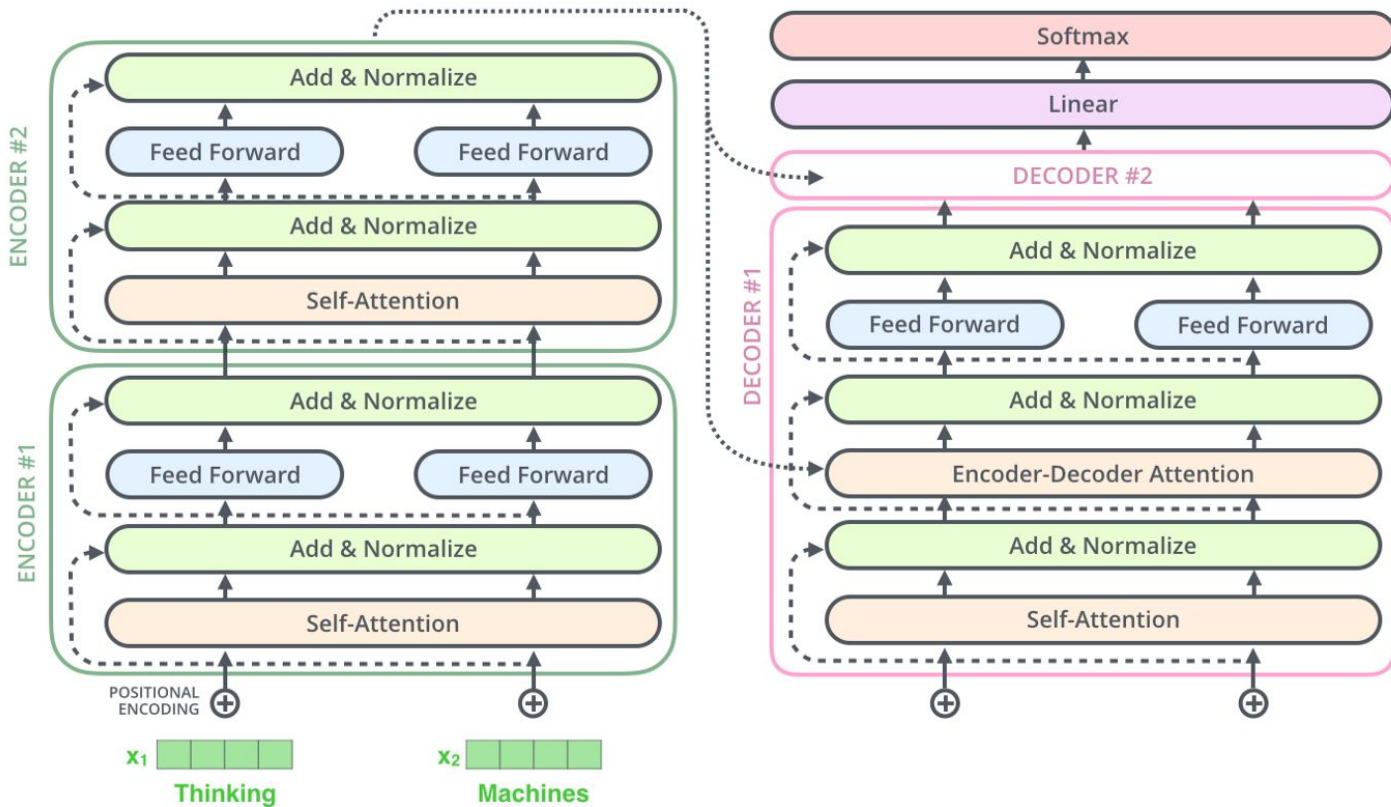
자연어 처리 관련 라이브러리

Transformer 모델 - Residual Connection



자연어 처리 관련 라이브러리

Transformer 모델 - Residual Connection



자연어 처리 관련 라이브러리

사전 학습 모델

- BERT
- BERT를 활용한 미세 조정 학습

자연어 처리 관련 라이브러리

BERT

- BERT는 Transformer가 변형된 형태
- Transformer는 이전 단어를 보고 다음 단어를 예측
- BERT는
 - 문장에 **masking**처리를 한 후 **mask**된 단어를 예측
 - 주변 단어들을 보고 **masked** 단어를 예측
 - 문장들 사이의 관계를 학습하기 위해 '다음 문장' 이라는 **label**을 추가함

자연어 처리 관련 라이브러리

BERT - Task 1

15%의 [MASK] token을 만들어 낼 때, 몇가지 추가적인 처리를 더 해주게 됩니다. 그것은 다음과 같습니다.

- 80%의 경우 : token을 [MASK]로 바꿉니다. eg., my dog is hairy -> my dog is [MASK]
- 10%의 경우 : token을 random word로 바꾸어 줍니다. eg., my dog is hariy -> my dog is apple
- 10%의 경우 : token을 원래의 단어로 그대로 놔둡니다. 이는 실제 관측된 단어에 대한 표상을 bias해주기 위해 실시합니다.

자연어 처리 관련 라이브러리

BERT - Task 2

BERT에서는 corpus에서 두 문장을 이어 붙여 이것이 원래의 corpus에서 바로 이어 붙여져 있던 문장인지를 맞추는 **binarized next sentence prediction task**를 수행합니다.

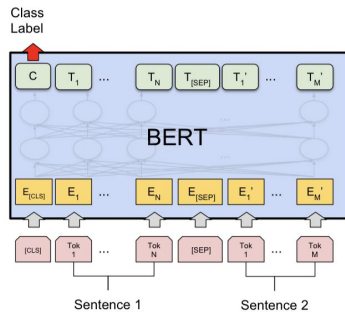
- 50% : sentence A, B가 실제 next sentence
- 50% : sentence A, B가 corpus에서 random으로 뽑힌(관계가 없는) 두 문장
- 예를 들어

Input = [CLS] the man went to [MASK] store [SEP] he bought a gallon [MASK] milk [SEP] LABEL = IsNext

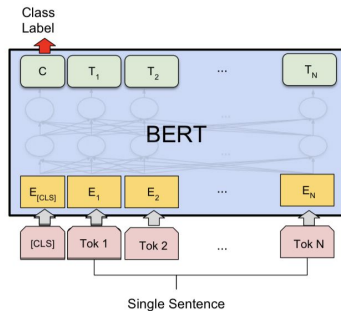
Input = [CLS] the man [MASK] to the store [SEP] penguin [MASK] are flight ##less birds [SEP] LABEL = NotNext

자연어 처리 관련 라이브러리

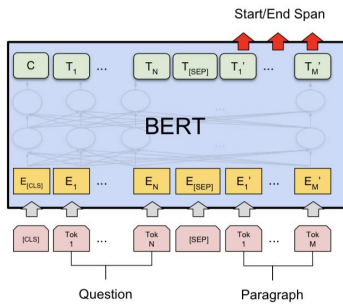
BERT - Fine tuning



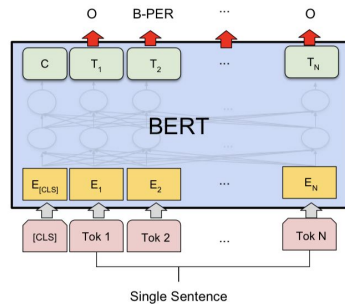
(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG



(b) Single Sentence Classification Tasks:
SST-2, CoLA



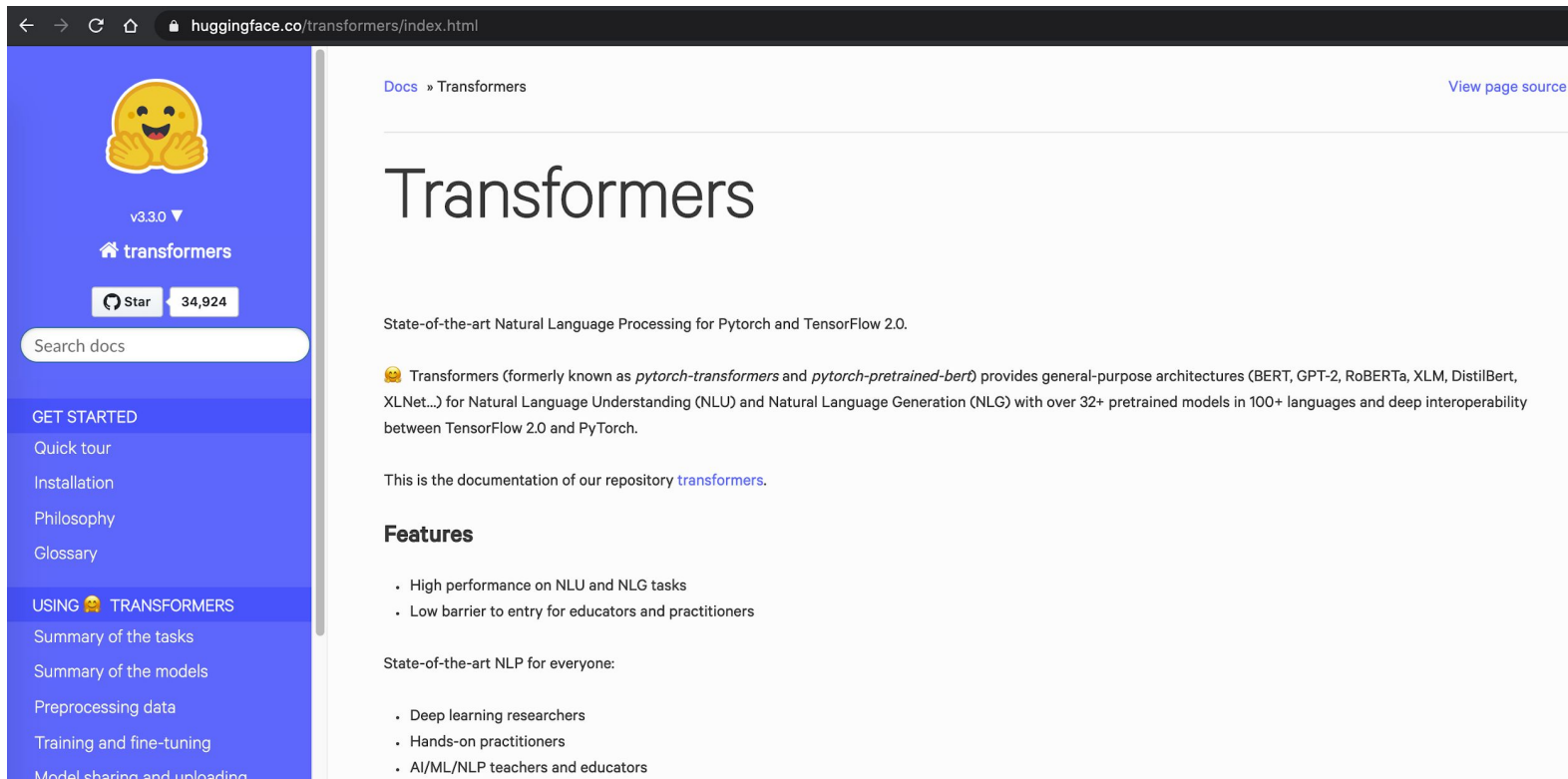
(c) Question Answering Tasks:
SQuAD v1.1



(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

자연어 처리 관련 라이브러리

BERT



The screenshot shows the Hugging Face Transformers documentation page. The left sidebar is blue and contains a navigation menu with the following items: a home icon, 'transformers' with a house icon, a 'Star' button showing '34,924', a 'Search docs' input field, 'GET STARTED' with sub-items 'Quick tour', 'Installation', 'Philosophy', and 'Glossary', 'USING 🧠 TRANSFORMERS' with sub-items 'Summary of the tasks', 'Summary of the models', 'Preprocessing data', 'Training and fine-tuning', and 'Model sharing and uploading'. The main content area is white and has a breadcrumb 'Docs » Transformers' and a 'View page source' link. The title 'Transformers' is in large black font. Below it is the subtitle 'State-of-the-art Natural Language Processing for Pytorch and TensorFlow 2.0.' followed by a paragraph describing the library. A 'Features' section lists two bullet points. At the bottom, another section titled 'State-of-the-art NLP for everyone:' lists three bullet points.

huggingface.co/transformers/index.html

Docs » Transformers [View page source](#)

Transformers

State-of-the-art Natural Language Processing for Pytorch and TensorFlow 2.0.

🧠 Transformers (formerly known as *pytorch-transformers* and *pytorch-pretrained-bert*) provides general-purpose architectures (BERT, GPT-2, RoBERTa, XLM, DistilBert, XLNet...) for Natural Language Understanding (NLU) and Natural Language Generation (NLG) with over 32+ pretrained models in 100+ languages and deep interoperability between TensorFlow 2.0 and PyTorch.

This is the documentation of our repository [transformers](#).

Features

- High performance on NLU and NLG tasks
- Low barrier to entry for educators and practitioners

State-of-the-art NLP for everyone:

- Deep learning researchers
- Hands-on practitioners
- AI/ML/NLP teachers and educators

<https://huggingface.co/transformers/index.html>