자연어처리

그림출처 : https://bit.ly/2P0ZlBi

목 차

1. 자언여 처리

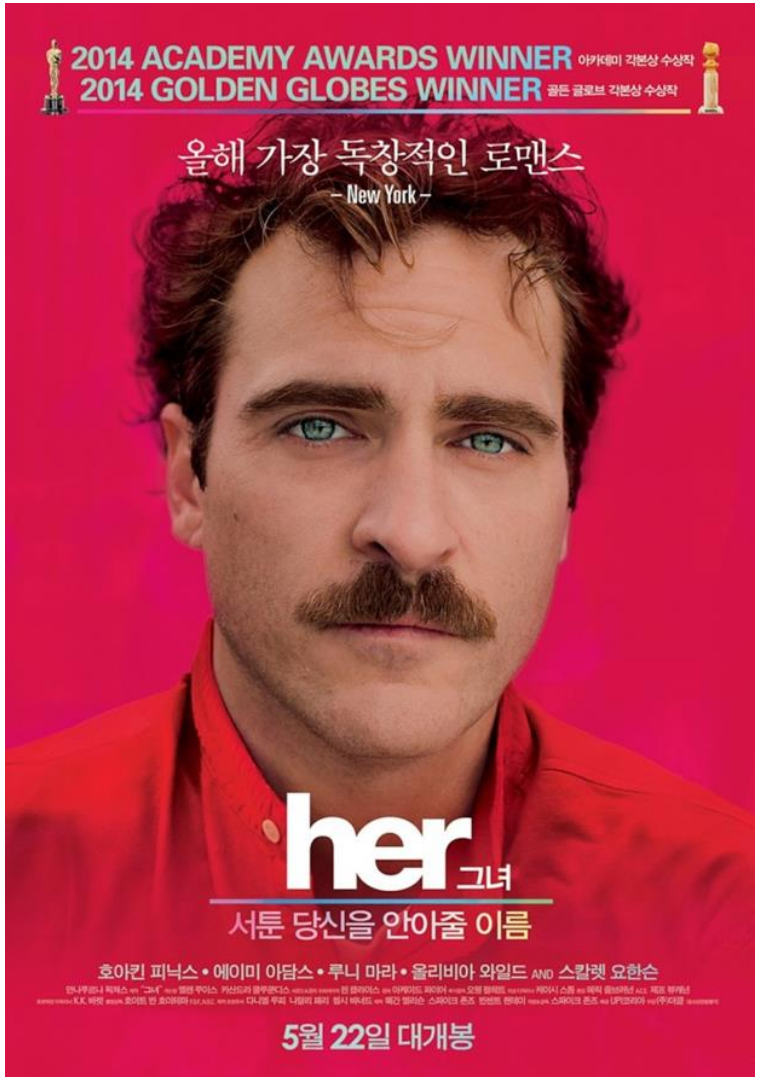
2.임베딩

3. 언어 모델

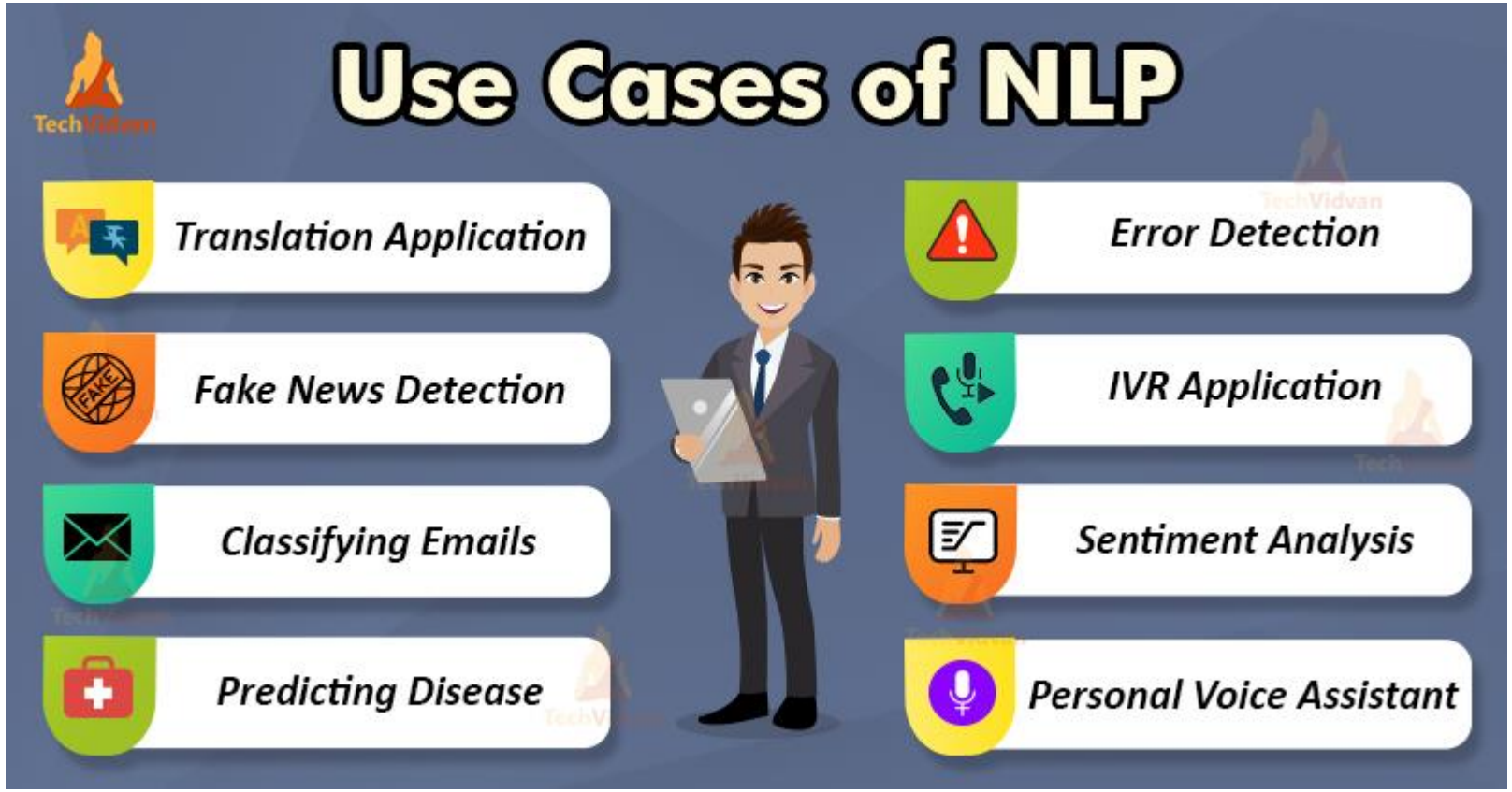
4. BERT

1. 자연어 처리

NLP(Natural Language Processing)

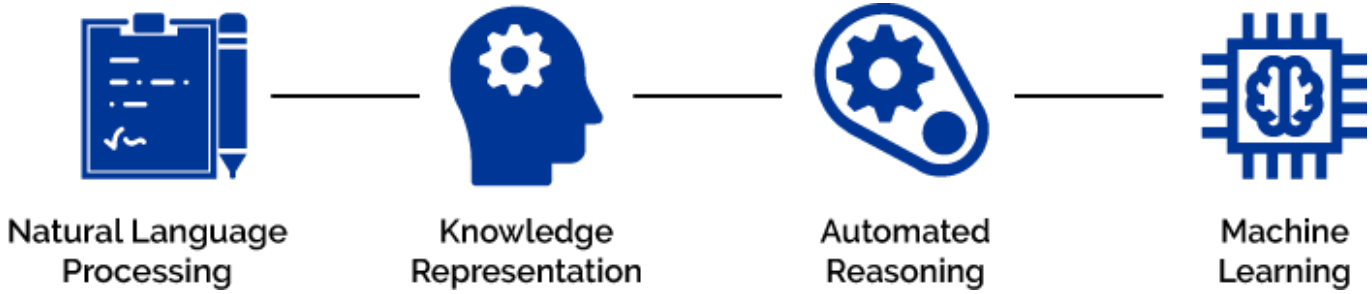


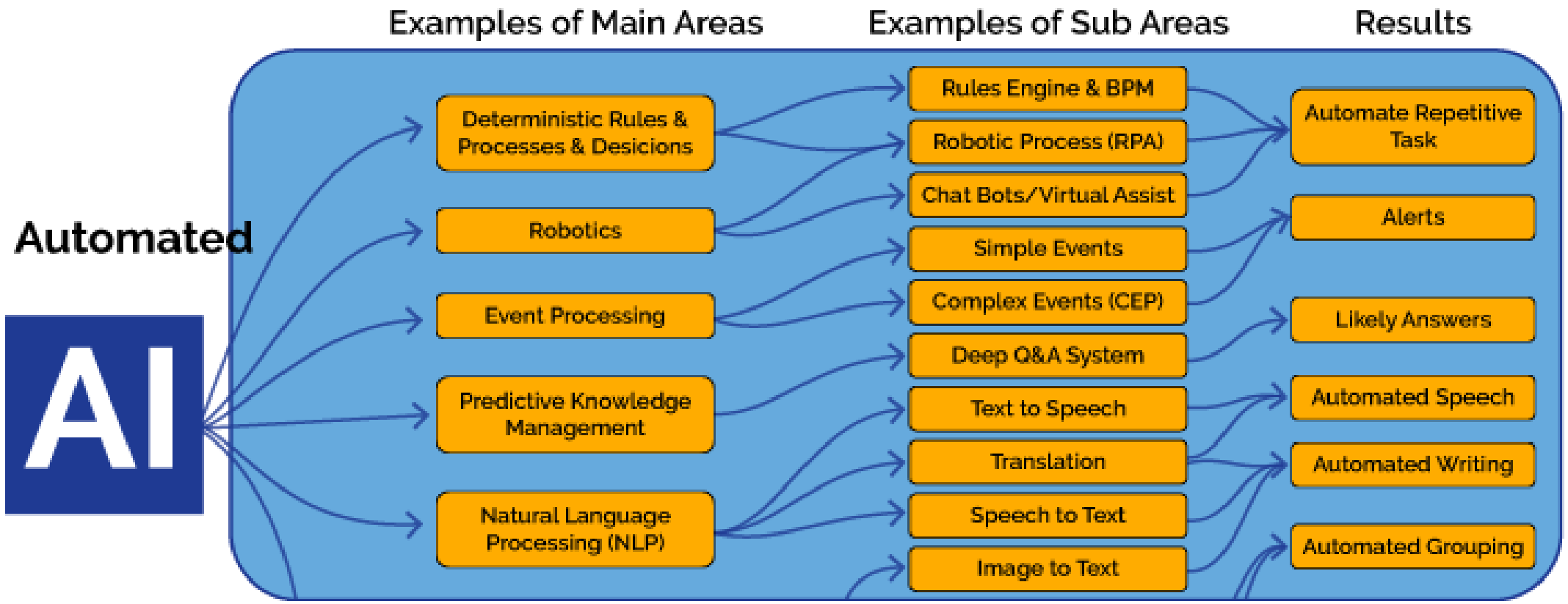
Use Cases of NLP



출처 : https://techvidvan.com/tutorials/natural-language-processing-nlp/

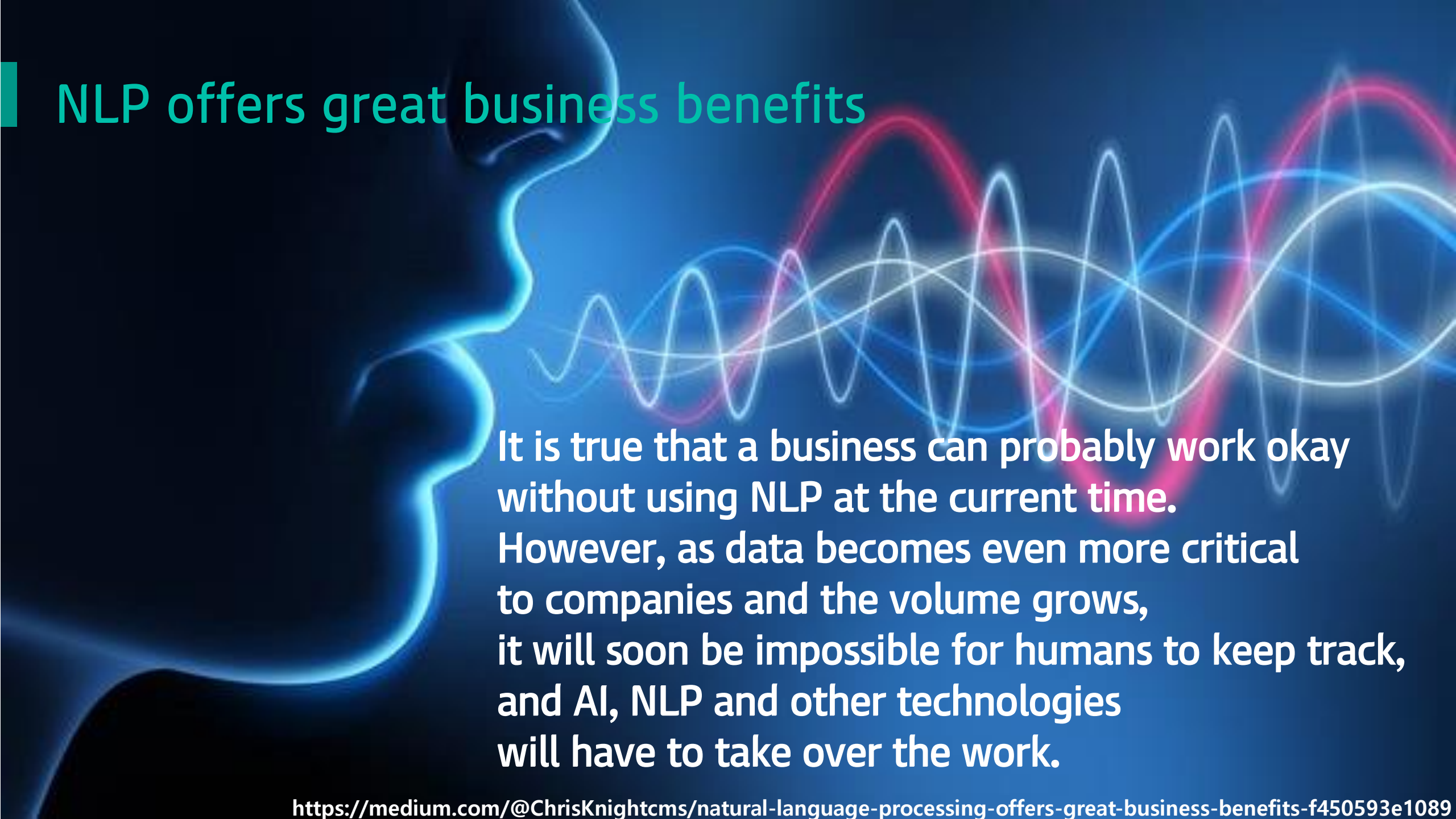
NLP makes AI SYSTEM Enabled





출처 : https://www.upwork.com/hiring/for-clients/artificial-intelligence-and-natural-language-processing-in-big-data/

NLP offers great business benefits



It is true that a business can probably work okay without using NLP at the current time.
However, as data becomes even more critical to companies and the volume grows, it will soon be impossible for humans to keep track, and AI, NLP and other technologies will have to take over the work.

https://medium.com/@ChrisKnightcms/natural-language-processing-offers-great-business-benefits-f450593e1089

자연어 처리

자연어 처리는 컴퓨터가 인간의 언어를 이해하고 모사할 수 있도록 연구하고 구현하는 AI 주요분야입니다.
어떻게 자연어를 컴퓨터에게 인식시킬 수 있을까요?

■ 단어 표현

- 문자 이진화 값 : 아스키코드, 유니코드

  '언' : 1100010110111000
  '어' : 1100010110110100

  } **언어적인 특성이 전혀 없습니다.**

- 언어적인 특성을 반영하여 단어를 수치화 하여야 합니다. : 벡터로 표시(Word Embedding)

■ 자연어 처리 핵심 문제

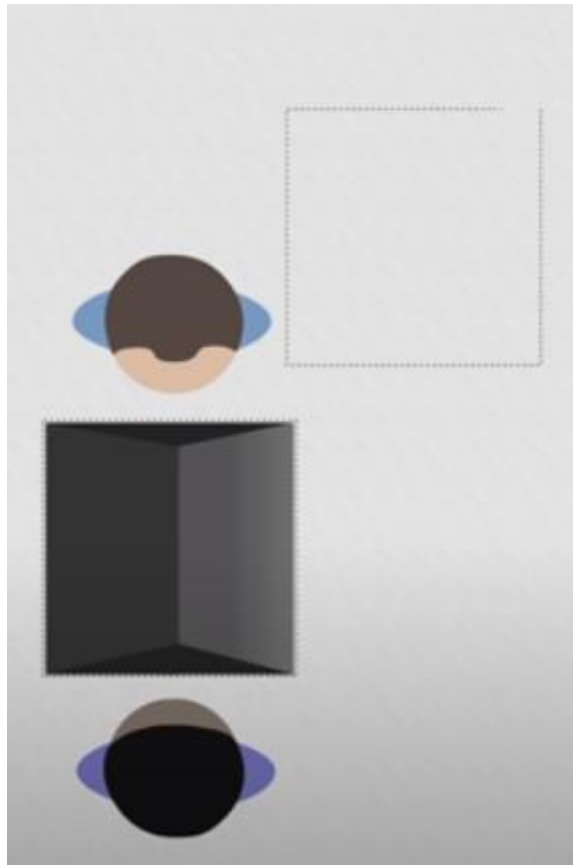
① 텍스트 분류 : 스팸메일 분류, 감정 분류, 뉴스 기사 분류

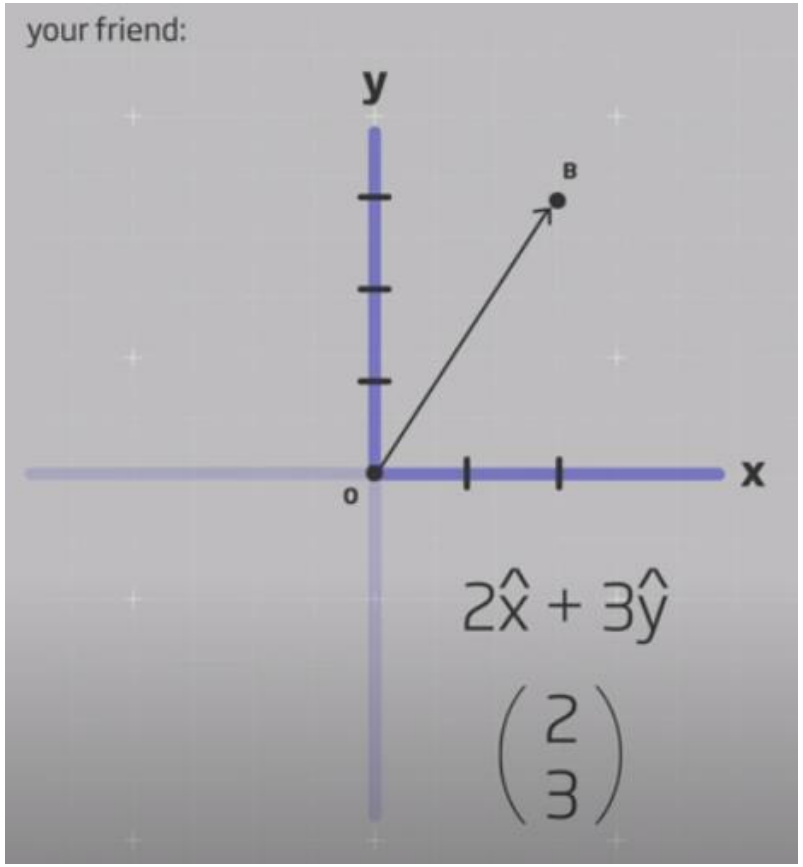
② 텍스트 유사도 : 이 노래 누가 만들었을까? 지금 나오는 노래의 작곡가는 누구야?

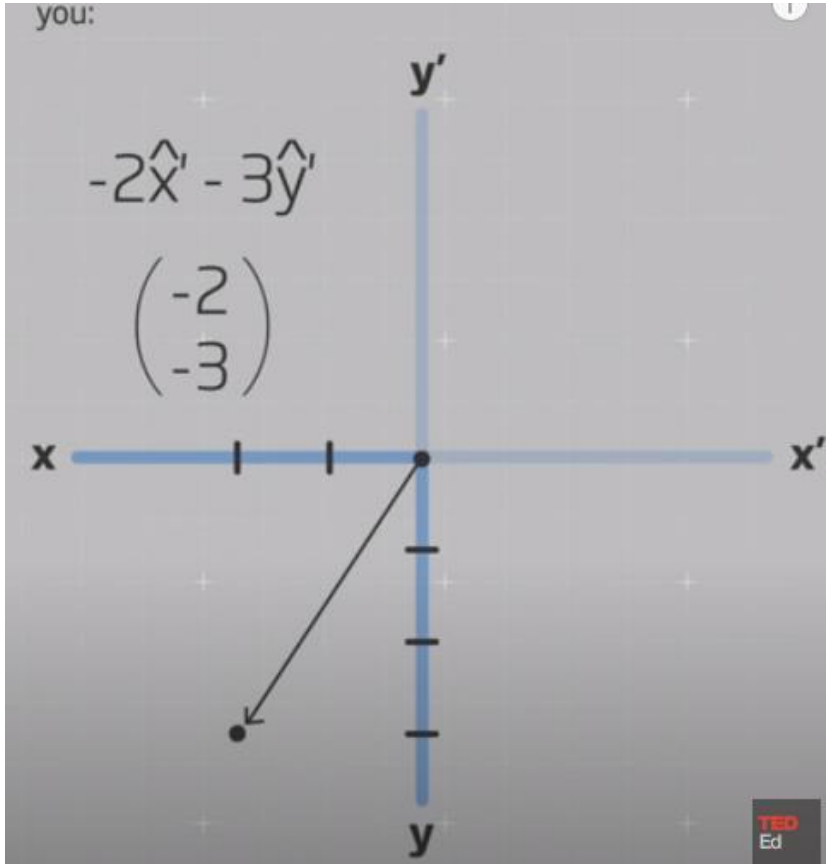
③ 텍스트 생성 : 챗봇 서비스, 신문기사 작성

④ 텍스트 이해 : 정보를 학습하고 사용자 질의에 응답

벡터(Vector)







출처 : https://youtu.be/ml4NSzCQobk

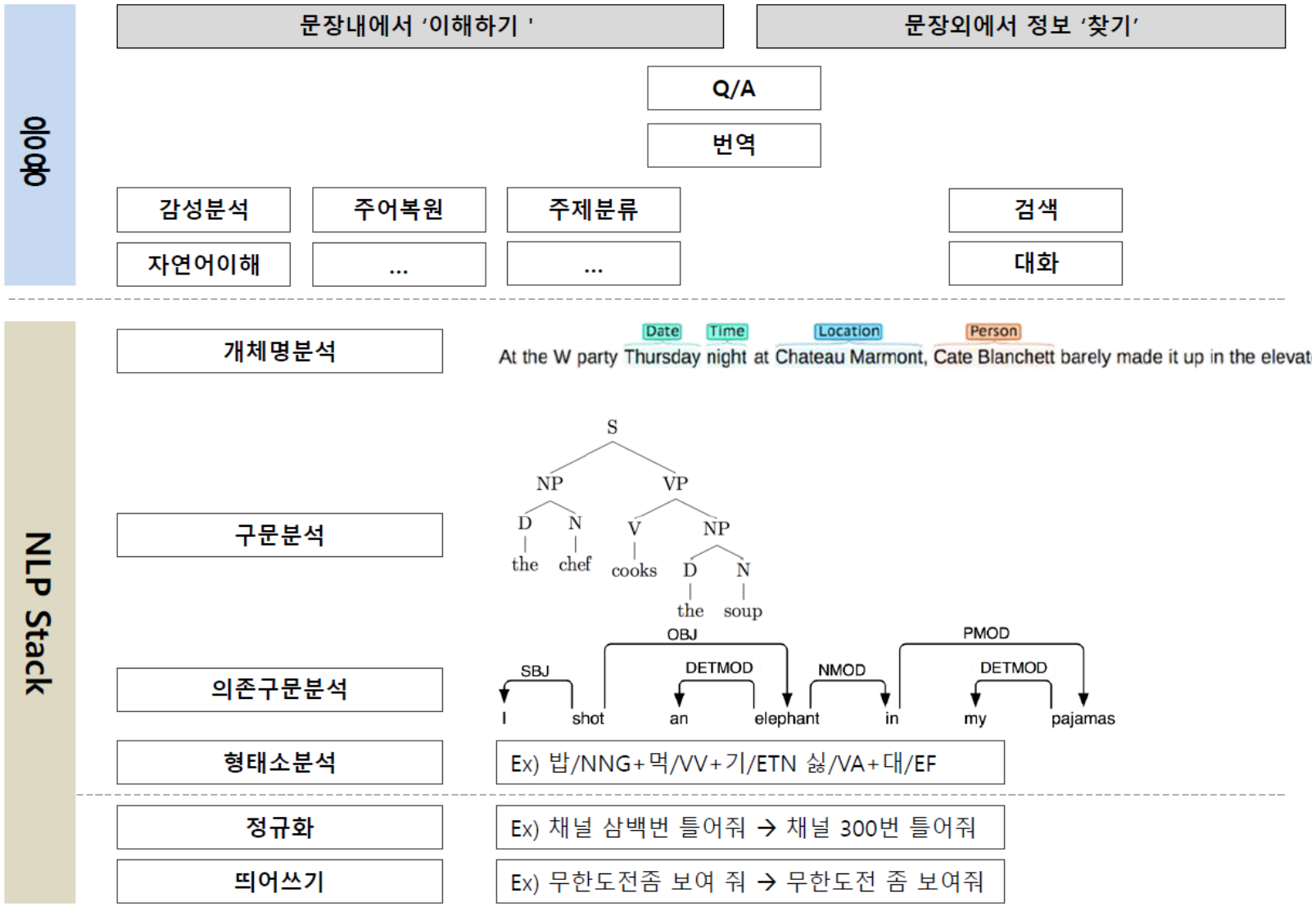
형태소 분석

형태소는 의미를 가지는 언어 단위 중 가장 작은 단위로 의미 혹은 문법적 기능의 최소단위입니다.
KoNLPy(코엔엘파이)로 형태소 분석, 품사분석 등 한국어 정보처리를 위한 파이썬 패키지입니다.

https://konlpy.org/ko/latest/


```
>>> from konlpy.tag import Kkma
>>> from konlpy.utils import pprint
>>> kkma = Kkma()
>>> pprint(kkma.sentences(u'네, 안녕하세요. 반갑습니다.'))
[네, 안녕하세요.,
 반갑습니다.]
>>> pprint(kkma.nouns(u'질문이나 건의사항은 깃헙 이슈 트래커에 남겨주세요.'))
[질문,
 건의,
 건의사항,
 사항,
 깃헙,
 이슈,
 트래커]
>>> pprint(kkma.pos(u'오류보고는 실행환경, 에러메세지와함께 설명을 최대한상세히!^^'))
[(오류, NNG),
 (보고, NNG),
 (는, JX),
 (실행, NNG),
 (환경, NNG),
 (,, SP),
 (에러, NNG),
 (메세지, NNG),
 (와, JKM),
 (함께, MAG),
 (설명, NNG),
 (을, JKO),
 (최대한, NNG),
 (상세히, MAG),
 (!, SF),
 (^^, EMO)]
```

자연어 처리 문제



출처 : Deep Learning을 이용한 자연어 처리, 정상근, 2017.11

2. 임베딩

임베딩(Embedding)

컴퓨터가 자연어를 처리할 수 있게 하려면 자연어를 계산 가능한 형식인 임베딩으로 바꿔줘야 합니다.

■ 임베딩(embedding)

- 임베딩은 자연어를 숫자의 나열인 벡터로 바꾼 결과 혹은 그 일련의 과정 전체를 가리키는 용어입니다.
- 단어나 문장 각각을 벡터로 변환해 벡터 공간에 '끼워 넣는다(embed)'는 취지에서 임베딩이라는 이름이 붙었습니다.

■ 임베딩이 중요한 이유

- 임베딩에는 말뭉치(corpus)의 의미, 문법 정보가 응축돼 있습니다.
- 임베딩은 벡터이기 때문에 사칙연산이 가능하며, 단어/문서 관련도(relevance) 역시 계산할 수 있습니다.
- 최근 임베딩이 중요해진 이유는 전이 학습(transfer learning) 때문입니다.
- 전이 학습이란 특정 문제를 풀기 위해 학습한 모델을 다른 문제를 푸는 데 재사용하는 기법입니다..
- 예컨대 대규모 말뭉치를 미리 학습(pretrain)한 임베딩을 문서 분류 모델의 입력값으로 쓰고, 해당 임베딩을 포함한 모델 전체를 문서 분류 과제를 잘할 수 있도록 업데이트(fine-tuning)하는 방식이 바로 그것입니다.
- 대규모 말뭉치를 학습시켜 임베딩을 미리 만들고(pretrain), 이후 임베딩을 포함한 모델 전체를 문서 분류 과제에 맞게 업데이트합니다(fine-tuning).

출처 : https://bit.ly/2NLIAGV

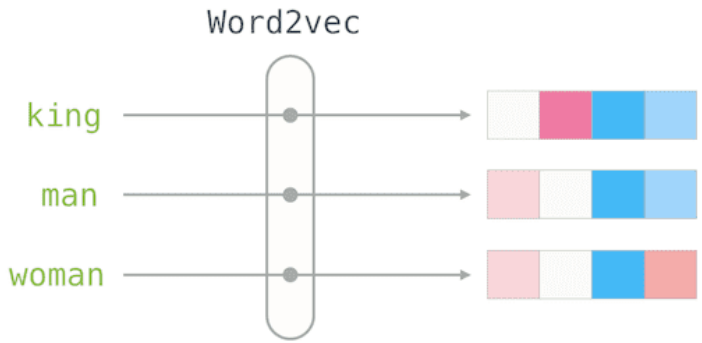
임베딩 종류

■ 단어 수준 임베딩

- NPLM : Neural Probabilistic Language Model
- Word2Vec
- FastText
- 잠재 의미 분석(LSA : Latent Semantic Analysis)
- GloVe
- Swivel

■ 문장 수준 임베딩

- LSA
- Doc2Vec
- 잠재 디리클레 할당(LDA : Latent Dirichlet Allocation)
- ELMo(Embeddings from Language Model)
- BERT(Bidirectional Encoder Representations from Transformers)



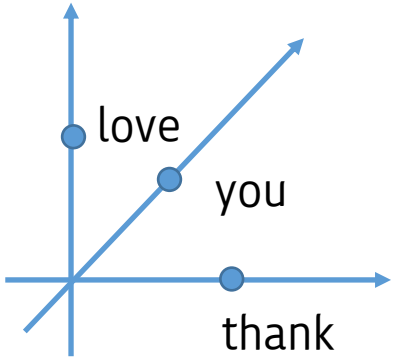
원 핫 인코딩(one hot encoding)

단어 집합의 크기를 벡터의 차원으로 하고 표현 단어의 인덱스에 1의 값을 부여하는 단어의 벡터 표현 방식입니다.

■ 원 핫 인코딩(one hot encoding)

- 단어를 범주형 변수로 변환
- 이진벡터로 표현
- 벡터의 각 차원이 단어 하나를 나타냄
- 모든 단어들의 유사도가 없음

	단어	임베딩
“thank you”	thank	[1, 0, 0]
	you	[0, 1, 0]
“love you”	love	[0, 0, 1]



■ one-hot encoding using Keras

```python
import numpy as np
from keras.utils import to_categorical
### Categorical data to be converted to numeric data
colors = ["red", "green", "yellow", "red", "blue"]

### Universal list of colors
total_colors = ["red", "green", "blue", "black", "yellow"]

### map each color to an integer
mapping = {}
for x in range(len(total_colors)):
  mapping[total_colors[x]] = x

# integer representation
for x in range(len(colors)):
  colors[x] = mapping[colors[x]]

one_hot_encode = to_categorical(colors)
```

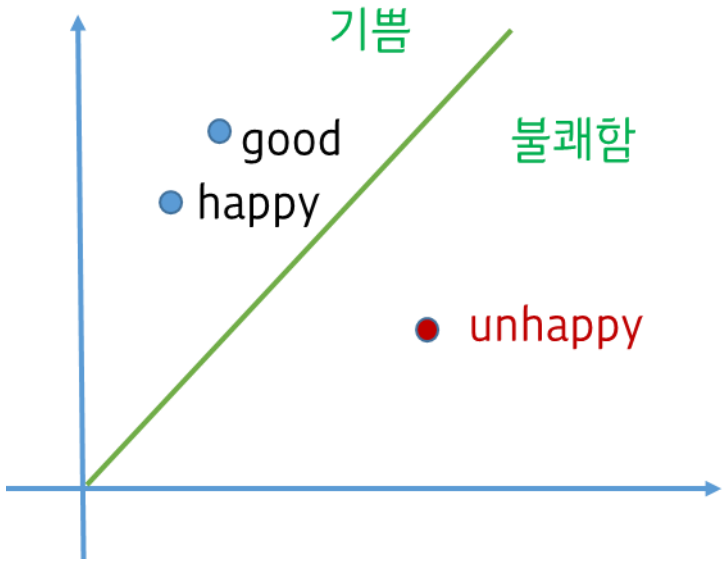
출처 : https://www.educative.io/edpresso/how-to-perform-one-hot-encoding-using-keras

Word2Vec

Word2Vec은 2013년 구글 연구팀이 발표한 기법으로 가장 널리 쓰이고 있는 단어 임베딩 모델입니다.

■ Word2Vec

- 비슷한 의미를 지닌 단어들이 벡터 공간에서 서로 이웃으로 존재하도록 변환하는 알고리즘입니다.
- 말뭉치(corpus) 데이터만 있으면, 비지도 학습으로 쉽게 Word2Vec을 구할 수 있습니다. (Label이 자동 생성되는 장점)



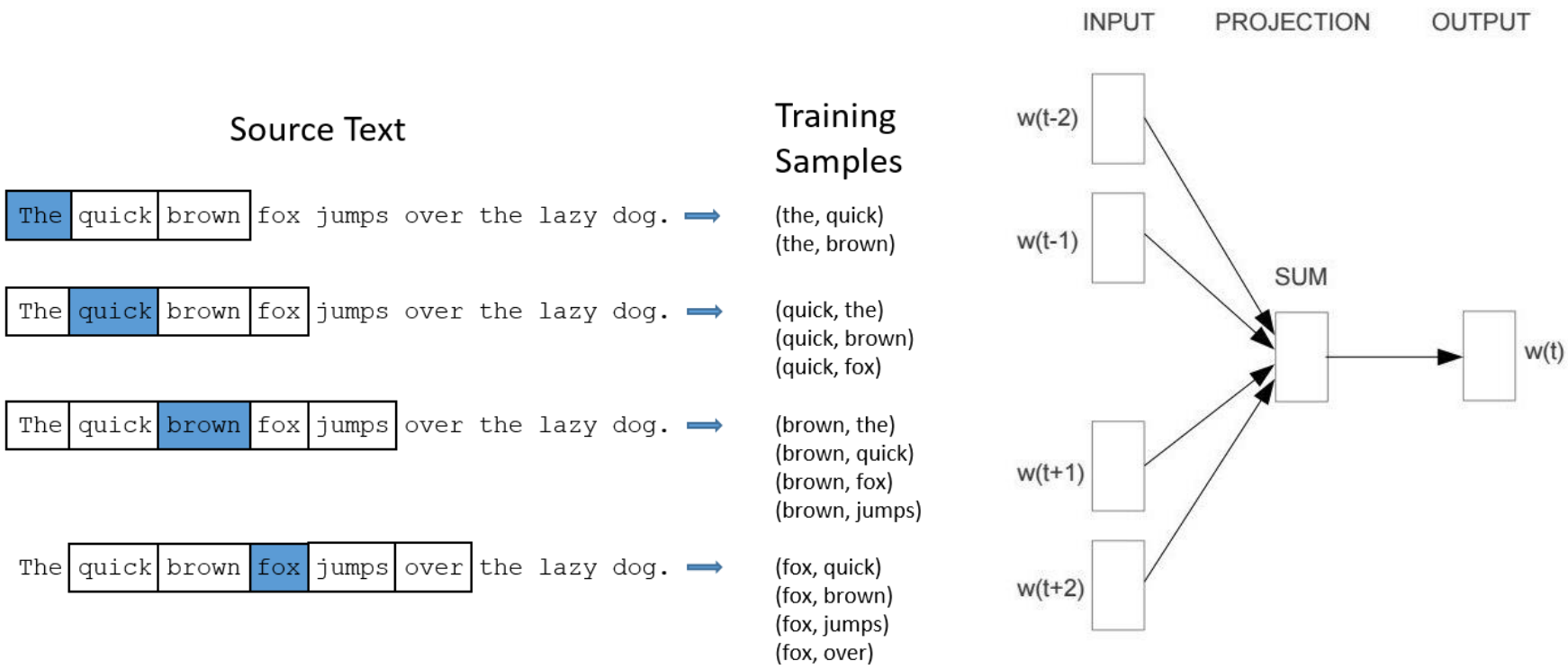
■ Word2Vec 논문 설명

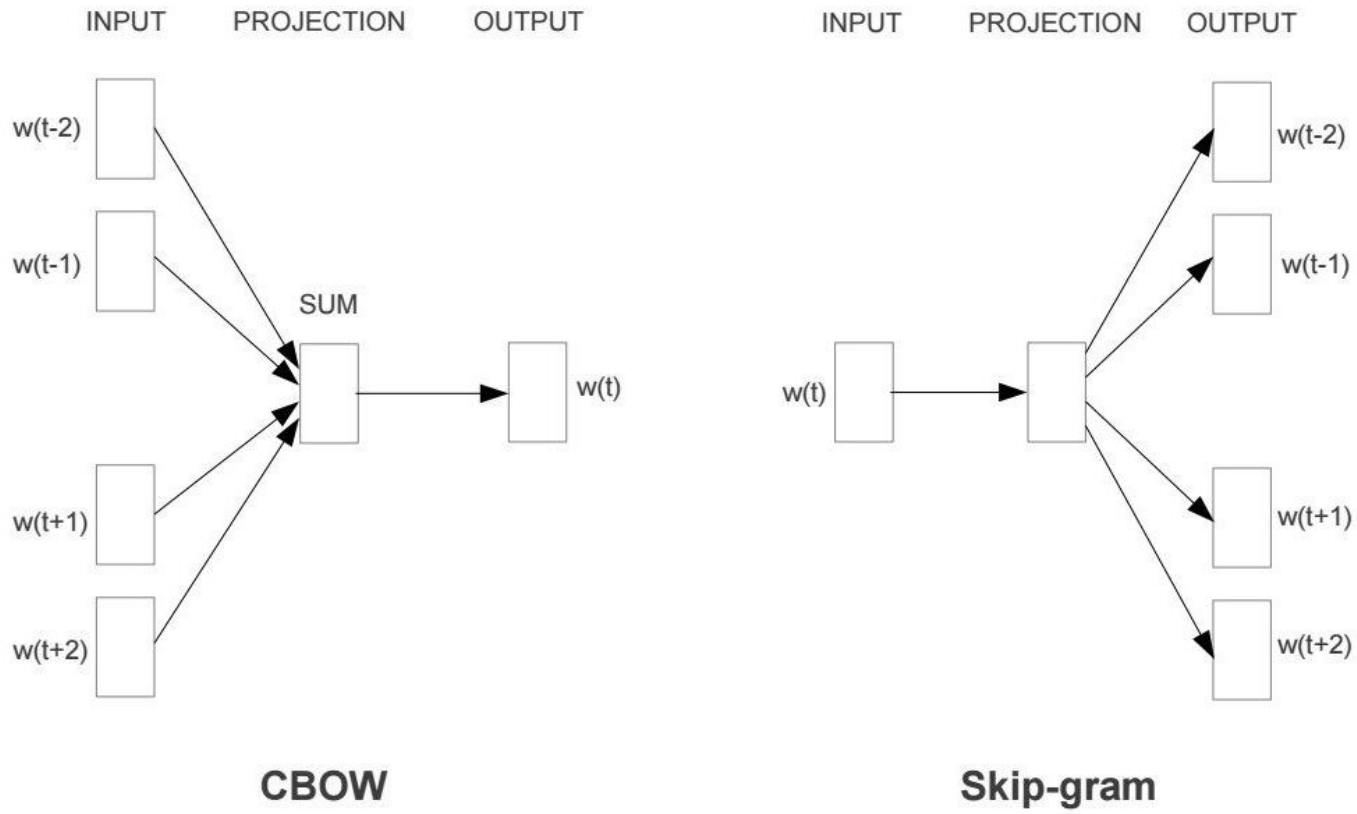
Efficient Estimation of Word Representations in Vector Space

Distributed Representations of Words and Phrases and their Compositionality

Word2Vec 모델 기본구조

Word2Vec은 예측분석을 사용하여 주변 단어와 관련하여 동시에 발생하는 단어를 가중 추측합니다. CBOW는 주변 단어들의 임베딩 벡터의 합을 이용하여 타켓단어를 예측하며, Skip-gram 은 타켓의 임베딩을 이용하여 주변 단어들을 예측하는 것입니다.



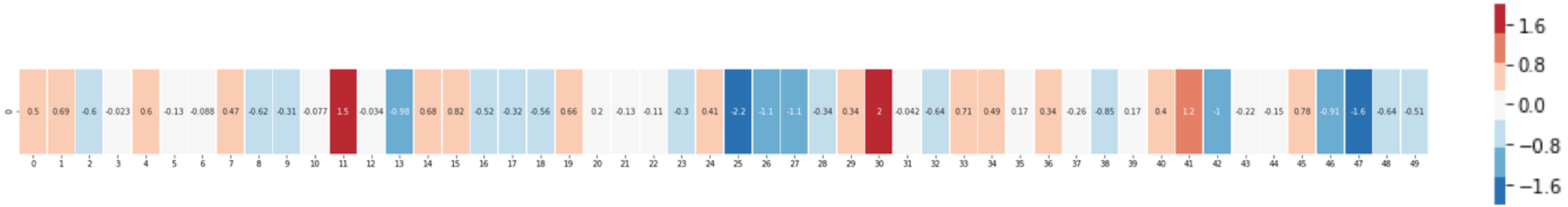


출처 : https://mubaris.com/posts/word2vec/

Word2Vec 설명

■ "king" 단어임베딩

[0.50451 , 0.68607 , −0.59517 , −0.022801, 0.60046 , −0.13498 , −0.08813 , 0.47377 , −0.61798 , −0.31012 , −0.076666, 1.493 , −0.034189, −0.98173 , 0.68229 , 0.81722 , −0.51874 , −0.31503 , −0.55809 , 0.66421 , 0.1961 , −0.13495 , −0.11476 , −0.30344 , 0.41177 , −2.223 , −1.0756 , −1.0783 , −0.34354 , 0.33505 , 1.9927 , −0.04234 , −0.64319 , 0.71125 , 0.49159 , 0.16754 , 0.34344 , −0.25663 , −0.8523 , 0.1661 , 0.40102 , 1.1685 , −1.0137 , −0.21585 , −0.15155 , 0.78321 , −0.91241 , −1.6106 , −0.64426 , −0.51042]

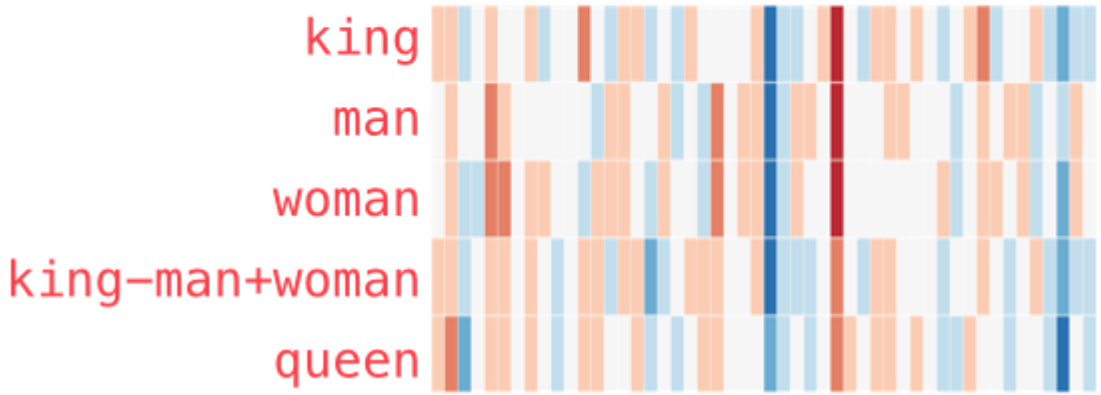


■ Analogies

```
model.most_similar(positive=["king","woman"], negative=["man"])
```

```
[('queen', 0.8523603677749634),
 ('throne', 0.7664333581924438),
 ('prince', 0.7592144012451172),
 ('daughter', 0.7473883032798767),
 ('elizabeth', 0.7460219860076904),
 ('princess', 0.7424570322036743),
 ('kingdom', 0.7337411642074585),
 ('monarch', 0.721449077129364),
 ('eldest', 0.7184862494468689),
 ('widow', 0.7099430561065674)]
```

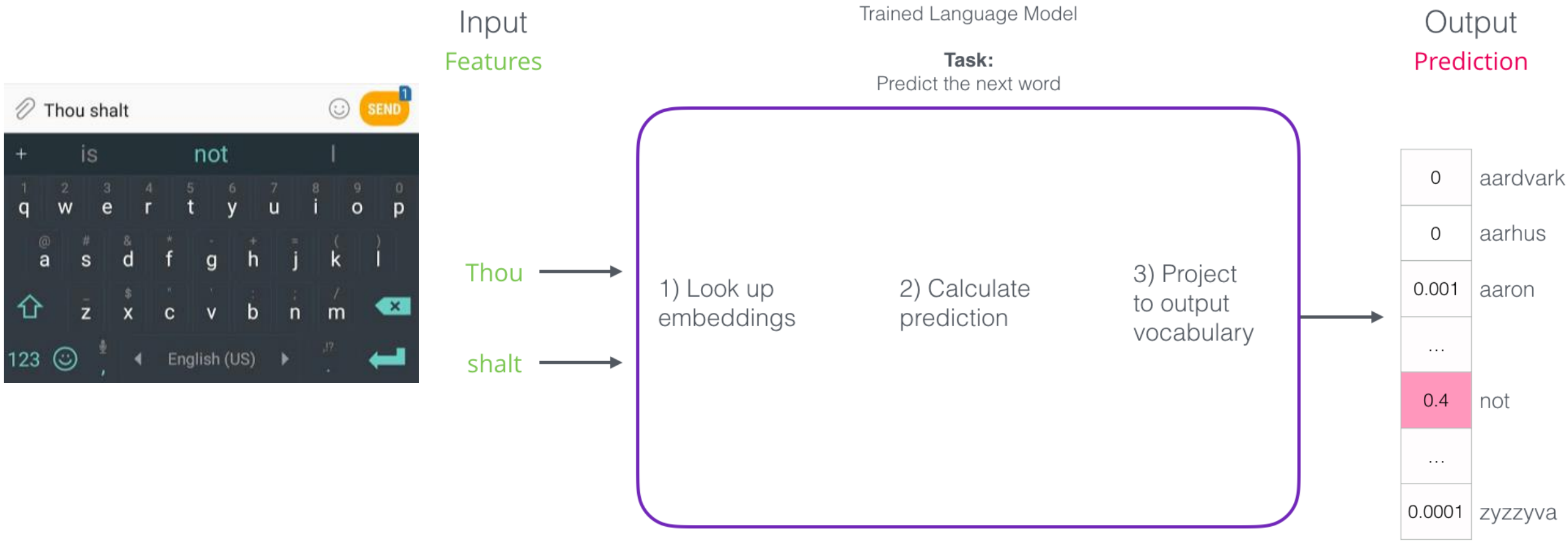
king − man + woman ~= queen



출처 : https://bit.ly/3i9picr https://jalammar.github.io/illustrated-word2vec/

Word2Vec 설명

단어 시퀀스에 대한 확률분포(probability distribution)로 뒤에 오는 단어를 예측할 수 있습니다.



출처 : https://bit.ly/3i9picr https://jalammar.github.io/illustrated-word2vec/

Word2Vec 훈련 및 단어 유사도

■ Word2Vec

```python
import gensim
from gensim.models import Word2Vec

# Word2Vec Skip-gram 모델 학습
# corpus_name = "wiki_ko_mecab.txt"
corpus_name = "korquad_mecab.txt"
model_name = corpus_name.split(".")[0]

with open(corpus_name, 'r', encoding='utf-8') as f:
    corpus = [line.strip().split(" ") for line in f.readlines()]
    model = Word2Vec(corpus, size=100, workers=4, sg=1)
    model.save(model_name)


# 코사인 유사도 상위 단어
model = gensim.models.Word2Vec.load(model_name)
result = model.wv.most_similar("대한민국")
print(result)
result = model.wv.most_similar("인공지능")
print(result)

# Pre-trained Word2Vec 모델을 사용하는 방법
model = gensim.models.Word2Vec.load("ko.bin")
result = model.wv.most_similar("대한민국")
print(result)
result = model.wv.most_similar("인공지능")
print(result)
```

■ "대한민국" 단어 유사도

[('한국', 0.6662066578865051),
('미국', 0.5982763767242432),
('우리나라', 0.5102909207344055),
('방송인', 0.4776734709739685),
('조달청', 0.4723966717720032),
('관세청', 0.4702566266059875 5),
('일본', 0.46454429626464844),
('교육원', 0.4511479139328003),
('사업단', 0.4458354115486145),
('특허청', 0.4383789598941803)]

■ "인공지능" 단어 유사도

[('컴퓨팅', 0.6520194411277771),
('가상현실', 0.6393702030181885),
('심리학', 0.63037109375),
('모델링', 0.625065267086029),
('신경망', 0.6200424432754517),
('로봇', 0.6109743118286133),
('시뮬레이션', 0.6101070642471313),
('지능', 0.6092983484268188),
('기술', 0.6087720990180969),
('기술인', 0.5957075953483582)]

영어 모델 다운로드 경로 : https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit
한국어 모델 다운로드 경로 : https://drive.google.com/file/d/0B0ZXk88koS2KbDhXdWg1Q2RydlU/view

단어 수준 임베딩의 한계점

주변 단어를 통해 학습이 이루어지기 때문에 문맥을 고려하지 못하며, 동형어 , 다의어 등에서 성능이 안 좋습니다.

10과 4의 차는 6이다. → 차[8]
표준국어대사전

차가 식으니 어서 드세요.
표준국어대사전

결혼 10년 차에 내 집을 장만했다. → 차[4]
표준국어대사전

잠이 막 들려던 차에 전화가 왔다. → 차[4]
표준국어대사전

부모와 자식 간의 세대 차가 크다. → 차[8]
표준국어대사전

100에서 49를 빼면 그 차가 얼마인가? → 차[8]
표준국어대사전

그녀는 손님에게 대접할 차를 내왔다. → 차[2]
표준국어대사전

차[4] 次 ★

1. 의존명사 '번', '차례'의 뜻을 나타내는 말.
2. 의존명사 어떠한 일을 하던 기회나 순간.
3. 의존명사 수학 방정식 따위의 차수를 이르는 말.

유의어 번[4] 차례[2]

표준국어대사전

차[2] ★

1. 명사 차나무의 어린잎을 달이거나 우린 물.
2. 명사 식물의 잎이나 뿌리, 과실 따위를 달이거나 우리거나 하여 만든 마실 것을 통틀어 이르는 말. 인삼차, 생강차, ...
3. 명사 식물 [같은 말] 차나무(차나뭇과의 상록 활엽 관목).

유의어 차나무

표준국어대사전

차[6] 車

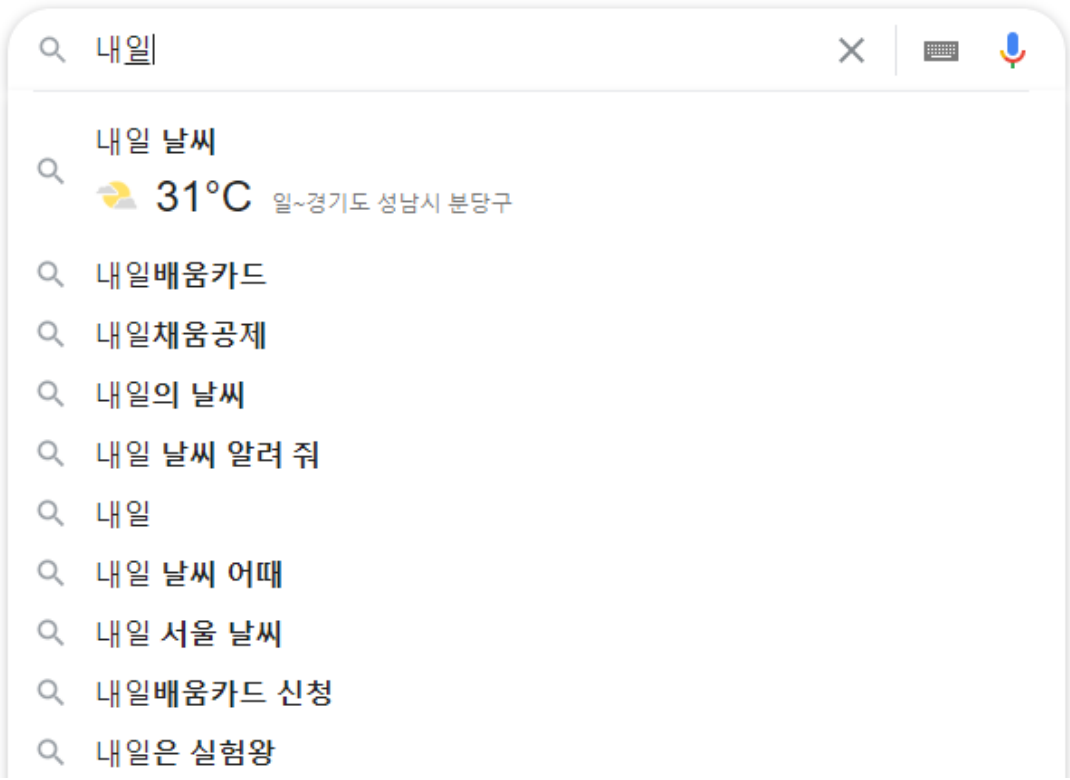
1. 명사 바퀴가 굴러서 나아가게 되어 있는, 사람이나 짐을 실어 옮기는 기관. 자동차, 기차, 전차, 우차, 마차 따위를 ...
2. 명사 화물을 '[1]'에 실어 그 분량을 세는 단위.
3. 명사 운동 '車' 자를 새긴 장기짝. 한편에 둘씩 모두 넷이 있고 일직선으로 가로나 세로로 몇 칸이든지 다닌다.

출처 : https://ko.dict.naver.com/#/search?query=%EC%B0%A8

3. 언어모델

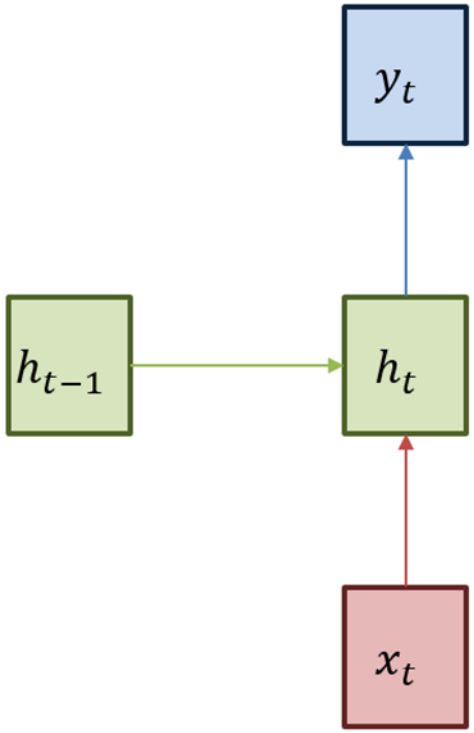
언어 모델(Language Model)

가장 자연스러운 단어 시퀀스를 찾아내는 모델로, 이전 단어들이 주어졌을 때 다음 단어를 예측하도록 하는 것입니다.



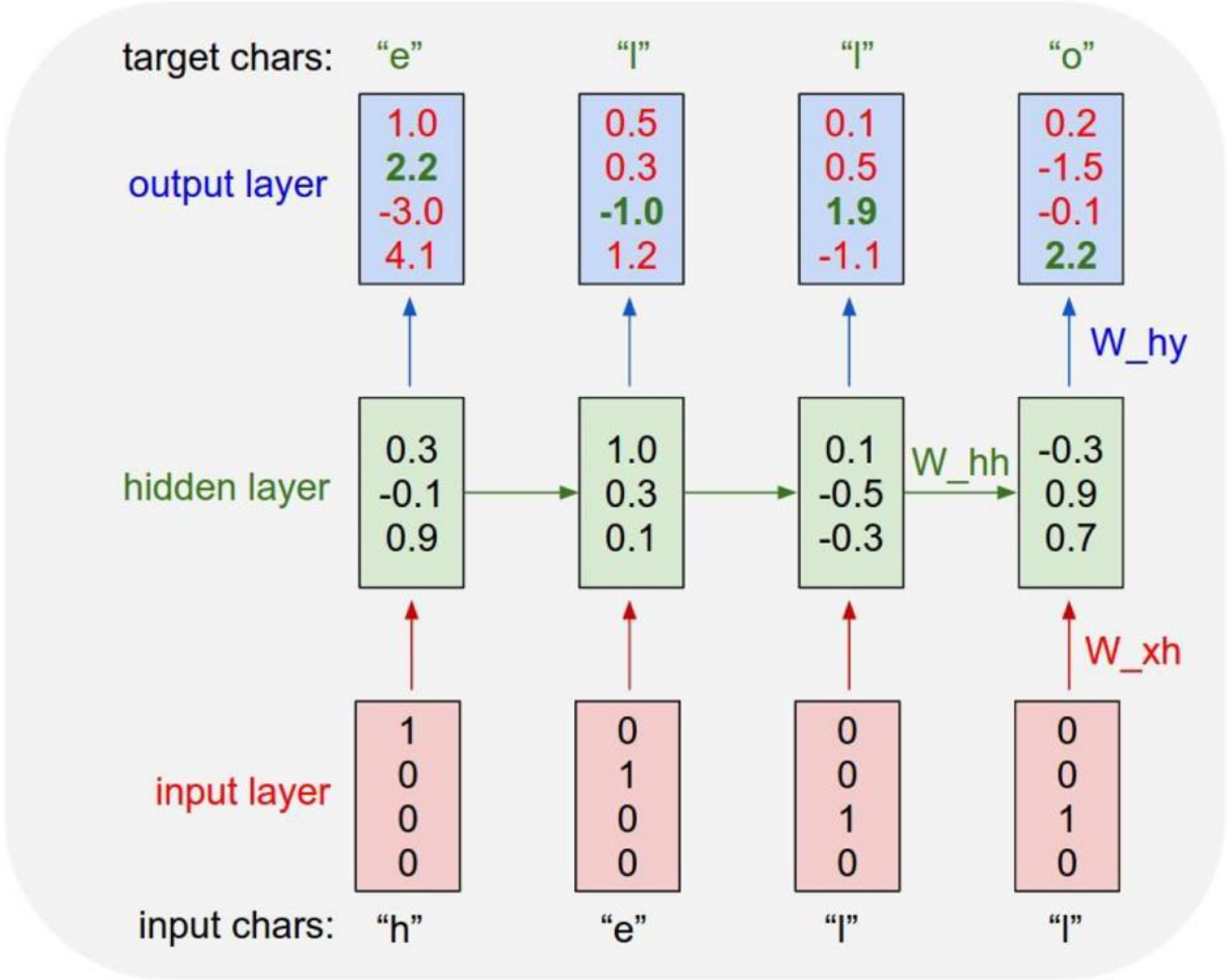
RNN 기반의 언어 모델

순환구조를 이루는 인공신경망으로 이전 state 정보가 다음 state 를 예측하는데 사용되어 시계열 데이터 처리에 특화



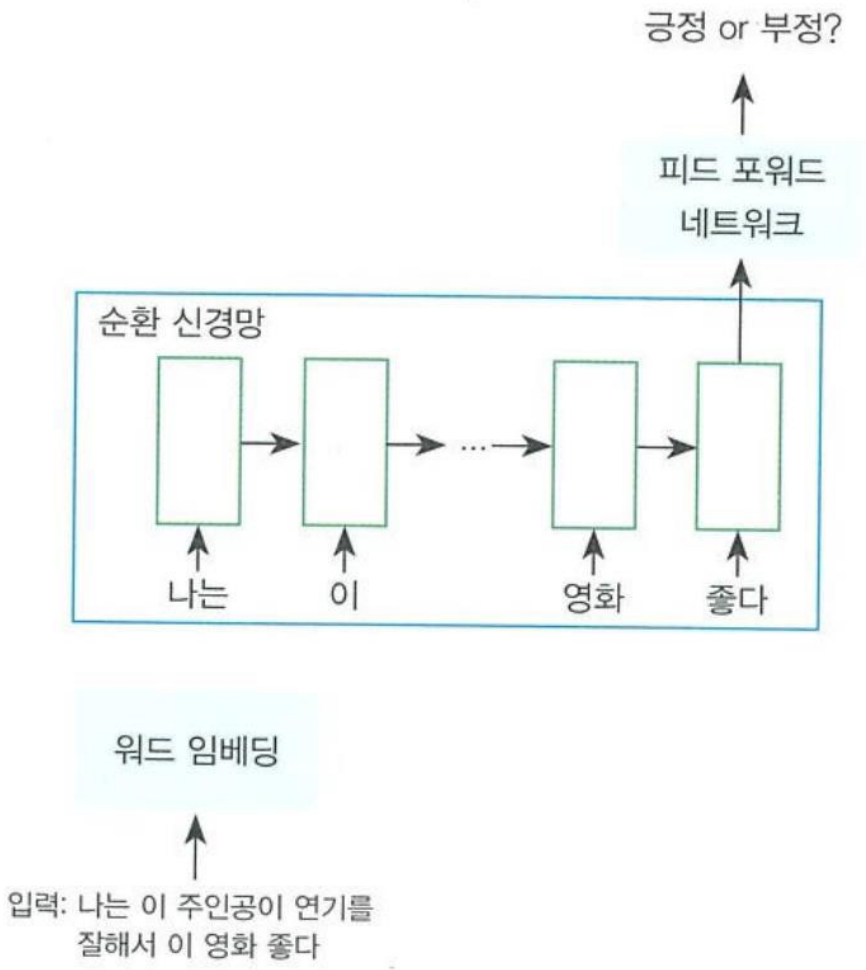
$$y_t = W_{hy} h_t + b_y$$

$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t + b_h)$$



RNN 기반 분류기 구현

마지막 출력은 앞선 단어들의 문맥 고려해서 만들어진 Context Vector로 이 값에 대해 Classification Layer를 붙이면 문장 분류를 위한 신경망 모델이 됩니다.



■ **Source Code**

https://bit.ly/2A8iZ7C

■ **TF1 버전의 코드를 TF2에서 실행하는 방법**

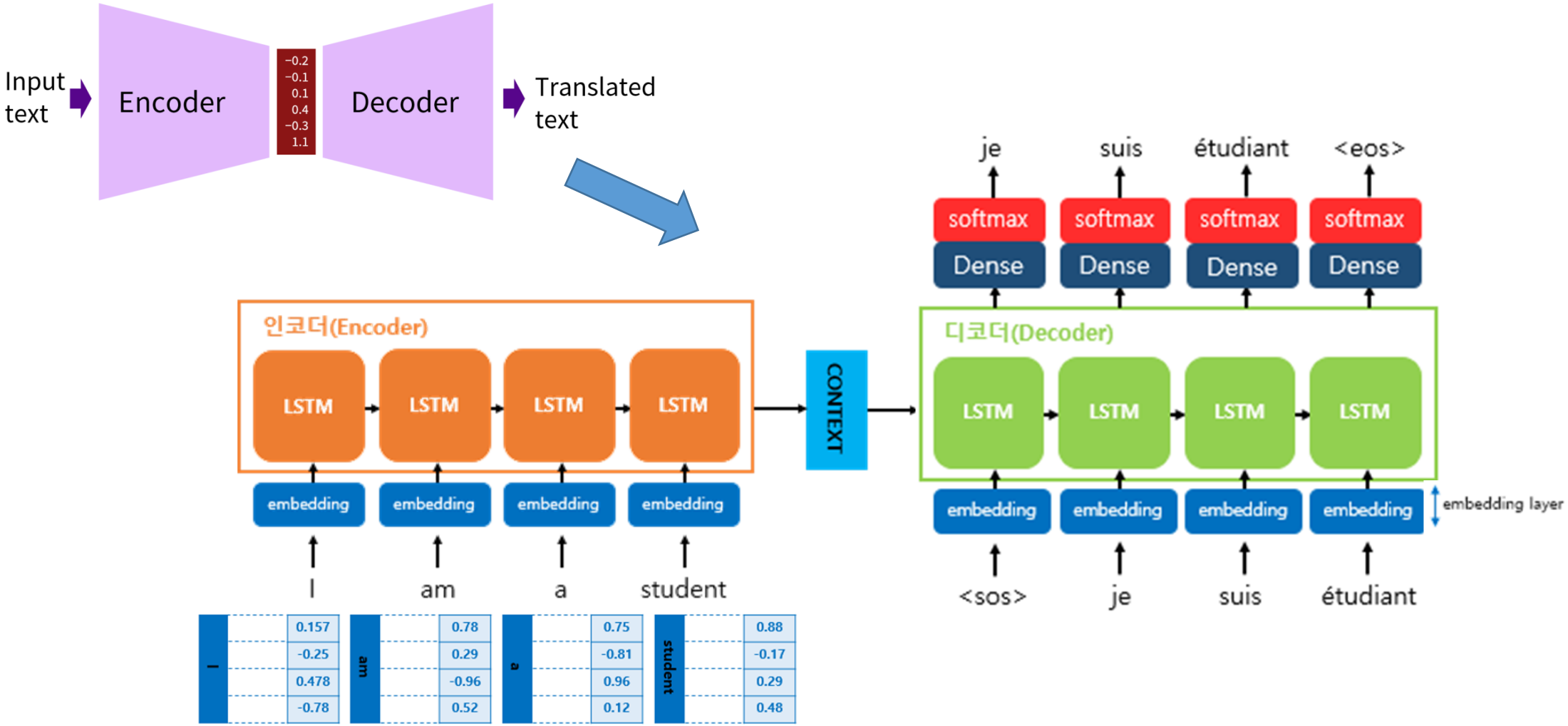
```
import tensorflow.compat.v1 as tf

tf.disable_v2_behavior()
```

출처 : https://bit.ly/2A8iZ7C

Seq2Seq(Sequence-to-Sequence)

Encoder Layer에서는 Context Vector를 만들고 Decoder Layer에서는 Context Vector 를 입력으로 출력을 예측합니다.



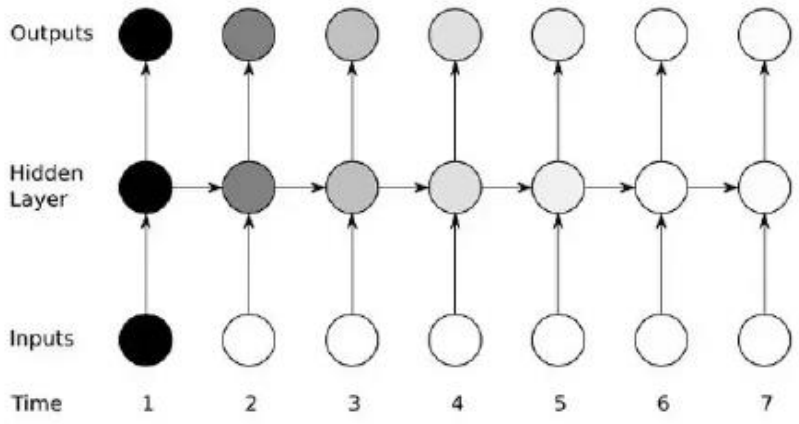
출처 : https://wikidocs.net/24996

RNN의 문제점

입력 sequence 의 길이가 매우 긴 경우 , 처음 입력의 정보가 희석되고, 고정된 context vector 사이즈로 인해 긴 sequence 에 대한 정보를 함축하기 어렵습니다.

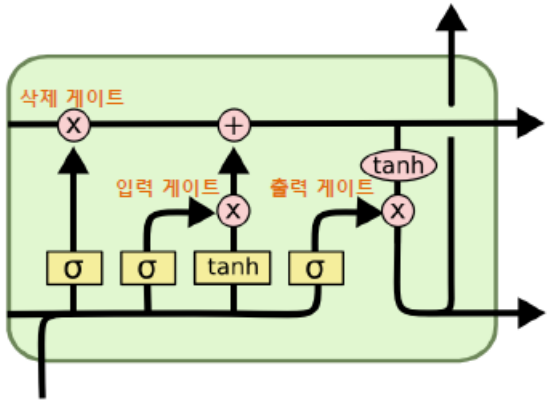
■ RNN

처음 입력의 정보가 뒤로 갈수록 사라짐



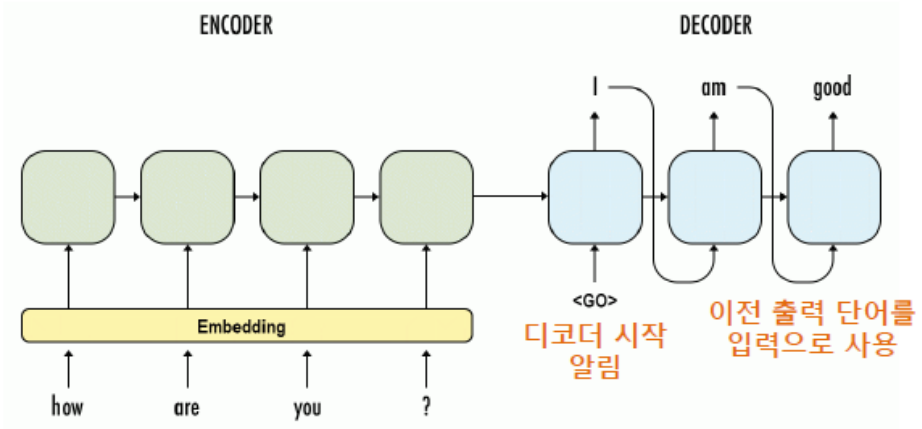
■ LSTM

정보의 흐름을 조정하는 게이트만으로는 부족



■ Seq2Seq

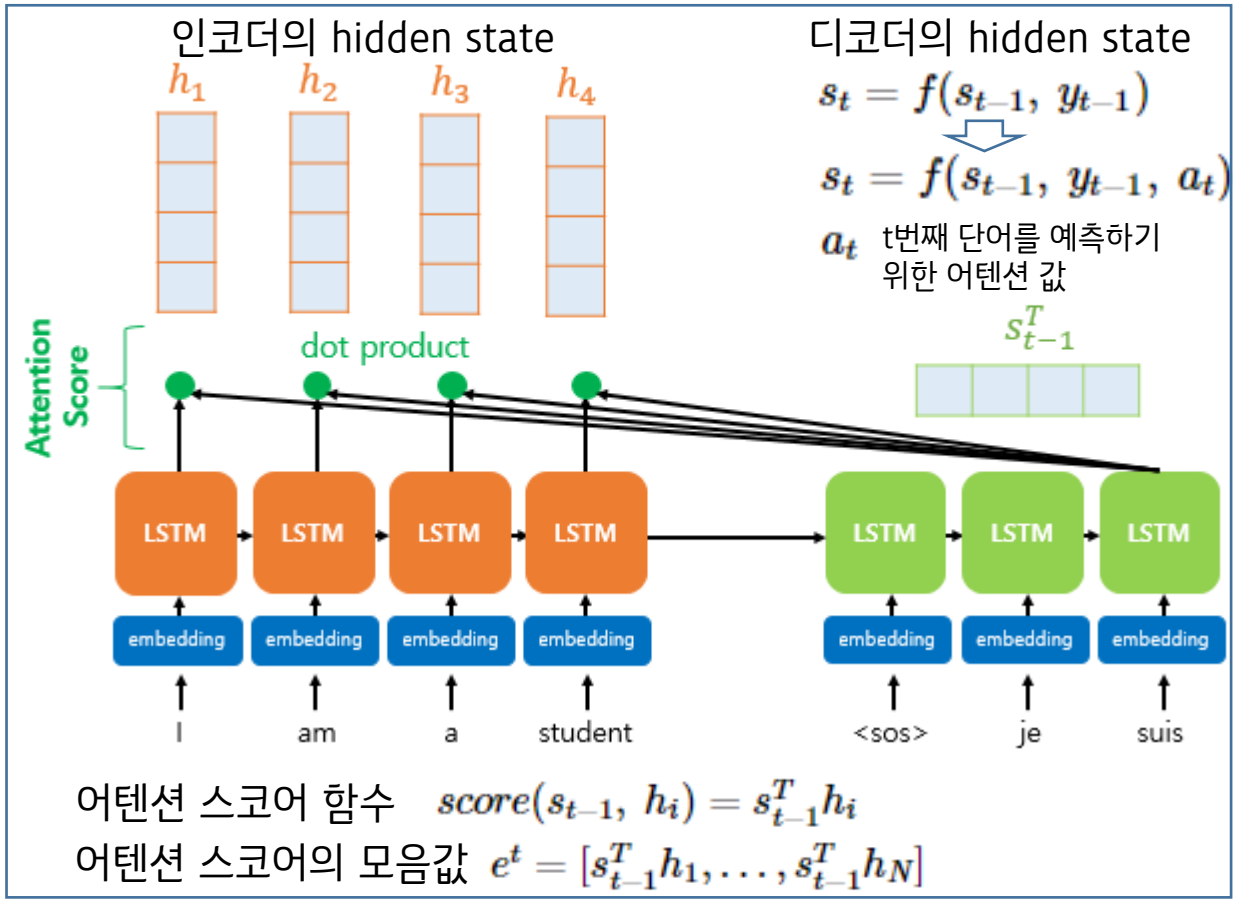
입력 문장이 길어지면 답변의 정확도 떨어짐



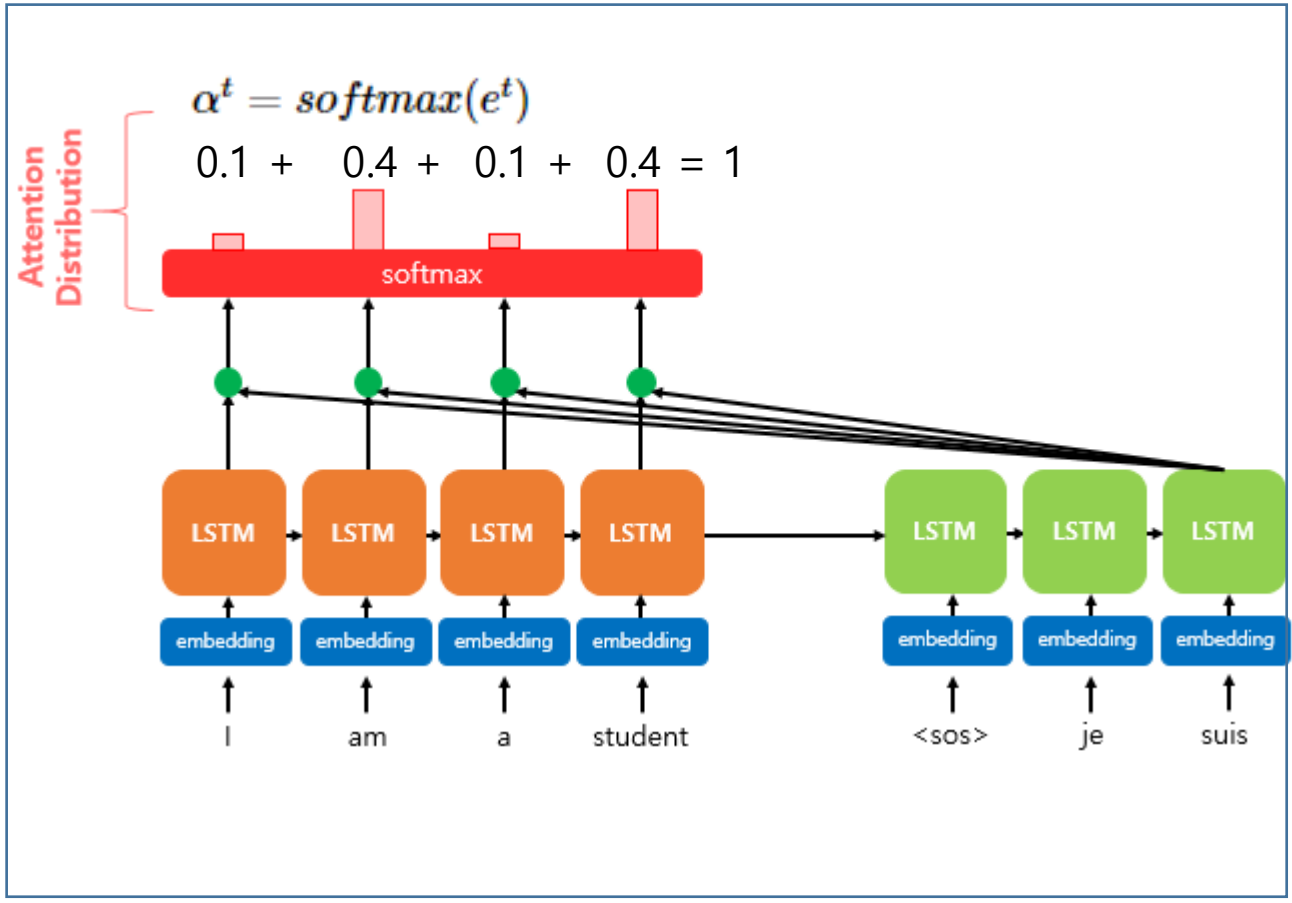
Attention 메커니즘

입력 시퀀스가 길어지면 출력 시퀀스의 정확도가 떨어지는 것을 보정하는 기법으로, 디코더에서 출력 단어를 예측하는 시점(time-step)마다, 예측해야 할 단어와 연관이 있는 입력단어 부분을 좀 더 집중(attention)해서 보는 메커니즘입니다.

1) 어텐션 스코어(Attention Score)를 구한다.



2) 소프트맥스 함수로 어텐션 분포를 구한다.

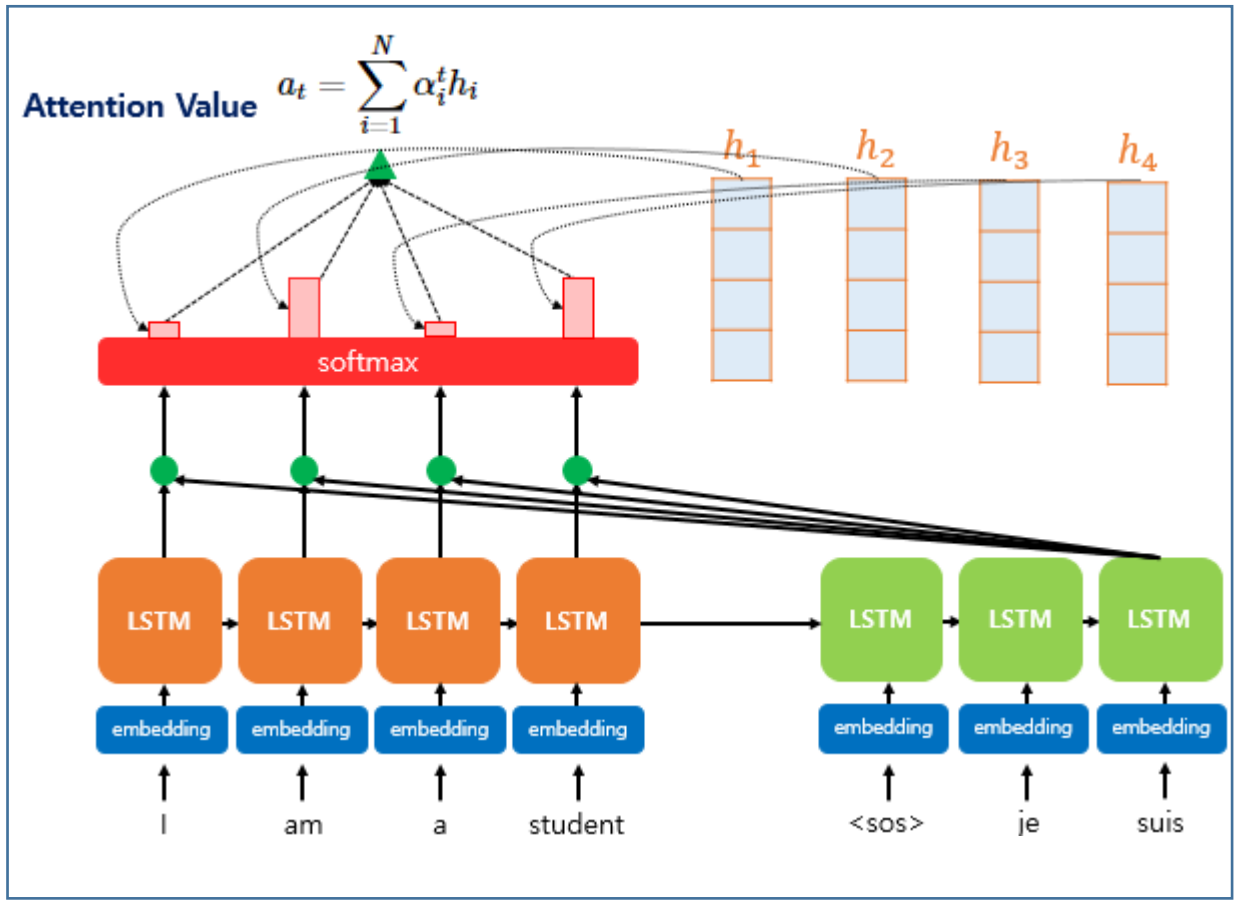


출처 : https://wikidocs.net/22893

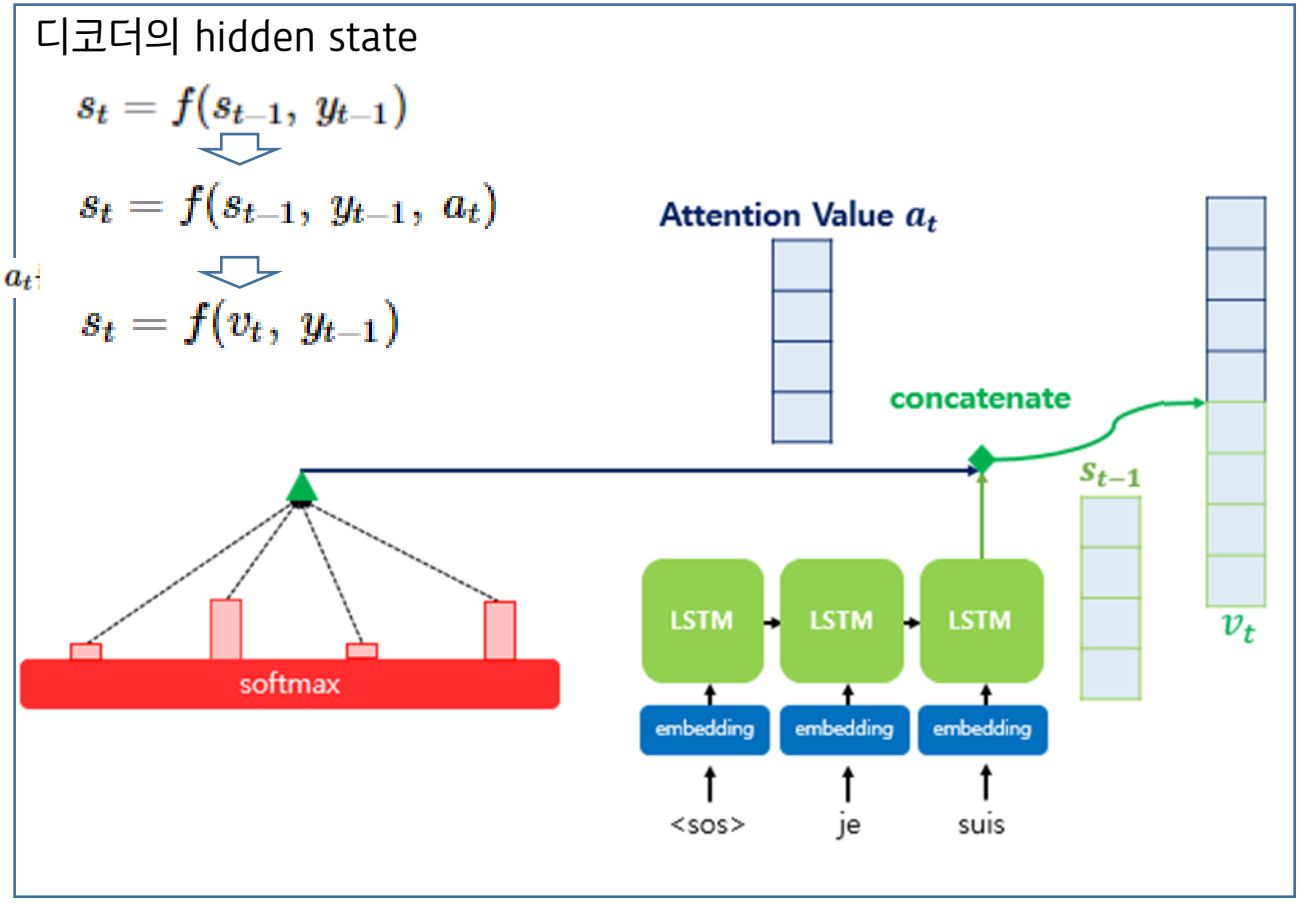
Attention 메커니즘

어텐션 값은 인코더의 문맥(context)을 포함하게 되며, 인코더의 마지막 은닉상태를 컨텍스트 벡터라고 합니다.
현재 t 시점의 은닉 상태를 계산하기 위해서 사용하던 s_{t-1}대신에 v_t를 사용하며 하여 다음 단어를 예측합니다.

3) 어텐션 값(Attention Value)을 구한다.

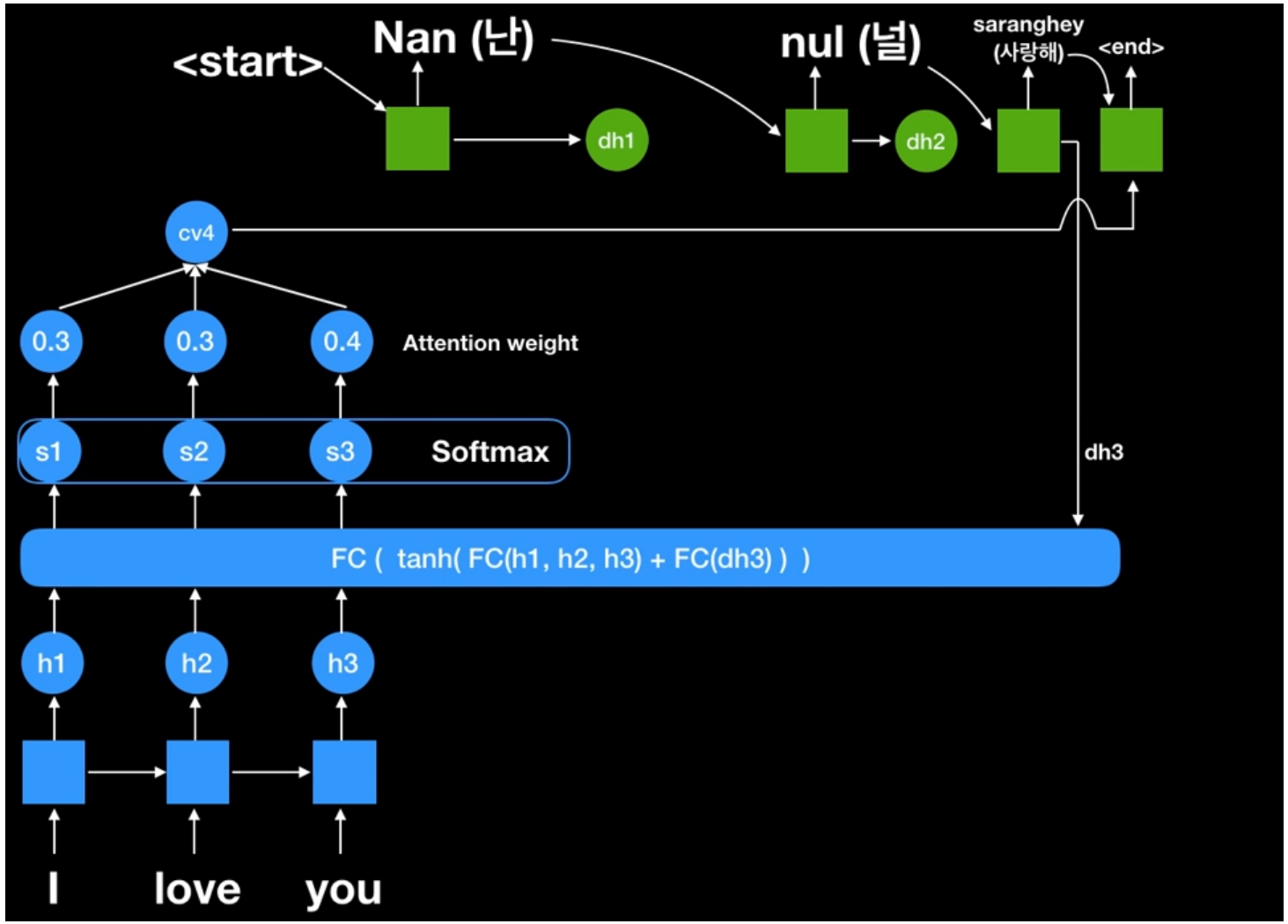


4) 어텐션 값과 디코더의 t-1 시점의 은닉 상태를 결합한다.



출처 : https://wikidocs.net/22893

Seq2Seq with Attention

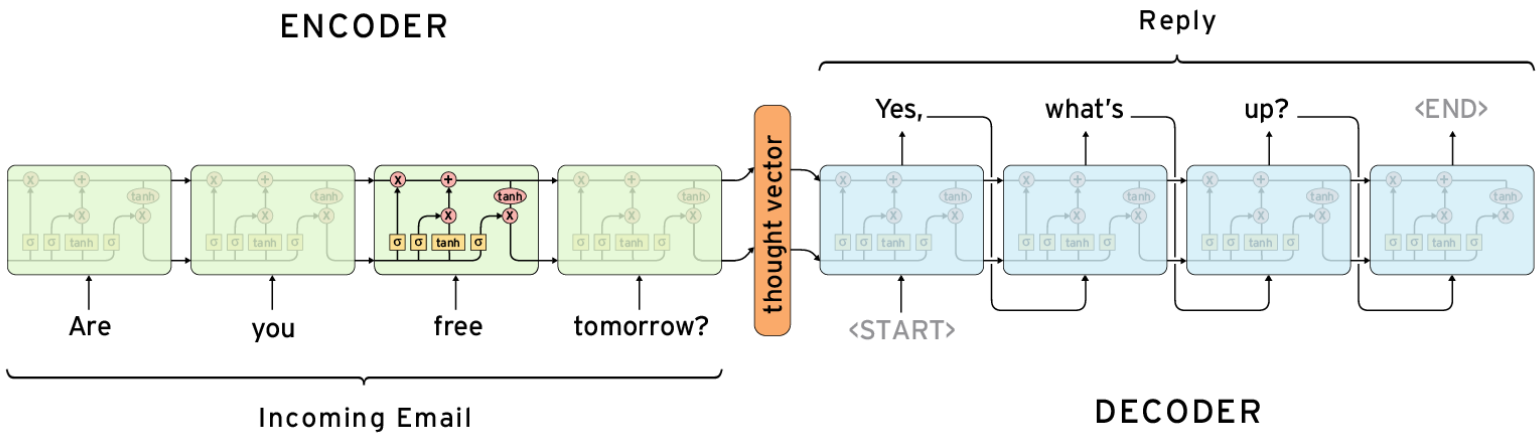


출처 : https://www.youtube.com/watch?v=WsQLdu2JMgI

Seq2Seq 기반 Chatbot 구현

■ Source Code

https://github.com/tensorlayer/seq2seq-chatbot



Training

```
python3 main.py
```

Results

```
Query > happy birthday have a nice day
 > thank you so much
 > thank babe
 > thank bro
 > thanks so much
 > thank babe i appreciate it
```

Transformer

Attention is all you need 논문으로 큰 주목을 받은 트랜스포머는 RNN을 쓰지 않고 Attention만을 사용하여 인코더-디코더 구조를 만들어 학습 속도가 무척 빠릅니다.

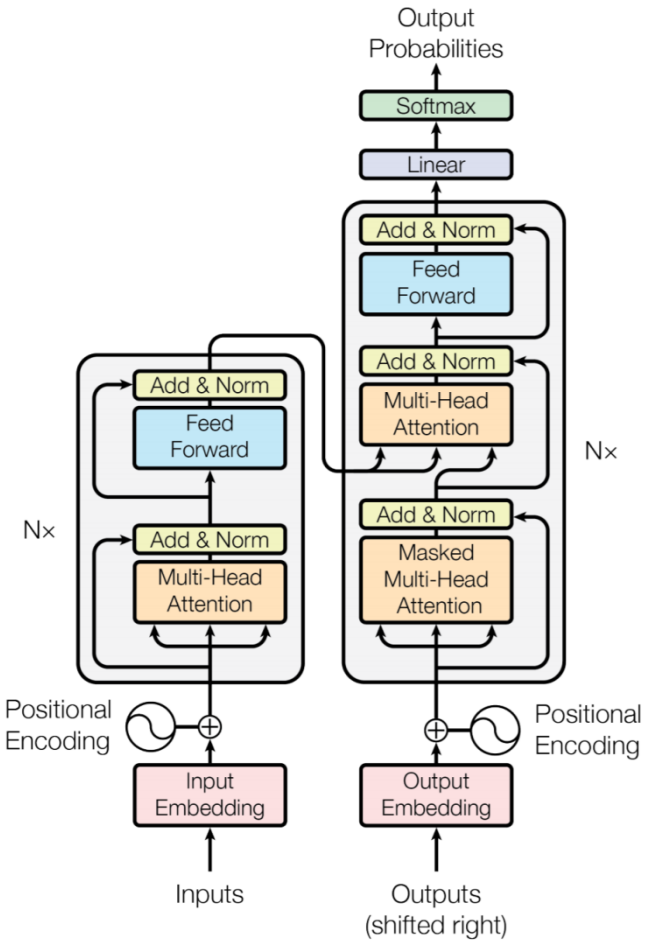
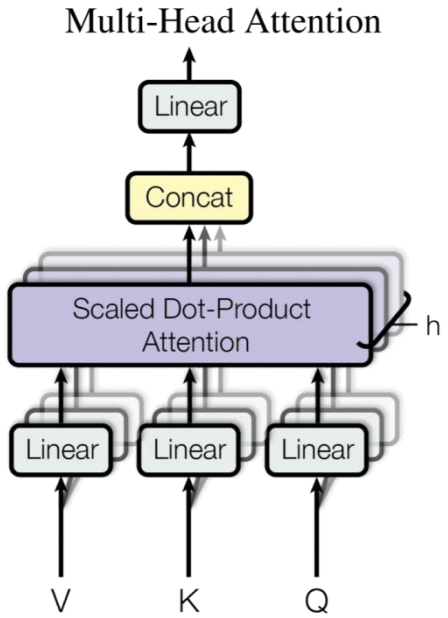
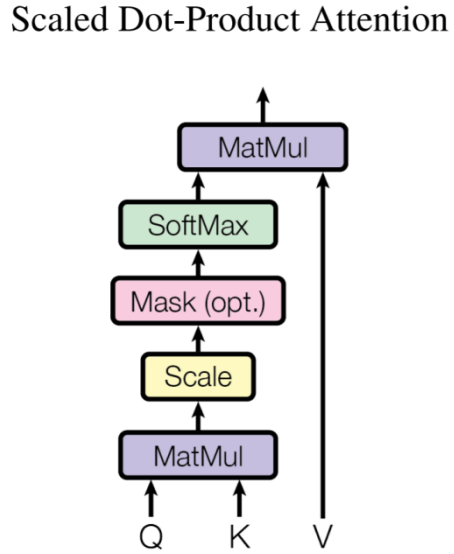


Figure 1: The Transformer - model architecture.

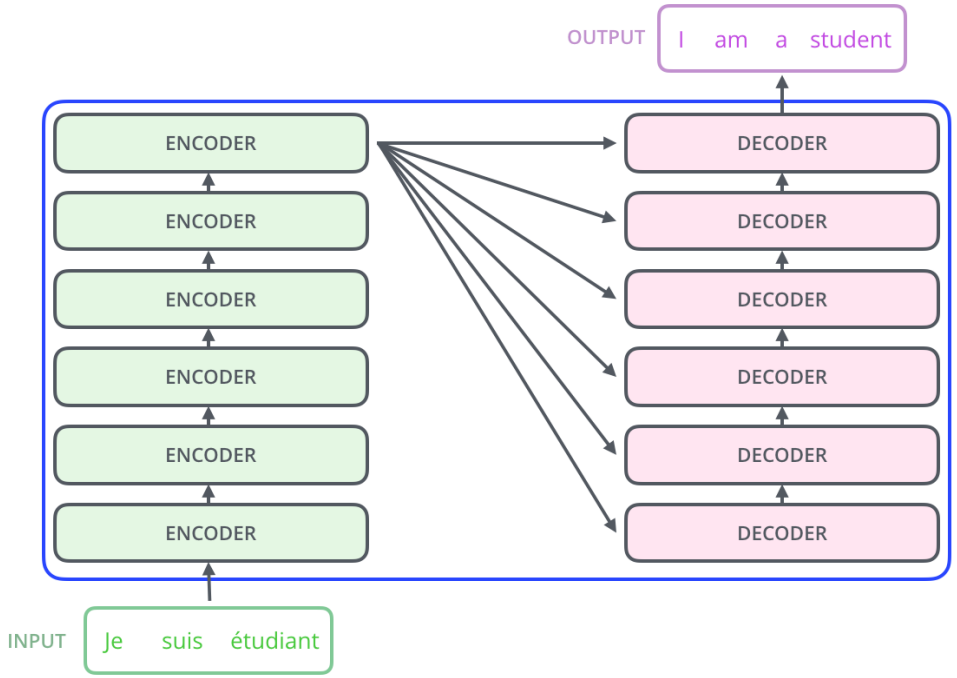




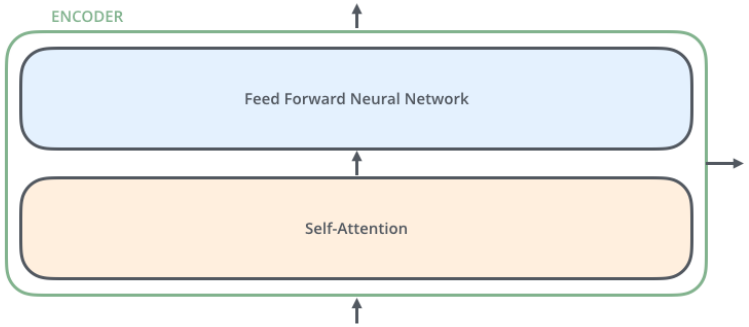
출처 : https://arxiv.org/pdf/1706.03762.pdf

Transformer

■ 기계 번역



■ Encoder



■ 행렬 형태로 표현한 self-attention 계산

$$\text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) V = Z$$

출처 : https://nlpinkorean.github.io/illustrated-transformer/

BERT(Bidirectional Encoder Representation from Transformer)

대용량 Unsupervised dataset으로 학습하는 Language Representation 모델입니다. 11개의 다양한 NLP태스크에서 SOTA(State Of TheArt) 성능을 보였으며, 일부에서는 사람보다도 뛰어난 결과가 나왔습니다.

■ BERT

- 2018년 10월 구글 발표
- 사전 훈련 기반 딥러닝 언어 모델
- 트랜스포머로 구성된 양방향 언어 표현
  (Bidirectional Encoder Representations from Transformers)
- 다양한 자연어처리 분야에 응용 가능

GLUE Test Results of BERT

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$BERT_{BASE}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$BERT_{LARGE}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

https://arxiv.org/abs/1810.04805

■ 사전 훈련 데이터(대용량 코퍼스)

https://googlebooks.byu.edu/

8억 단어

https://corpus.byu.edu/wiki/

25억 단어

(리스트, 표, 헤더 등 모두 무시하고 텍스트만 추출)

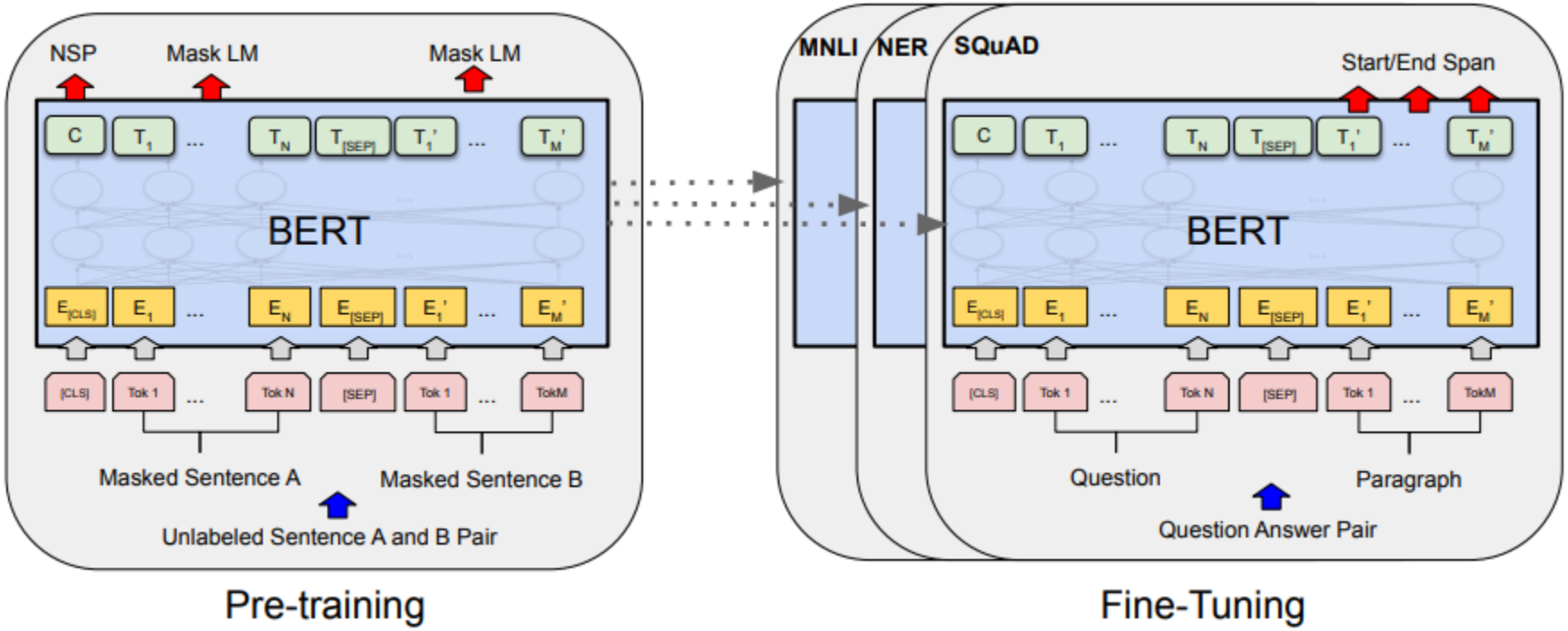
GLUE Task

Dataset	Description	Data example	Metric
CoLA	Is the sentence grammatical or ungrammatical?	"This building is than that one." = **Ungrammatical**	Matthews
SST-2	Is the movie review positive, negative, or neutral?	"The movie is funny , smart , visually inventive , and most of all , alive ." = **.93056 (Very Positive)**	Accuracy
MRPC	Is the sentence B a paraphrase of sentence A?	A) "Yesterday , Taiwan reported 35 new infections , bringing the total number of cases to 418 ." B) "The island reported another 35 probable cases yesterday , taking its total to 418 ." = **A Paraphrase**	Accuracy / F1
STS-B	How similar are sentences A and B?	A) "Elephants are walking down a trail." B) "A herd of elephants are walking along a trail." = **4.6 (Very Similar)**	Pearson / Spearman
QQP	Are the two questions similar?	A) "How can I increase the speed of my internet connection while using a VPN?" B) "How can Internet speed be increased by hacking through DNS?" = **Not Similar**	Accuracy / F1
MNLI-mm	Does sentence A entail or contradict sentence B?	A) "Tourist Information offices can be very helpful." B) "Tourist Information offices are never of any help." = **Contradiction**	Accuracy
QNLI	Does sentence B contain the answer to the question in sentence A?	A) "What is essential for the mating of the elements that create radio waves?" B) "Antennas are required by any radio receiver or transmitter to couple its electrical connection to the electromagnetic field." = **Answerable**	Accuracy
RTE	Does sentence A entail sentence B?	A) "In 2003, Yunus brought the microcredit revolution to the streets of Bangladesh to support more than 50,000 beggars, whom the Grameen Bank respectfully calls Struggling Members." B) "Yunus supported more than 50,000 Struggling Members." = **Entailed**	Accuracy
WNLI	Sentence B replaces sentence A's ambiguous pronoun with one of the nouns - is this the correct noun?	A) "Lily spoke to Donna, breaking her concentration." B) "Lily spoke to Donna, breaking Lily's concentration." = **Incorrect Referent**	Accuracy

출처 : https://huffon.github.io/2019/11/16/glue/

BERT Training

NLP의 가장 큰 문제는 training data의 부족이었으며, BERT는 언어 모델을 이해하는 데 필요한 복잡한 문제를 해결할 수 있는 새로운 토대를 제공 하였습니다.

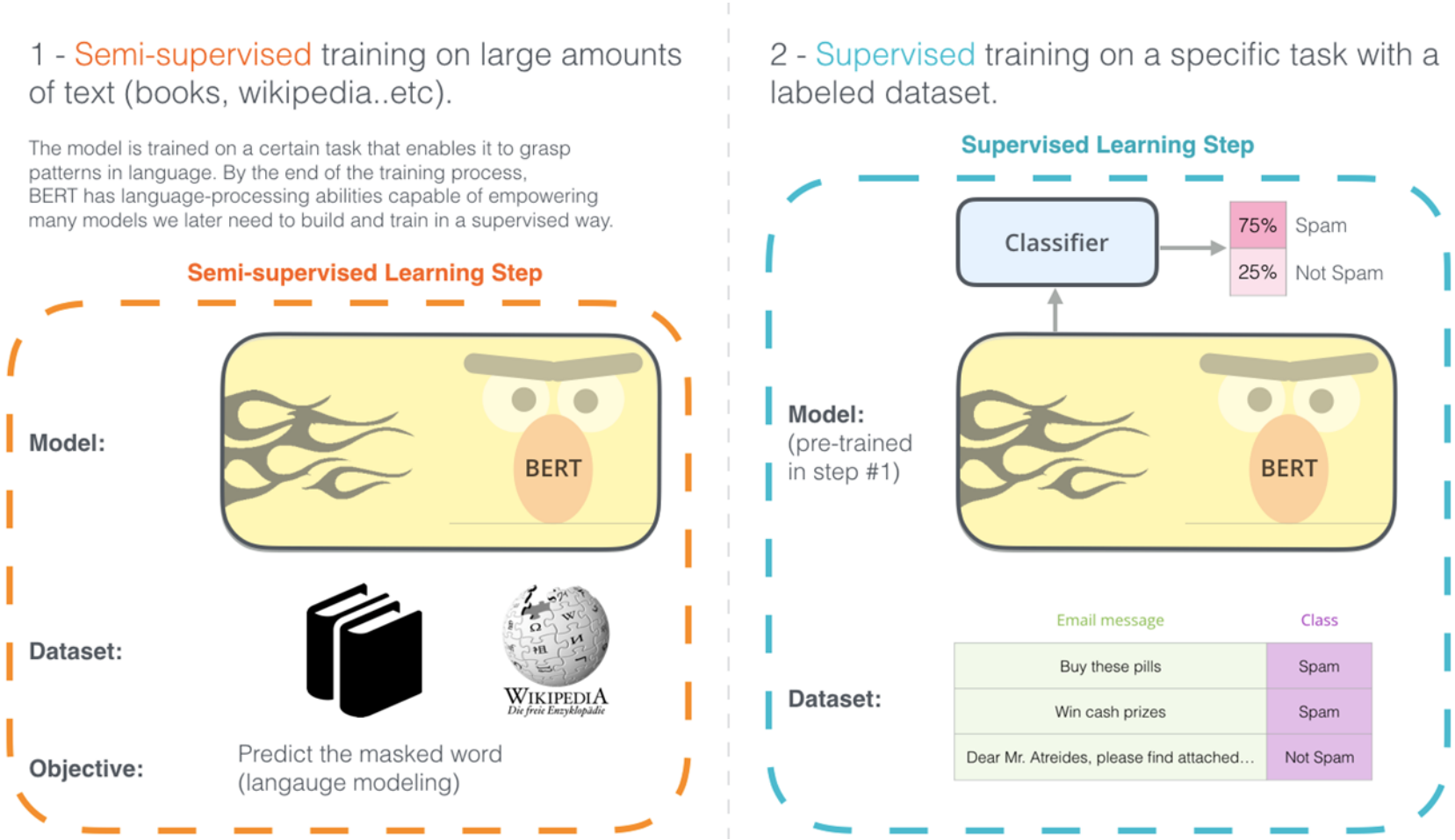


언어 전반에 대해 깊게 이해하는 단계

깊은 언어의 이해를 바탕으로 특정문제에 맞춰 적응하는 단계

BERT Training

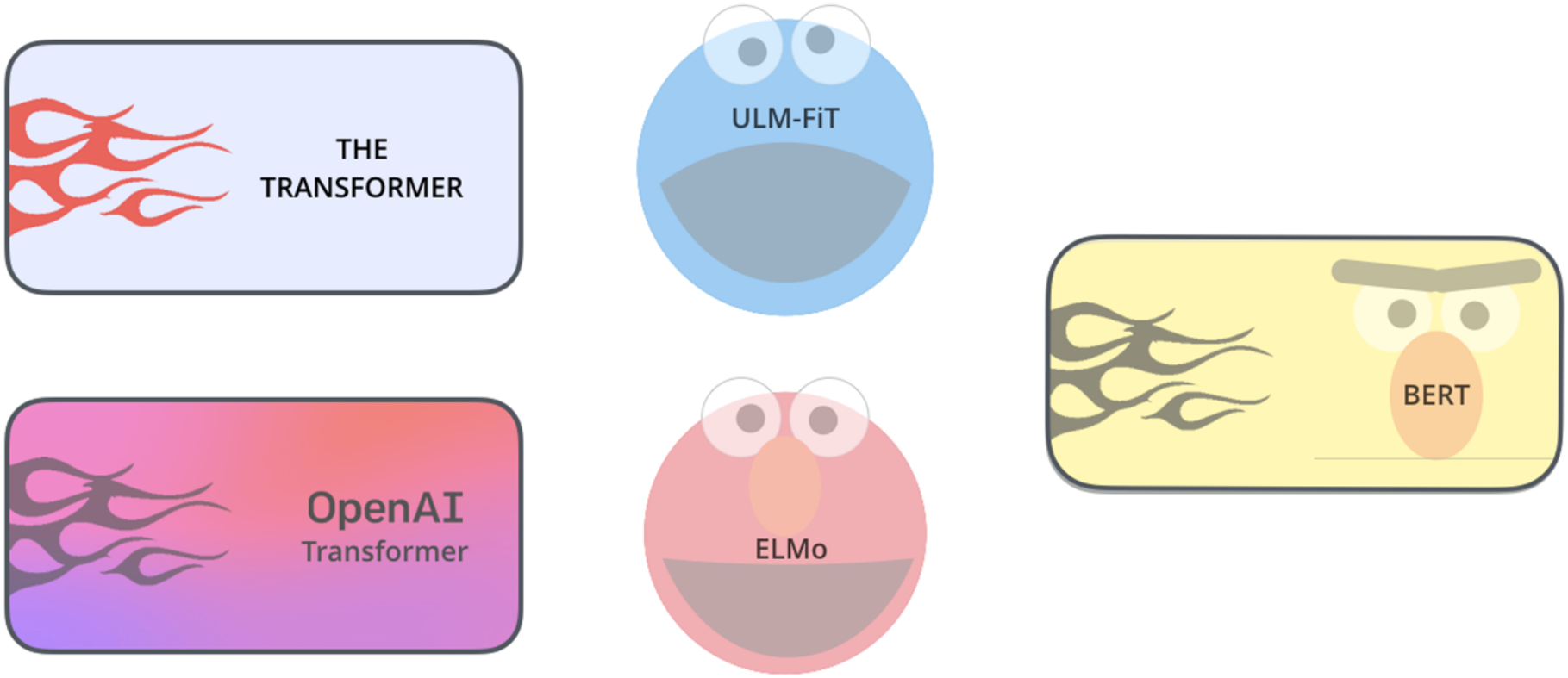
BERT가 어떻게 개발되었는 지를 보여주는 두가지 스텝입니다.
Step 1 에서는 pre-train된 모델을 다운로드 할 수 있으며, step 2 에서는 이제 fine-tuning에 대해서만 신경쓰면 됩니다.



출처 : https://jalammar.github.io/illustrated-bert/

NLP의 ImageNet 시대

NLP 커뮤니티는 다운로드만 하면 모델과 파이프라인에 쉽게 적용시킬 수 있는 매우 강력한 모델들을 내놓고 있습니다.



출처 : https://jalammar.github.io/illustrated-bert/

BERT Applications

카톡 대화 데이터를 BERT로 잘 학습시킬 수 있을까?

End-to-End BERT: 만능 모델 BERT 학습부터 추론

NLP 비전공자가 챗봇 프로젝트를 구현하기까지

기계독해를 위한 BERT 언어처리 모델 활용

딥러닝으로 동네생활 게시글 필터링하기 - 당근마켓

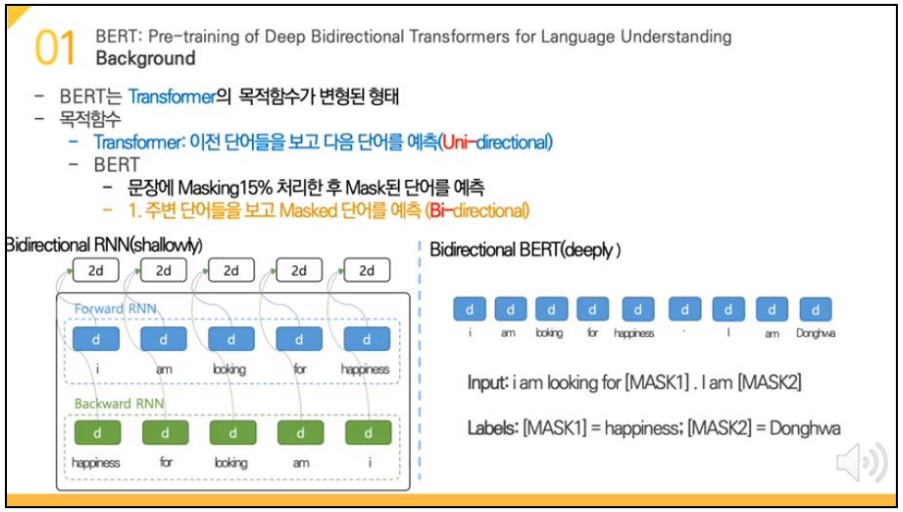
Dialog-BERT: 100억 건의 메신저 대화로 일상대화 인공지능 서비스하기

심화 학습

■ Transformer

https://www.youtube.com/watch?v=mxGCEWOxfe8

■ Transformer & BERT



https://www.youtube.com/watch?v=xhY7m8QVKjo

■ ALBERT

https://www.slideshare.net/RokJang/albert-187327421

Hands-on

Thank you