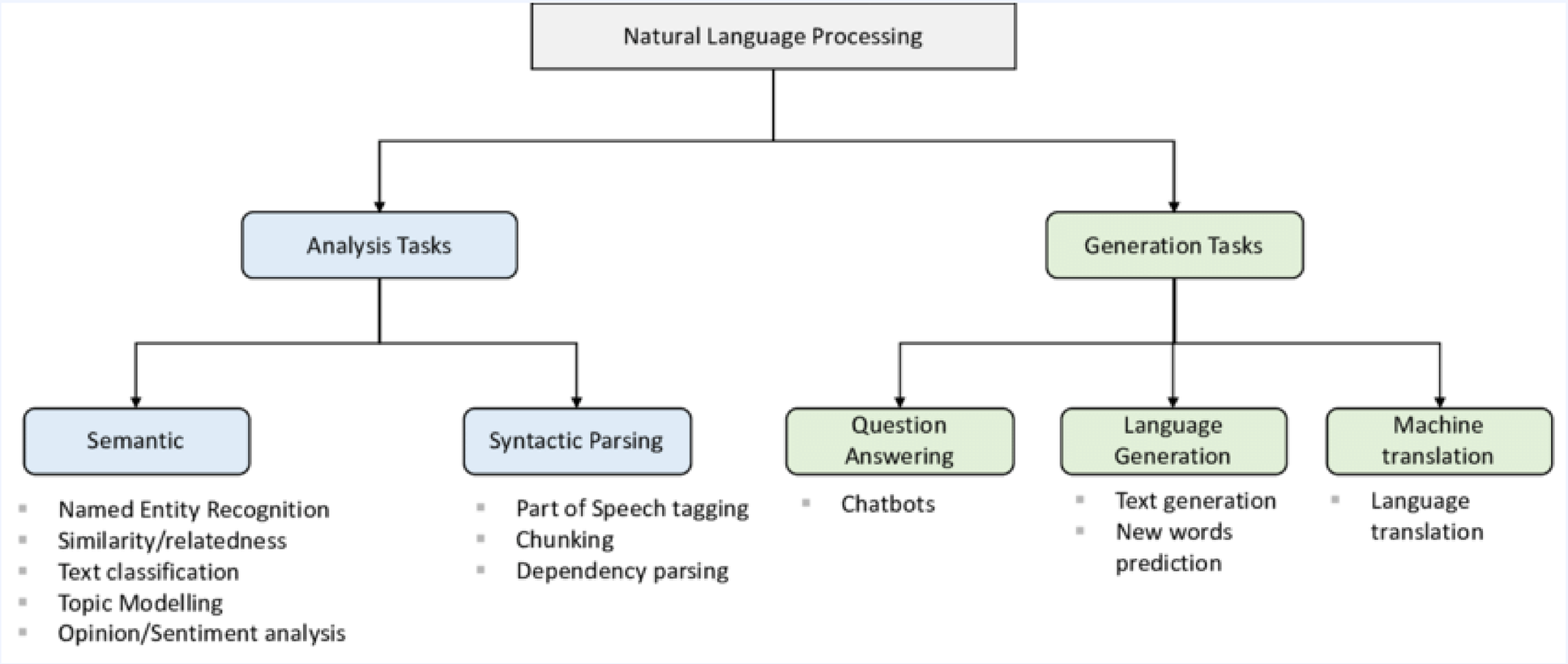
NLP

자연어 처리 분야의 주요 연구분야

NLP Tasks

- NLP Tasks Categorized in Two Broader Categories
  - Analysis Tasks (light blue) and Generation Tasks (light green)

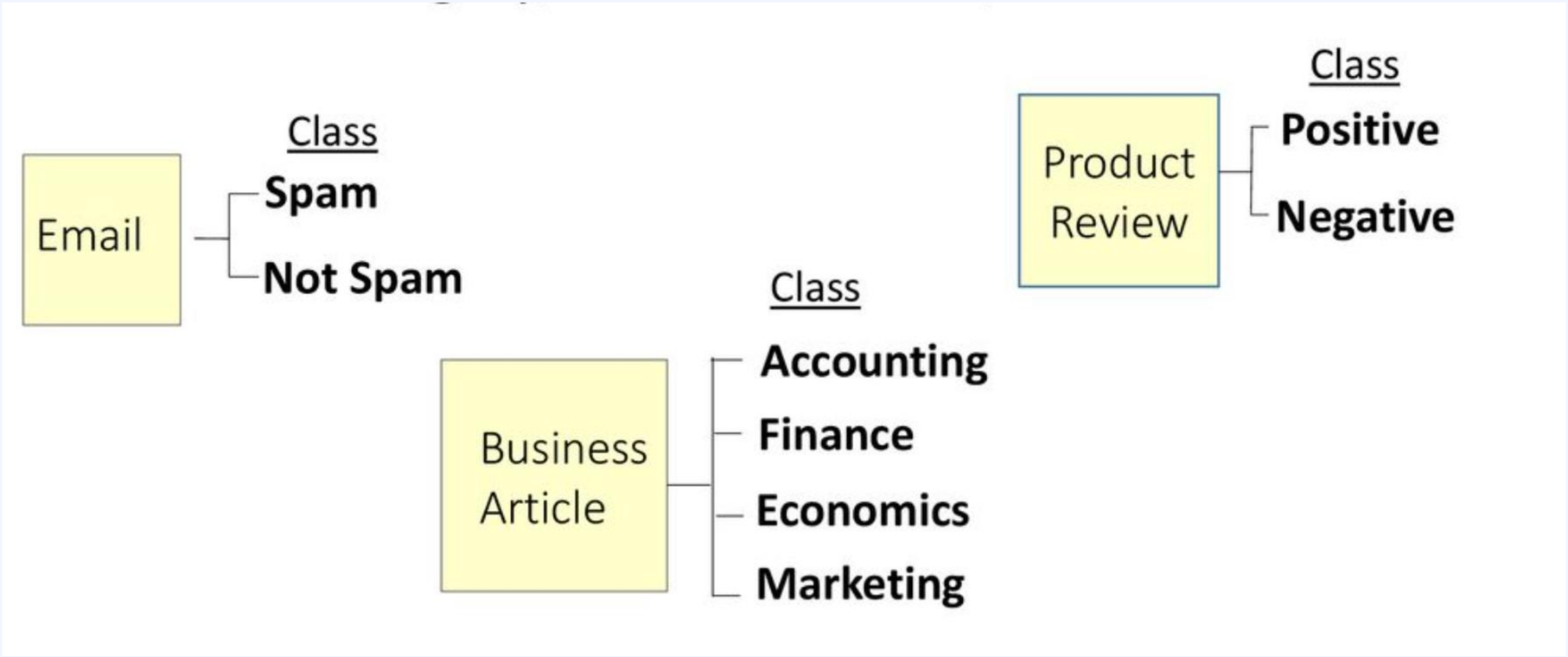


NER(Named-entity recognition)

contentSkip to site indexPoliticsSubscribeLog InSubscribeLog InToday's PaperAdvertisementSupported ORG byF.B.I. Agent Peter Strzok PERSON , Who Criticized Trump PERSON in Texts, Is FiredImagePeter Strzok, a top F.B.I. GPE counterintelligence agent who was taken off the special counsel investigation after his disparaging texts about President Trump PERSON were uncovered, was fired. CreditT.J. Kirkpatrick PERSON for The New York TimesBy Adam Goldman ORG and Michael S. SchmidtAug PERSON . 13 CARDINAL , 2018WASHINGTON CARDINAL — Peter Strzok PERSON , the F.B.I. GPE senior counterintelligence agent who disparaged President Trump PERSON in inflammatory text messages and helped oversee the Hillary Clinton PERSON email and Russia GPE investigations, has been fired for violating bureau policies, Mr. Strzok PERSON 's lawyer said Monday DATE .Mr. Trump and his allies seized on the texts — exchanged during the 2016 DATE campaign with a former F.B.I. GPE lawyer, Lisa Page — in PERSON assailing the Russia GPE investigation as an illegitimate "witch hunt." Mr. Strzok PERSON , who rose over 20 years DATE at the F.B.I. GPE to become one of its most experienced counterintelligence agents, was a key figure in the early months DATE of the inquiry.Along with writing the texts, Mr. Strzok PERSON was accused of sending a highly sensitive search warrant to his personal email account.The F.B.I. GPE had been under immense political pressure by Mr. Trump PERSON to dismiss Mr. Strzok PERSON , who was removed last summer DATE from the staff of the special counsel, Robert S. Mueller III PERSON . The president has repeatedly denounced Mr. Strzok PERSON in posts on

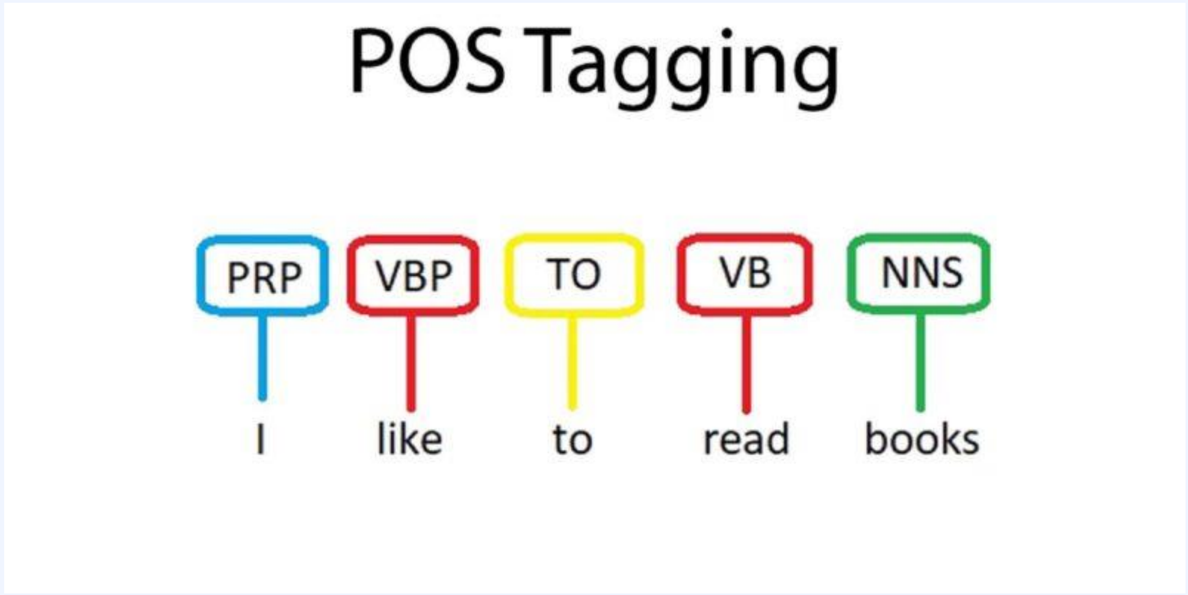
https://www.kdnuggets.com/2018/10/named-entity-recognition-classification-scikit-learn.html

Text Classification



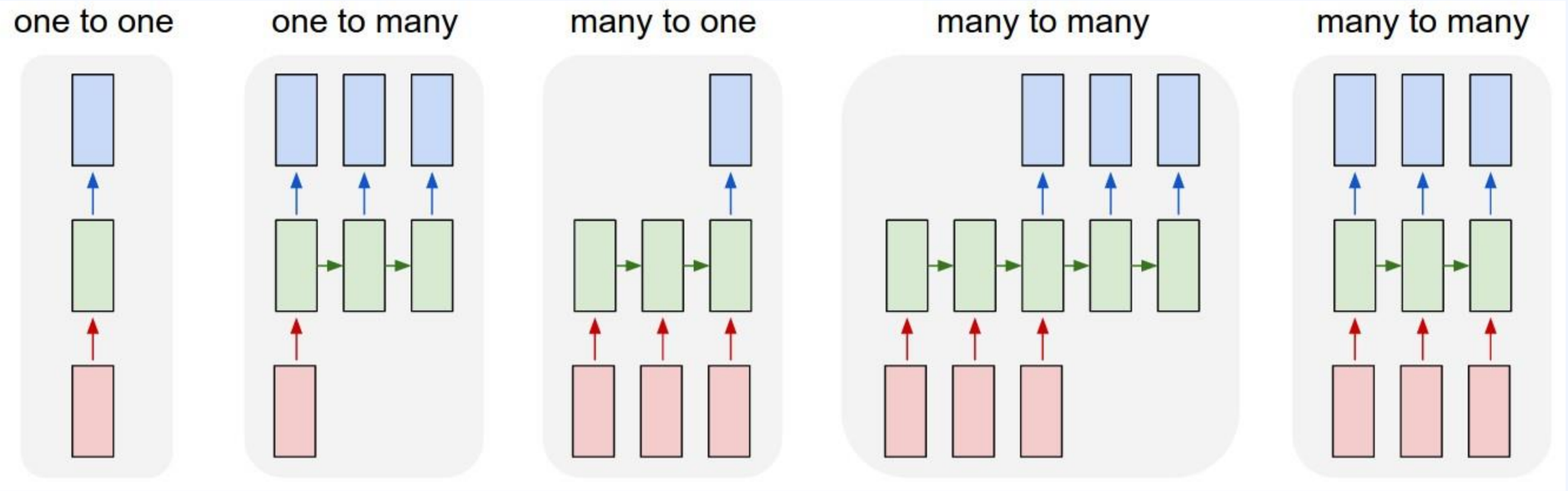
"Machine Learning: Text Classification“, Warton Univ.

Part Of Speech Tagging



https://byteiota.com/pos-tagging/

NLP Tasks



이미지 출처: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

NLP

텍스트 정제의 이해

텍스트 마이닝의 시작

➢ 모든 텍스트 마이닝의 첫 시작은 '텍스트 정제'부터!

텍스트 정제 활동의 필요성

지금봐도 질리지않음 지금 다시 방영해도 인기 최고일듯 요즘엔 이런 시트콤 왜 안해주지ㅜㅜㅜ해줬으면 좋겠다

31 답글

답글 2개

3:13 ㅋㅋㅋㅋㅋㅋㅋㅋㅋ말없이 흔드는거 개웃ㄱ결ㅋㅋㅋ

318 답글

답글 1개

신애 너무 기엽다 표정퓨ㅠㅜ

5 답글

ㄴㅏㅣㄱ ㅏㄴㅣㄱㄱㅓ ㄴㅣ ? 약속해줄께,◎원히,너만을, 사랑해♡〃 ..한사람밖에모르능은ㅈㅣ능밥오곰티_

♥...¿AΓzБぁЙ도될ㄱㅣ요¿...♥ ♡그てЙ를けrZr만보Ør도행복합ㅣてΓ♡ ♥—ー...ㅐŁøveyoц...ー—♥

ぇJ음본순간부EJュてИ를AŁzБ__54636;lØJ요, ュてЙ오r함께zr면ㅈㅣ옥Ølzr도항l복해요

ㅅΓさБ은,움ヌㅣフㅣ는ユØK_〃 ヌΓユㅣØK,ㅆΓさБぁИ'-'♥ ‖ㅣΓ능,xx뿌ㅣØK_‖ [〃♥〃] 목ㅁΓ른,ㅣИㅅΓさБ...

ㅅΓさБØ¶,날フИ를,달高 ε♡з 【ㅁΓ음°로,ぁΓ는ㅅΓさБ ε♡з】 ♧..ぁИけΓさΓフㅣフΓ__53965;〃♥〃

ユ°てと점도ØㅣぁИぁ「는것「♥」 〃ュてИㅁБ속..ㅣИスΓさㅣ는[¿] 「♧」スΓ꾸ㅁと 떨さㅣ는걸요.. .

「♥」ぁとてr는,ヱ백보てrぁь든말.. Øㅣ ㅅЛ상ØЛ서 ブΓズБØㅣ쁜 ぁЙ바己Γブㅣ +..ユ댄,LH맘,알기나,할까요ε♥з?¿..+

ㅇㅣ런것ㅇㅣ ㅅド己ぁ 인フト요..♥ ㅅド己ぁ앞ㅇJㅣ선 모두 평등ㅎトㅣ 우ズЬØ1란ø1름으로ベr≡БぁΓゴ¹。

ㅇㅣ별보㊂ ㅏ슬픈㉣rz6앞胱л슬프㉣|않길 ㅅトラㅎrス1만ㅇ1렇거l난ユ댈잊Ø¬갈ㄹЙ요

그댄 ΦΓLr요...그ㄷH뒤에ベ ㅓ 슬퍼ぁト던 작은Lト를。 실7r는㊂ㅐblr늘7r듯人rzБ엔ㅇㅣ☆◎ㅣ있ζr

유튜브 댓글, https://snskeyboard.com/darkchapter/

텍스트 정제 활동

- 대소문자 통일
- 문장부호/특수문자 제거
- 숫자제거
- 공백, 개행문자 제거
- 띄어쓰기 보정
- 품사분석(Part of Speech tagging)
- 불용어(stopword) 제거
- 토큰화
- …

Normalize text

Input	Output
The FOX was brown, with keen eyes.	the fox was brown with keen eyes
The ribbons were red, blue, and green.	the ribbons were red blue and green
It was (almost) completely empty.	it was almost completely empty
Size:LARGE color:RED ---low stock!!!	size large color red low stock
RED-BLUE-GREEN	red blue green

https://medium.com/analytics-vidhya/a-simple-yet-effective-way-of-text-cleaning-using-nltk-4f90a8ff21d4

텍스트 정제

- 토큰화
  - 자연어를 어떤 단위로 살펴볼 것인가
    - 어절: 문장 성분의 최소 단위로서 띄어쓰기의 단위
    - 형태소: 의미를 가지는 요소로서는 더 이상 분석할 수 없는 가장 작은 말의 단위
    - n-gram

His mother went to school with him

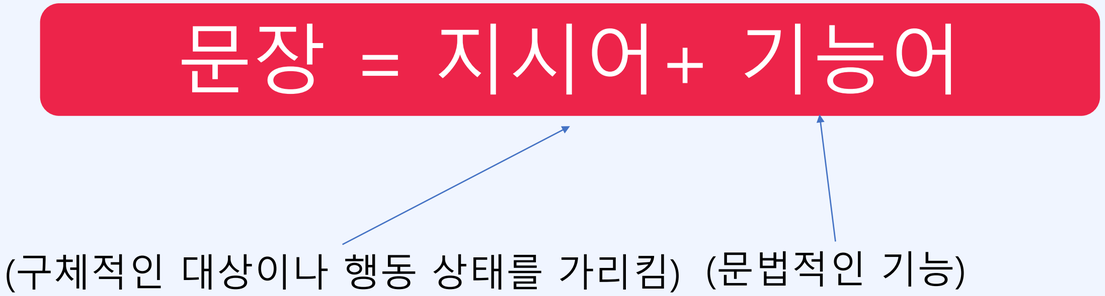
His / mother / went / to / school / with / him

그의 어머니는 그와 함께 학교에 갔다

그/의/ 어머니/는/ 그/와/ 함께/ 학교/에 가/았/다

텍스트 정제

➢ 불용어
- 기능어는 문장에서 실질적인 의미를 별로 가지고 있지 않기 때문에 불용어로 간주하여 제거
  - 영어: 관사, 전치사
  - 한국어: 조사



텍스트 정제

➢ 어근 동일화

- 어간 추출(stemming)
  - 단어의 어미나 접두사, 접미사 등으로 해서 형태가 달라진 단어들을 형태소 분석을 통해 그 어간을 추출하여 동일한 단어로 간주하는 작업
- 표제어 추출(lemmatization)
  - 어간 추출의 단점 보완: 단어의 의미적 단위를 고려하지 않고 단어를 축약형으로 정리

	어간 추출(stemming)	표제어 추출(lemmatization)
love, loves, loving, loved	lov	love
innovation, innovations, innovate, innovates, innovative	innovat	• **innovation**(innovation, innovations) • **innovate**(innovate, innovates) • **innovative**(innovative)

텍스트 정제

➢ 품사분석(Part of Speech tagging)

- 브라운 말뭉치(Brown corpus) 기반
  - 미국의 브라운 대학에서 구축, 87개의 표식 목록

His / mother / went / to / school / with / him

His(PPR$ 소유대명사) mother(NN 단수명사) went(VRD 동사과거형) to(TO, to)
school(NN 단수명사) with(IN 전치사) him(PRP, 인칭대명사)

- 국립국어원의 세종 말뭉치 사전에 기반

그/의/ 어머니/는/ 그/와/함께/ 학교/에/가/있/다

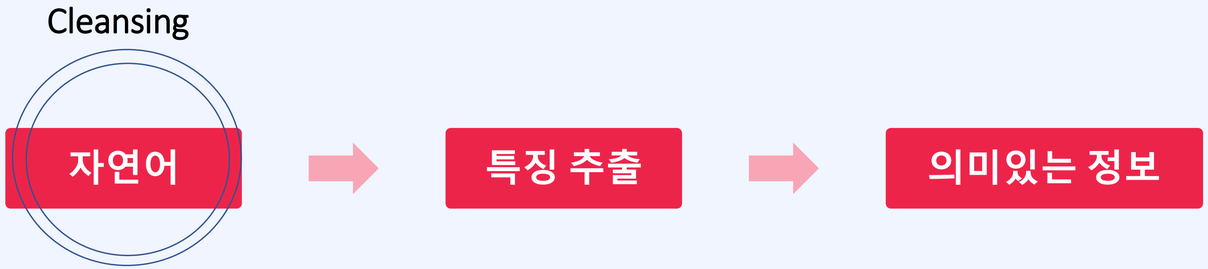
그(NP 대명사)의(JKG 관형격 조사)어머니(NNG 일반명사)는(JKS 주격조사)
그(NP 대명사)와(JKB 부사격조사)함께(MAG 일반부사)학교(NNG 일반명사)
에(JKB 부사격조사)가(VW동사)있(EC 연결어미)다(EF 종결어미)

Summary

➢ Text Cleansing

- 필요성
  - 할많하않^^!

- 활동
  - 대소문자 통일
  - 문장부호/특수문자 제거
  - 숫자제거
  - 공백, 개행문자 제거
  - 띄어쓰기 보정
  - 품사분석(Part of Speech tagging)
  - 불용어(stopword) 제거
  - 토큰화
  - …



NLP

자연어 처리 단계

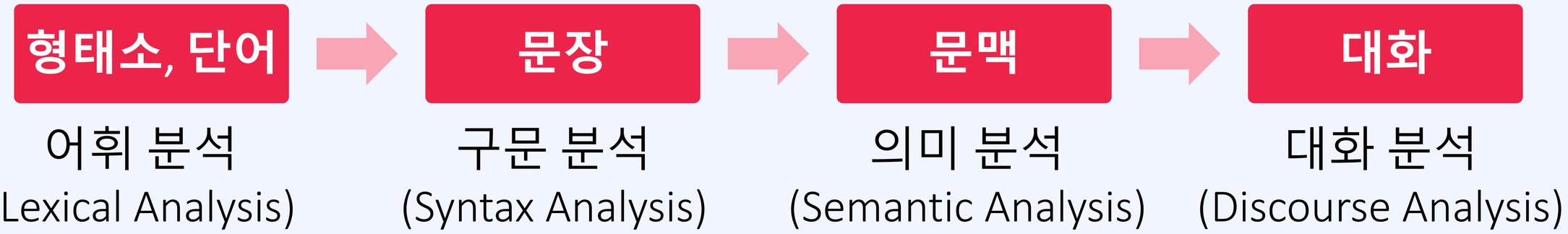
텍스트 마이닝(Text Mining)

- 자연어 처리 기술을 활용하여 텍스트 데이터를 정형화하고, 특징을 추출하여 의미 있는 정보를 발견할 수 있도록 하는 기술

자연어 → **특징 추출** → **의미있는 정보**

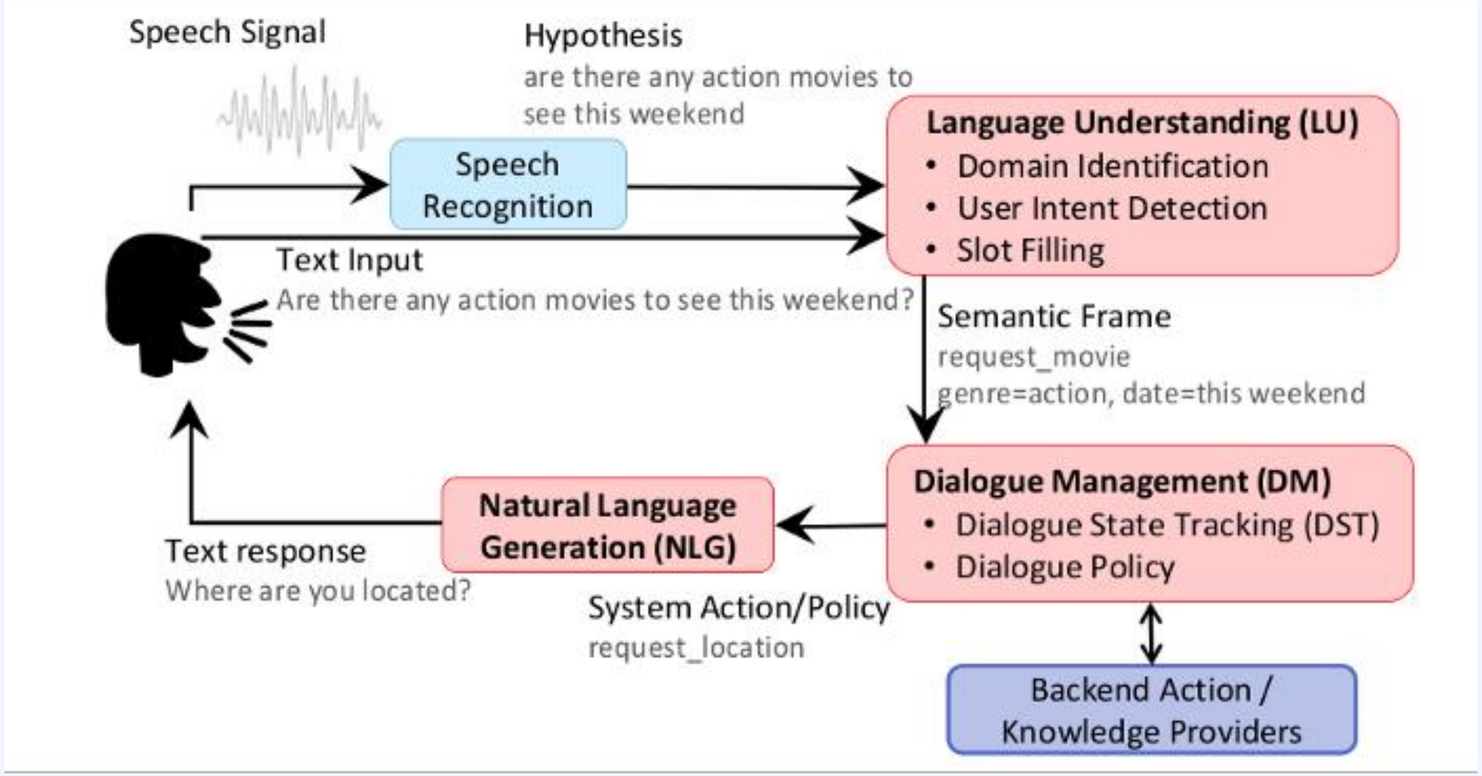
이미지 출처: https://www.youtube.com/watch?v=FHI4gX5EbYI

자연어 처리의 단계



자연어 처리의 단계

➢ Dialogue System



https://medium.com/@nisar.shah1/introduction-to-dialogue-systems-part-1-475a06ab78ad

Quiz

Q **자연어 처리 시스템의 구성요소 중 자연어 데이터를 인텐트(intent)와 슬롯(slot) 형태의 정형데이터로 추출하는 기능을 수행하는 모듈은?**

① NLU(Natural Language Understanding)

② DM(Dialog Manager)

③ NLG(Natural Language Generator)

④ KB(Knowledge Base)

해설

▶ NLU는 자연어를 이해하는 모듈로 입력된 텍스트의 의도와 주요 정보를 인텐트와 슬롯 형태로 각각 추출하는 기능을 수행한다.

Quiz

Q **자연어 처리 시스템의 구성요소 중 대화의 컨텍스트(Context)를 기반으로 가장 적합한 응답을 선택하는 기능을 수행하는 모듈은? ?**

① NLU(Naturla Language Understanding)

② DM(Dialog Manager)

③ NLG(Natural Language Generator)

④ KB(Knowledge Base)

해설

▶ DM은 대화를 관리하는 모듈로 입력된 문장에 알맞은 응답을 생성하기 위해 지식 베이스(Knowledge base)를 조회하여 적절한 응답을 선택하는 기능을 수행한다.

NLP

자연어 전처리에 사용하는 파이썬

자연어 전처리에 사용하는 파이썬

➢ 파이썬 내장 함수

▪ 문자열 함수

split	문자열 분리	str.split([sep])
strip	문자열 삭제	str.strip([chars])
join	문자열 연결	str.join(seq)
find	문자열 찾기	str.find(search_str, start, end)
count	문자열 일치 횟수 반환	str.count(search_str) str.count(search_str, start) str.count(search_str, start, end)
startswith	문자열이 특정 문자열로 시작하는지 검사	str.startswith(prefix) str.startswith(prefix, start) str.startswith(prefix, start, end)
endswith	문자열이 특정 문자열로 끝나는지 검사	str.endswith(suffix) str.endswith(suffix, start) str.endswith(suffix, start, end)
replace	문자열 바꾸기	str.replace(old, new[, count])
lower	소문자로 변경	str.lower()
upper	대문자로 변경	str.upper()

자연어 전처리에 사용하는 파이썬

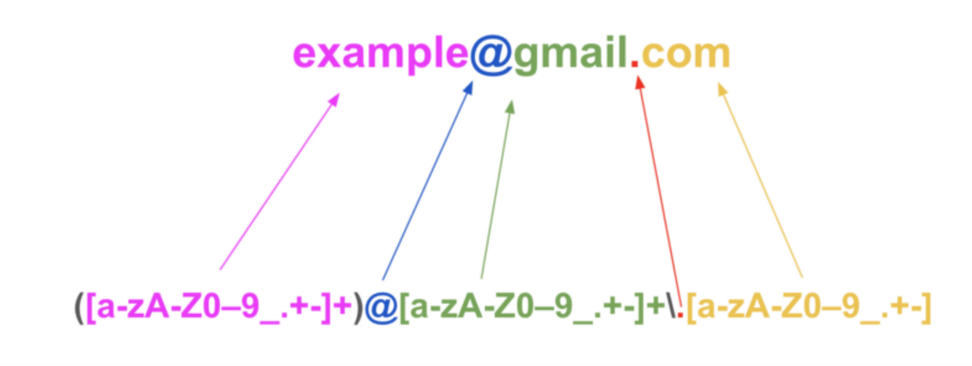
➢ 기본 파이썬

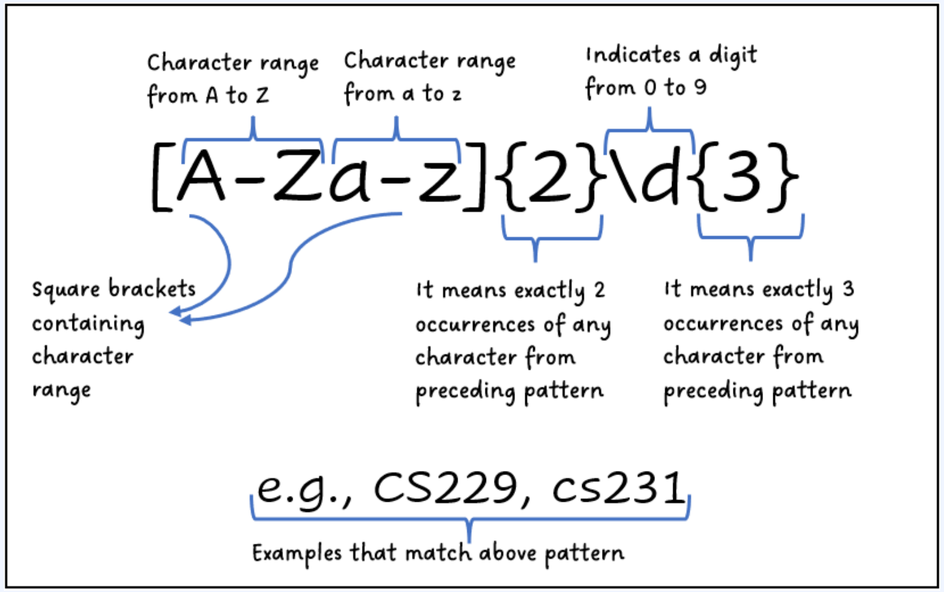
▪ 문자열 함수

isalpha()	문자열이 숫자, 특수 문자, 공백이 아닌 문자로 구성되어 있을 때만 True, 그 밖에는 False 반환	str.isalpha()
isdigit()	문자열이 모두 숫자로 구성되어 있을 때만 True, 그 밖에는 False 반환	str.isdigit()
isalnum()	문자열이 특수 문자나 공백이 아닌 문자와 숫자로 구성되어 있을 때만 True, 그 밖에는 False 반환	str.isalnum()
isspace()	문자열이 모두 공백 문자로 구성되어 있을 때만 True, 그 밖에는 False 반환	str.isspace()
isupper()	문자열이 모두 로마자 대문자로 구성되어 있을 때만 True, 그 밖에는 False 반환	str.isupper()
islower()	문자열이 모두 로마자 소문자로 구성되어 있을 때만 True, 그 밖에는 False 반환	str.islower()

자연어 전처리에 사용하는 파이썬

➢ re
- 정규 표현식(Regular Expression)을 사용해서 입력 문장의 패턴을 찾을 수 있음





https://towardsdatascience.com/easiest-way-to-remember-regular-expressions-regex-178ba518bebd
https://towardsai.net/p/l/regular-expression-regex-in-python-the-basics

자연어 전처리에 사용하는 파이썬

➢ NLTK(Natural Language Toolkit)

- 자연어 처리 및 문서분석용 파이썬 패키지(https://www.nltk.org/index.html)
- 분류, 토큰화, 형태소 분석, 태깅, 구문 분석, 의미 추론을 위한 자연어 처리 라이브러리 제공
- WordNet 등 50개 이상의 말뭉치 및 어휘 리소스 제공
  - nltk.corpus
    - https://www.nltk.org/api/nltk.corpus.html
    - http://www.nltk.org/nltk_data/

```
import nltk
from nltk.corpus import brown as cb
from nltk.corpus import gutenberg as cg
```

```
print cb.categories()
```

```
[u'adventure', u'belles_lettres', u'editorial', u'fiction', u'government', u'hobbie
s', u'humor', u'learned', u'lore', u'mystery', u'news', u'religion', u'reviews', u'ro
mance', u'science_fiction']
```

자연어 전처리에 사용하는 파이썬

- KoNLPy
  - 한국어 정보처리를 위한 파이썬 패키지
  - https://konlpy.org/ko/latest/index.html
  - 5가지 형태소 분석기를 제공
    - Hannanum, Kkma, Komoran, Mecab, Okt

```
>>> 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)]
```

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

자연어 전처리에 사용하는 파이썬

- soynlp
  - https://github.com/lovit/soynlp
  - 한국어 분석을 위한 pure python code
    - Noun Extractor, NewsNounExtractor
    - Word Extraction
    - Tokenizer: Ltokenizer, MaxScoreTokenizer, RegexTokenizer
    - Part of Speech Tagger
    - Vectorizer
    - Normalizer
    - Point-wise Mutual Information (PMI)

```
from soynlp.noun import NewsNounExtractor
noun_extractor = NewsNounExtractor()
nouns = noun_extractor.train_extract(sentences)
```

```
덴마크  옷돈  너무너무너무  가락동  매뉴얼  지도교수
전망치  강구  언니들  신산업  기획전  노스
할리우드  플라자  불법조업  월스트리트저널  2022년  불허
고씨  어플  1987년  불씨  적기  레스
스퀘어  충당금  건축물  뉴질랜드  사각  하나씩
근대  투자주체별  4위  태권  네트웍스  모바일게임
연동  런칭  만성  손질  제작법  현실화
오해영  심사위원들  단점  부장조리  차관급  게시물
인터폰  원화  단기간  편곡  무산  외국인들
세무조사  석유화학  워킹  원피스  서장  공범
```

https://github.com/lovit/soynlp

자연어 전처리에 사용하는 파이썬

➢ Gensim

- 토픽 모델링, 문서 인덱싱, 유사성 검색 및 기타 자연어 처리를 위한 오픈 소스 라이브러리

- `corpora.bleicorpus` – Corpus in Blei's LDA-C format
- `corpora.csvcorpus` – Corpus in CSV format
- `corpora.dictionary` – Construct word<->id mappings
- `corpora.hashdictionary` – Construct word<->id mappings
- `corpora.indexedcorpus` – Random access to corpus documents
- `corpora.lowcorpus` – Corpus in GibbsLda++ format
- `corpora.malletcorpus` – Corpus in Mallet
- `corpora.mmcorpus` – Corpus in Matrix Ma
- `corpora.opinosiscorpus` – Topic related
- `corpora.sharded_corpus` – Corpus storec
- `corpora.svmlightcorpus` – Corpus in SVM
- `corpora.textcorpus` – Tools for building
- `corpora.ucicorpus` – Corpus in UCI form
- `corpora.wikicorpus` – Corpus from a Wi

- `models.nmf` – Non-Negative Matrix factorization
- `models.lsimodel` – Latent Semantic Indexing
- `models.ldaseqmodel` – Dynamic Topic Modeling in Python
- `models.tfidfmodel` – TF-IDF model
- `models.rpmodel` – Random Projections
- `models.hdpmodel` – Hierarchical Dirichlet
- `models.logentropy_model` – LogEntropy m
- `models.normmodel` – Normalization mode
- `models.translation_matrix` – Translation
- `models.fasttext` – FastText model
- `models.word2vec` – Word2vec embeddin

- `topic_coherence.aggregation` – Aggregation module
- `topic_coherence.direct_confirmation_measure` – Direct confirmation measure module
- `topic_coherence.indirect_confirmation_measure` – Indirect confirmation measure module
- `topic_coherence.probability_estimation` – Probability estimation module
- `topic_coherence.segmentation` – Segmentation module
- `topic_coherence.text_analysis` – Analyzing the texts of a corpus to accumulate statistical information about word occurrences
- `scripts.package_info` – Information about gensim package
- `scripts.glove2word2vec` – Convert glove format to word2vec

자연어 전처리에 사용하는 파이썬

- sklearn
  - 머신러닝 및 데이터 전처리를 위한 기능 제공

The `sklearn.feature_extraction.text` submodule gathers utilities to build feature vectors from text documents.

`feature_extraction.text.CountVectorizer`(*[, ...])	Convert a collection of text documents to a matrix of token counts.
`feature_extraction.text.HashingVectorizer`(*)	Convert a collection of text documents to a matrix of token occurrences.
`feature_extraction.text.TfidfTransformer`(*)	Transform a count matrix to a normalized tf or tf-idf representation.
`feature_extraction.text.TfidfVectorizer`(*[, ...])	Convert a collection of raw documents to a matrix of TF-IDF features.

자연어 전처리에 사용하는 파이썬

➢ torchtext

- torchtext
- torchtext.nn
  - MultiheadAttentionContainer
  - InProjContainer
  - ScaledDotProduct
- torchtext.data.functional
  - generate_sp_model
  - load_sp_model
  - sentencepiece_numericalizer
  - sentencepiece_tokenizer
  - custom_replace
  - simple_space_split
  - numericalize_tokens_from_iterator
  - filter_wikipedia_xml
  - to_map_style_dataset
- torchtext.data.metrics
  - bleu_score

- torchtext.data.utils
  - get_tokenizer
  - ngrams_iterator
- torchtext.datasets
  - Text Classification
  - Language Modeling
  - Machine Translation
  - Sequence Tagging
  - Question Answer
  - Unsupervised Learning
- torchtext.vocab
  - Vocab
  - vocab
  - build_vocab_from_iterator
  - Vectors
  - GloVe
  - FastText
  - CharNGram

- torchtext.transforms
  - SentencePieceTokenizer
  - GPT2BPETokenizer
  - CLIPTokenizer
  - BERTTokenizer
  - VocabTransform
  - ToTensor
  - LabelToIndex
  - Truncate
  - AddToken
  - Sequential
  - PadTransform
  - StrToIntTransform

NLP

카운트 기반 핵심어 분석

핵심어

- 핵심어(Keyword)
  - 텍스트 자료의 중요한 내용을 압축적으로 제시하는 단어 또는 문구

- 핵심어 분석 방법
  - 텍스트에서 많이 등장하는 단어의 등장 빈도를 분석함으로써 핵심어를 추출
    - 단순 빈도를 제시하는 방법
    - TF-IDF(Term Frequency-Inverse Document Frequency, 어휘 빈도-문서 역빈도)를 계산

Counter

```
import matplotlib.pyplot as plt
from collections import Counter

article = '''
강원 동해안 해수욕장 다음 달 8일부터 52일간 개장
거리두기 해제로 피서객 1천500만 명 이상 방문 예상
입력 : 2022.06.05 07:00:06댓글 0
프린트카카오톡이메일페이스북트위터카카오스토리공유
경포 해수욕장 찾은 피서객. [연합뉴스 자료 사진]
사진설명경포 해수욕장 찾은 피서객. [연합뉴스 자료 사진]
강원 동해안 6개 시군의 해수욕장이 다음 달 8일부터 52일간 문을 연다.
강원도 환동해본부는 동해안 84개 해수욕장을 다음 달 8일 강릉과 양양을 시작으로 8월 28일까지 개장할 계획이라고 5일 밝혔다.
개장 시간은 오전 9시부터 오후 6시까지다.
일부 해수욕장은 성수기(7월 22일~8월 7일) 야간 입수를 허용할 예정이다.
강릉 경포·속초 해수욕장은 오후 9시까지, 주문진·옥계·정동진·사근진·강문·안목·사천진 등 강릉 지역 7개 해수욕장은 오후 7시까지 각각 입수 시간을 연장하는 방안을 검토하고 있다.
동해안 해수욕장 찾은 피서객.[연합뉴스 자료 사진]
사진설명동해안 해수욕장 찾은 피서객.[연합뉴스 자료 사진]
해수욕장 개장 기간에는 피서객들에 즐거움과 재미를 선사하기 위한 다양한 축제가 열린다.
강릉시는 다음 달 11~18일 경포 블루페스티벌을, 동해시는 망상 힙합 경연대회를 다음 달 28일부터 8월 2일까지 개최한다.
속초에서는 내달 30일부터 8월 5일까지 썸머 페스티벌이 열린다.
해수욕장을 찾을 피서객들은 바다 여행 홈페이지(바다 여행, seantour.kr)를 통해 미리 해변 혼잡 정도를 확인할 수 있다.
최성균 환동해본부장은 "올해 해수욕장 운영 목표를 관광객 2천만명 달성과 안전사고 제로로 정해 피서객 유치와 물놀이 사고 예방에 온 힘을 기울이겠다"고 말했다.
'''
kkma = Kkma()
tokens = kkma.nouns(article)
counted_tokens = Counter(tokens)
top_20 = counted_tokens.most_common(20)
print(top_20)
```

```
[('시', 3), ('강원', 2), ('일', 2), ('동해', 2), ('초', 2), ('동해안', 1), ('해수욕장', 1), ('다음', 1), ('달', 1), ('8', 1), ('8일', 1), ('52', 1), ('개장', 1), ('거리', 1), ('기', 1), ('해제', 1), ('피서객', 1),
```

Word Cloud

➢ 텍스트에 등장하는 단어를 그 등장 빈도에 따라 서로 크기가 다르게 구름형태로 표현

```
1  from wordcloud import WordCloud
2  wc = WordCloud(background_color="white", font_path='NanumBarunGothic.ttf')
3  wc.generate_from_frequencies(counted_tokens)
4  figure = plt.figure()
5  figure.set_size_inches(10, 10)
6  ax = figure.add_subplot(1, 1, 1)
7  ax.axis("off")
8  ax.imshow(wc)
```

```
<matplotlib.image.AxesImage at 0x7f308be485d0>
```

Word Cloud

```python
# 마스크 이미지 로드
import numpy as np
from PIL import Image
from os import path
import os

mask = np.array(Image.open("mask-cloud.png"))

# 워드 클라우드 설정
wc2 = WordCloud(background_color="white", mask=mask, contour_width=1,
                font_path='NanumBarunGothic.ttf')
wc2.generate_from_frequencies(counted_tokens)
# 이미지 표시
plt.imshow(wc2, interpolation='bilinear')
plt.axis("off")
# 이미지 저장
wc2.to_file("wordcloud.png")
```

```
<wordcloud.wordcloud.WordCloud at 0x7f3082443690>
```

TF IDF

```python
from sklearn.feature_extraction.text import TfidfVectorizer

documents = [
    "I like animals",
    "I like food",
    "I hate math",
    "I want to study math",
]

# TF-IDF로 벡터화
# 1글자도 인식이 되도록 토큰 패턴 변경
tf_idf = TfidfVectorizer(token_pattern = r"(?u)₩b₩w+₩b")
tf_idf.fit(documents)

print(tf_idf.vocabulary_)
```

```
{'i': 3, 'like': 4, 'animals': 0, 'food': 1, 'hate': 2, 'math': 5, 'want': 8, 'to': 7, 'study': 6}
```

TF IDF

```python
from sklearn.feature_extraction.text

documents = [
    "I like animals",
    "I like food",
    "I hate math",
    "I want to study math",
]

# TF-IDF로 벡터화
# 1글자도 인식이 되도록 토큰 패턴 변
tf_idf = TfidfVectorizer(token_patte
tf_idf.fit(documents)

print(tf_idf.vocabulary_)
```

```
{'i': 3, 'like': 4, 'animals': 0, 'food': 1, 'hate': 2, 'math': 5, 'want': 8, 'to': 7, 'study': 6}
```

```python
# 다른 문서에도 많이 나온 단어는 낮은 수치
tf_idf.transform(["I like animals"]).toarray()
```

```
array([[0.72664149, 0.        , 0.        , 0.37919167, 0.5728925 ,
        0.        , 0.        , 0.        , 0.        ]])
```

```python
# 같은 문서에 많이 나온 단어는 높은 수치
tf_idf.transform(["I like animals and love animals"]).toarray()
```

```
array([[0.90406978, 0.        , 0.        , 0.23589056, 0.3563895 ,
        0.        , 0.        , 0.        , 0.        ]])
```

TF IDF 기반 유사도 판단

➢ 데이터 준비

```
1  # 첫 번째 데이터 출력
2  data.head(1)
3
```

	adult	belongs_to_collection	budget	genres	homepage	id	imdb_id	original_language	original_title	overview	...
0	False	{'id': 10194, 'name': 'Toy Story Collection', ...	30000000	[{'id': 16, 'name': 'Animation'}, {'id': 35, '...	http://toystory.disney.com/toy-story	862	tt0114709	en	Toy Story	Led by Woody, Andy's toys live happily in his ...	...

1 rows × 24 columns

spoken_languages	status	tagline	title	video	vote_average	vote_count
[{'iso_639_1': 'en', 'name': 'English'}]	Released	NaN	Toy Story	False	7.7	5415.0

TF IDF 기반 유사도 판단

➢ 데이터 준비

- 각 영화에 대한 overview를 분석하여 비슷한 영화가 무엇인지 찾아보자

```
# 첫 번째 데이터의 overview 출력
# 영화에 대한 설명
data.head(1)["overview"][0]

```

```
"Led by Woody, Andy's toys live happily in his room until Andy's birthday brings Buzz Lightyear onto the scene. Afraid of losing his place in Andy's heart, Woody plots against Buzz. But when circumstances separate Buzz and Woody from their owner, the duo eventually learns to put aside their differences."
```

TF IDF 기반 유사도 판단

➢ 만들고 싶은 함수: get_similar()

```
1  get_similar("Toy Story", indices, cosine_sim)
2
```

```
2997             Toy Story 2
8327               The Champ
1071    Rebel Without a Cause
3057          Man on the Moon
1932               Condorman
Name: title, dtype: object
```

```
1  get_similar("Star Wars", indices, cosine_sim)
2
```

```
1154    The Empire Strikes Back
1167         Return of the Jedi
1267               Mad Dog Time
5187        The Triumph of Love
309           The Swan Princess
Name: title, dtype: object
```

TF IDF 기반 유사도 판단

➢ 만들고 싶은 함수: get_similar()

```
1  get_similar("Toy Story", indices, cosine_sim)
2
```

```
2997              Toy Story 2
8327                The Champ
1071     Rebel Without a Cause
3057           Man on the Moon
1932                Condorman
Name: title, dtype: object
```

```
1  get_similar("Star Wars", indices
2
```

```
1154    The Empire Strikes Back
1167         Return of the Jedi
1267                Mad Dog Time
5187         The Triumph of Love
309            The Swan Princess
Name: title, dtype: object
```

```
1  # 중복을 제거하여 영화 제목을 시리즈로 생성
2  indices = pd.Series(data.index, index=data["title"]).drop_duplicates()
3
4  print(indices.head())
```

```
title
Toy Story                    0
Jumanji                      1
Grumpier Old Men             2
Waiting to Exhale            3
Father of the Bride Part II  4
dtype: int64
```

TF IDF 기반 유사도 판단

➢ 만들고 싶은 함수: get_similar()

```
1  get_similar("Toy Story", indices, cosine_sim)
2
```

```
2997              Toy Story 2
8327                The Champ
1071    Rebel Without a Cause
3057          Man on the Moon
1932                Condorman
Name: title, dtype: object
```

```
1  get_similar("Star Wars", in
2
```

```
1154    The Empire Strikes Back
1167         Return of the Jedi
1267               Mad Dog Time
5187        The Triumph of Love
309          The Swan Princess
Name: title, dtype: object
```

```
1  from sklearn.feature_extraction.text import TfidfVectorizer
2
3  # TF-IDF 변환
4  tf_idf = TfidfVectorizer(stop_words="english")
5  tf_idf_matrix = tf_idf.fit_transform(data["overview"])
6
7  # 데이터의 개수 : 10000
8  # 단어의 개수 : 32350
9  print(tf_idf_matrix.shape)
10
```

```
(10000, 32350)
```

```
1  from sklearn.metrics.pairwise import linear_kernel
2
3  # 10000 x 10000 서로 내적하여 코사인 유사도를 구함
4  # 각 항목은 두 영화의 유사도를 나타냄
5  cosine_sim = linear_kernel(tf_idf_matrix, tf_idf_matrix)
```

TF IDF 기반 유사도 판단

```python
# 유사한 영화를 구함
def get_similar(title, indices, cosine_sim):

    # 영화의 인덱스를 구함
    try:
        index = indices[title]
    except:
        return None

    # 해당 영화의 유사도를 배열로 변환
    # 0 : 인덱스, 1 : 유사도
    scores = list(enumerate(cosine_sim[index]))

    # 유사도(x[1] 항목)를 기준으로 높은 순으로 정렬
    scores = sorted(scores, key=lambda x: x[1], reverse=True)

    # 가장 유사도가 높은 자신을 제외하고 5개를 추출
    scores = scores[1:6]

    # 인덱스를 구함
    indices = [x[0] for x in scores]

    # 각 인덱스의 영화 제목을 구함
    titles = data["title"].iloc[indices]

    return titles
```

TF IDF 기반 유사도 판단

```
get_similar("Toy Story", indices, cosine_sim)
```

```
2997              Toy Story 2
8327                The Champ
1071    Rebel Without a Cause
3057           Man on the Moon
1932                Condorman
Name: title, dtype: object
```

```
get_similar("Star Wars", indices, cosine_sim)
```

```
1154    The Empire Strikes Back
1167         Return of the Jedi
1267               Mad Dog Time
5187        The Triumph of Love
309           The Swan Princess
Name: title, dtype: object
```

Summary

➢카운트 기반 연산을 위해 사용한 기능
- from collections import **Counter**
- from wordcloud import **WordCloud**
- from sklearn.feature_extraction.text import **TfidfVectorizer**

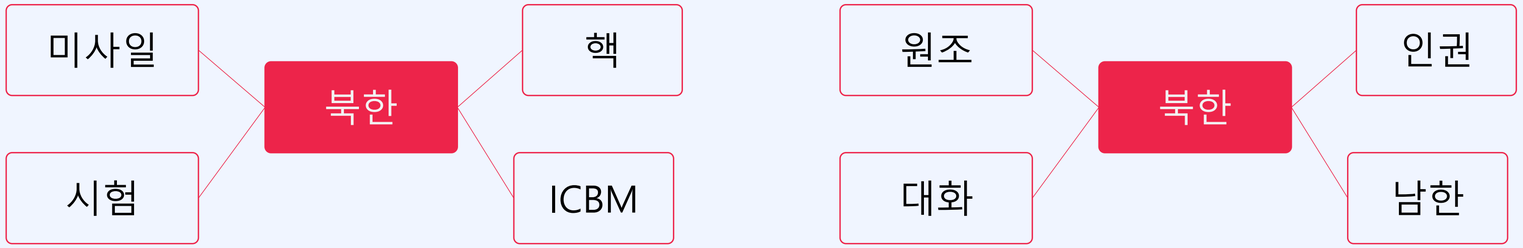
➢카운트 기반 연산으로 수행한 Task
- 핵심어 분석
  - WordCloud
- 유사도 분석

NLP

의미 연결망 분석 방법

의미 연결망 분석(Semantic Network Analysis)

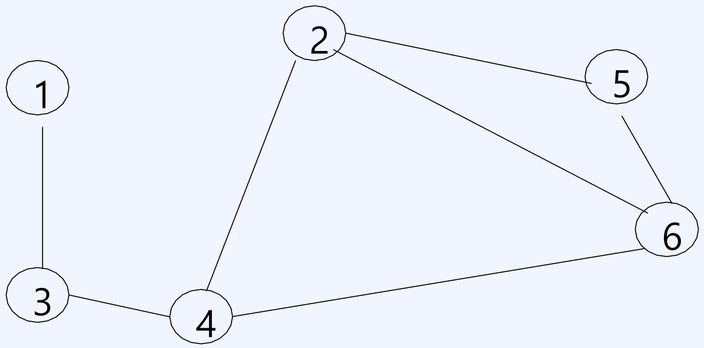
➢ 단어 간의 **관계를 분석**하기 위해 사용
- 사회 연결망 분석 기법을 단어의 관계에 적용하여 텍스트의 의미 구조를 파악하려는 분석 기법
- 특정 단어가 어떤 단어와 함께 자주 사용되었는가?
  - 문서의 저자가 강조하고자 하는 것이 무엇인지, 어떤 어조를 띄고 있는지 추측할 수 있음



➢ (참고)사회 연결망 분석(SNA, Social Network Analysis)
- 분석 대상(node)이 서로 어떻게 관련을 맺고 연결망(network)을 구성하는지 대상들 간의 관계를 연결망 구조로 표현하고 이를 계량적으로 제시하는 분석기법
- 사회학, 경영, 인문, 공학 등 다양한 분야에서 활용
  - 예) SNS에서 서로 어떻게 연결되는가 분석

의미 연결망 분석(Semantic Network Analysis)

➢ SNA 분석 방법
- 문서를 구성하고 있는 단어를 노드로 구성하고 노드와 노드를 연결
- 네트워크 = 문서를 구성하고 있는 단어와 관계



의미 연결망 분석(예시)

➢ '□ □ 피자' 시식평 SNA 분석

치즈매니아

피자는 치즈!

피자피자

토핑 종류가 많아요

굿피자

치즈도 많네요

의미 연결망 분석(예시)

➢ '□ □ 피자' 시식평 SNA 분석

1) 원문

- 피자는 치즈!, 토핑 종류가 많아요, 치즈도 많네요

2) 텍스트 정제

- 피자, 치즈, 토핑, 종류, 많다,

3) 인접행렬(Adjacent Matrix)

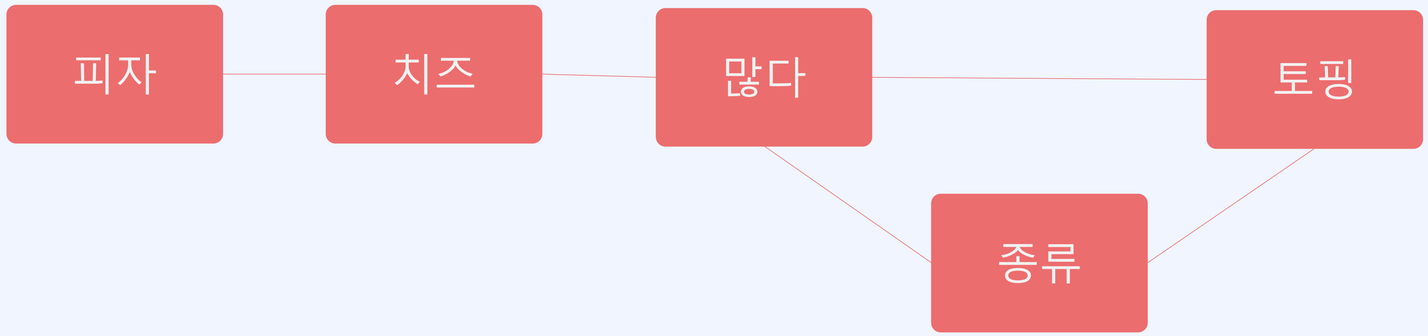
	피자	치즈	많다	토핑	종류
피자	0	1	0	0	0
치즈	1	0	1	0	0
많다	0	1	0	1	1
토핑	0	0	1	0	1
종류	0	0	1	1	0

의미 연결망 분석(예시)

➢ '□ □ 피자' 시식평 SNA 분석

4) 연결망 그래프

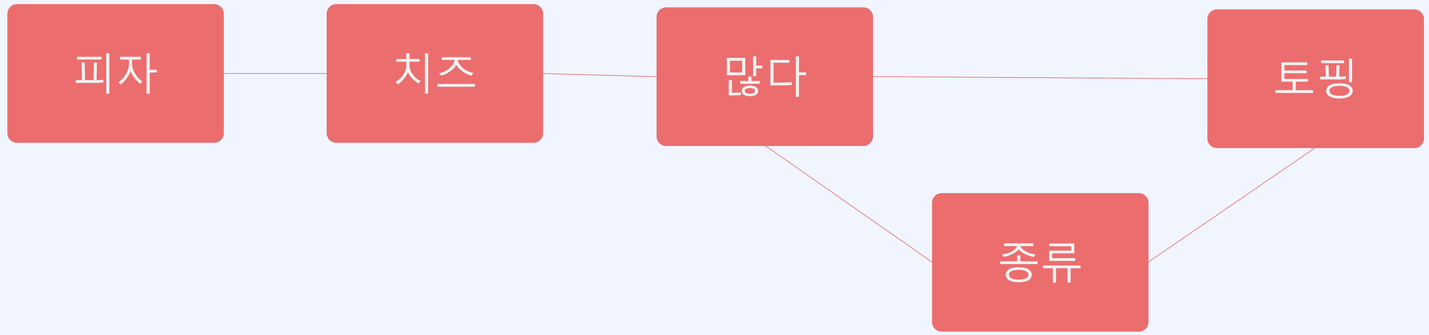
	피자	치즈	많다	토핑	종류
피자	0	1	0	0	0
치즈	1	0	1	0	0
많다	0	1	0	1	1
토핑	0	0	1	0	1
종류	0	0	1	1	0



의미 연결망 분석(예시)

➢ '□ □ 피자' 시식평 SNA 분석

5) SNA 속성



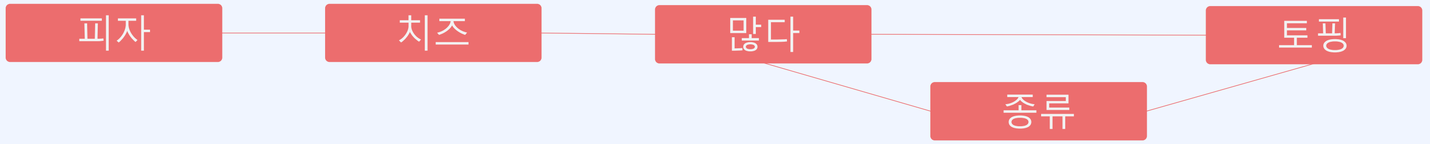
	피자	치즈	많다	토핑	종류
연결정도	1	2	3	2	2
연결중심성					
매개중심성					
근접중심성					

의미 연결망 분석(예시)

➢ '□□ 피자' 시식평 SNA 분석

5) SNA 속성: 연결 중심성(Degree centrality)

- 친구 수 (degree)가 많은 사람이 더 중심적 역할(한 단어에 직접 연결된 다른 단어의 수가 얼마나 많은지 측정)
- 연결망의 크기에 따라 값을 비교하기 어렵기 때문에, 표준화 필요
- $\frac{\text{특정 노드 i와 직접 연결된 노드 수}}{\text{노드 i와 직간접적으로 연결된 모든 노드 수}}$



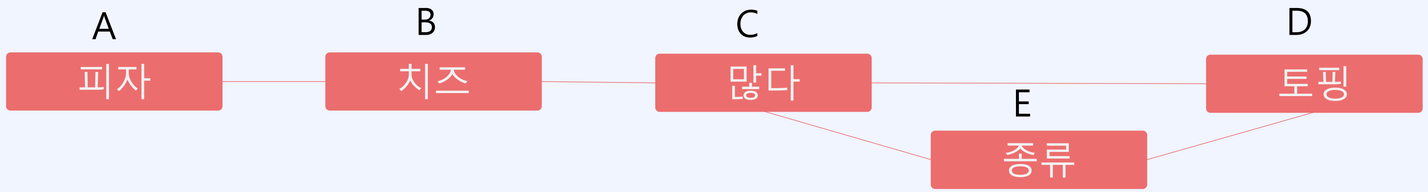
	피자	치즈	많다	토핑	종류
연결정도	1	2	3	2	2
연결중심성	¼=0.25	2/4=0.5	¾=0.75	2/4=0.5	2/4=0.5
매개중심성					
근접중심성					

의미 연결망 분석(예시)

➢ '□ □ 피자' 시식평 SNA 분석

5) SNA 속성: 매개 중심성(Betweenness centrality)

- 다리(bridge) 역할을 많이 하는 사람이 더 중심적 역할
- 한 단어가 다른 단어들과의 연결망을 구축하는 데 매개자 역할을 얼마나 수행하는지 측정
- 단어들의 등장 빈도가 낮더라도 매개 중심성이 높으면 단어들 간 의미 부여 역할이 커짐



	피자	치즈	많다	토핑	종류
연결정도	1	2	3	2	2
연결중심성	¼=0.25	2/4=0.5	¾=0.75	2/4=0.5	2/4=0.5
근접중심성	3/6=0.5	3/6=0.5	4/6=0.66	3/6=0.5	3/6=0.5
매개중심성					

$$\frac{D\text{를 포함하여 노드를 연결이 가능한경우의 수}}{D\text{를 제외한 노드 간의 가능한 연결의 경우의 수}}$$

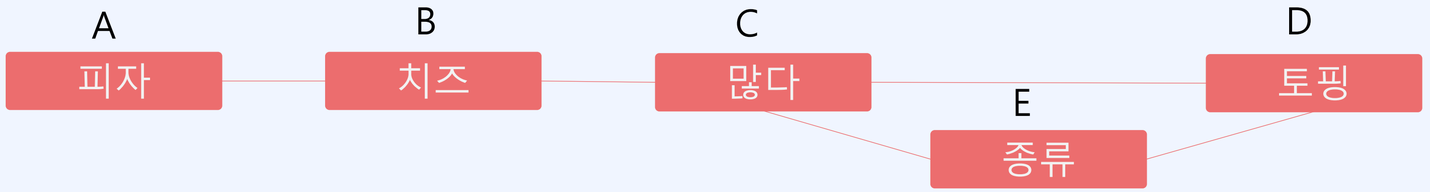
$$= \frac{(A,E),(B,E),(C,E)}{(A,B),(A,C),(A,E),(B,C),(B,E),(C,E)} = \frac{3}{6}$$

의미 연결망 분석(예시)

➢ '□ □ 피자' 시식평 SNA 분석

5) SNA 속성: 근접 중심성(Closeness centrality)

- 다른 노드와 더 가깝게 연결된 노드가 더 중심적 역할
- 한 단어가 다른 단어에 얼마나 가깝게 있는지 측정
- 직접 연결 뿐 아니라 간접적으로 연결된 모든 단어들 사이의 거리를 측정



$$= \frac{\text{전체 노드 수} - 1}{(A,B)\text{의 최단 경로} + (A,C)\text{의 최단 경로} + (A,D)\text{의 최단 경로} + (A,E)\text{의 최단 경로}}$$

$$= \frac{5-1}{1+2+3+3} = \frac{4}{9}$$

	피자	치즈	많다	토핑	종류
연결정도	1	2	3	2	2
연결중심성	¼=0.25	2/4=0.5	¾=0.75	2/4=0.5	2/4=0.5
매개중심성	0/6=0	3/6=0.5	4/6=0.66	3/6=0.5	3/6=0.5
근접중심성	4/9=0.44	4/7=0.57	4/5=0.8	4/7=0.57	4/7=0.57

SNA를 위한 파이썬 패키지

- NetworkX
  - 네트워크 분석을 위한 패키지
  - https://networkx.github.io/documentation/stable/tutorial.html

[IN]:
```
import networkx as nx

# 네트워크 생성
g = nx.Graph()

# 노드 생성
g.add_node(1)
g.add_nodes_from([1, 2, 3, 4, 5, 6]) # 중복으로 추가해도 에러 없음

# 노드 정보 확인
g.nodes()
```

[OUT]:
```
NodeView((1, 2, 3, 4, 5, 6))
```

SNA를 위한 파이썬 패키지

➢ NetworkX

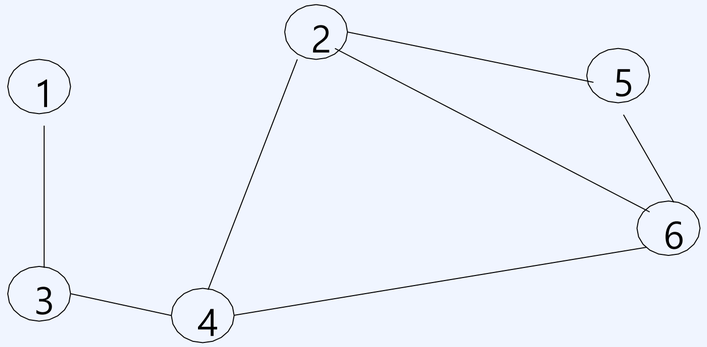
[IN]:

```
# 노드와 노드 연결(엣지 생성)
g.add_edge(1,3)
g.add_edges_from([(1,3), (2,4), (2,5), (2,6), (3,4), (4,6), (5,6)])

# 엣지 정보 확인
g.edges()
```

[OUT]:

```
EdgeView([(1, 3), (2, 4), (2, 5), (2, 6), (3, 4), (4, 6), (5, 6)])
```



SNA를 위한 파이썬 패키지

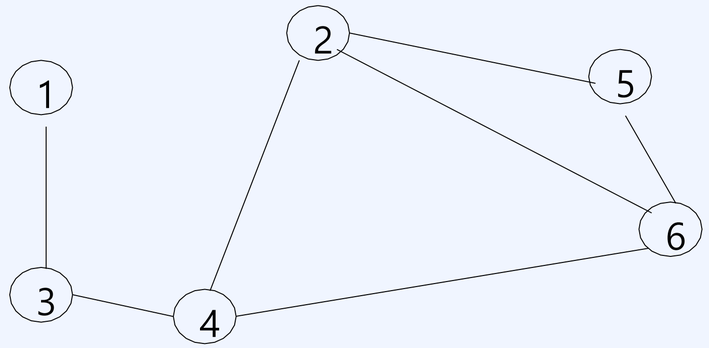
➢ NetworkX

[IN]:
```
#노드 수
print(g.number_of_nodes())
```

[OUT]: 6

[IN]:
```
#엣지 수
print(g.number_of_edges())
```

[OUT]: 7



SNA를 위한 파이썬 패키지

➢ NetworkX

[IN]:

```
#노드의 속성 설정
g.nodes[1]['gender']='male'
g.nodes[2]['gender']='female'
g.nodes[3]['gender']='male'
g.nodes[4]['gender']='female'
g.nodes[5]['gender']='male'
g.nodes[6]['gender']='male'

#노드의 속성 출력
print(nx.get_node_attributes(g, 'gender'))
```

[OUT]:

```
{1: 'male', 2: 'female', 3: 'male', 4: 'female', 5: 'male', 6: 'male'}
```

SNA를 위한 파이썬 패키지

➢ NetworkX

[IN]:

```
#노드 간의 연결 강도 설정
g[1][3]['weight'] = 3
g[2][4]['weight'] = 1
g[2][5]['weight'] = 4
g[2][6]['weight'] = 3
g[3][4]['weight'] = 2
g[4][6]['weight'] = 3
g[5][6]['weight'] = 4

print(nx.get_edge_attributes(g, 'weight'))
```

[OUT]:

```
{(1, 3): 3, (2, 4): 1, (2, 5): 4, (2, 6): 3, (3, 4): 2, (4, 6): 3, (5, 6): 4}
```

SNA를 위한 파이썬 패키지

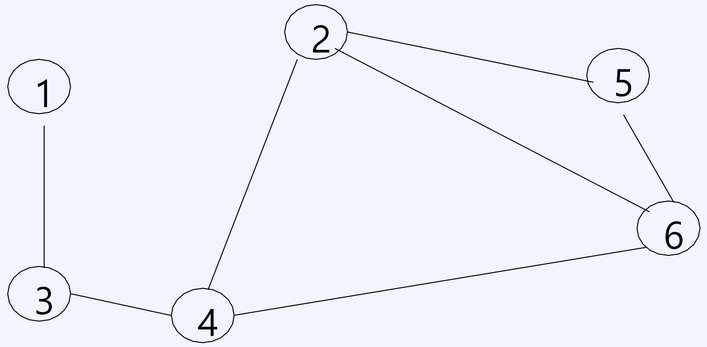
➢ NetworkX

[IN]:
```
#노드 4와 연결된 노드
g[4]
```

[OUT]:
```
AtlasView({2: {'weight': 1}, 3: {'weight': 2}, 6: {'weight': 3}})
```

[IN]:
```
#노드 4와 연결된 노드 수
len(list(g.neighbors(4)))
```

[OUT]:
```
3
```



SNA를 위한 파이썬 패키지

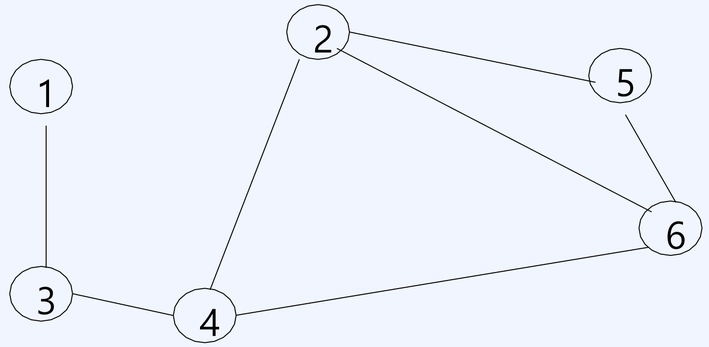
➢ NetworkX

[IN]:
```
#노드 4와 연결된 노드 수
g.degree(4)
```

[OUT]: 3

[IN]:
```
#노드 4와 노드 2간의 연결 강도
g[4][2]
```

[OUT]: {'weight': 1}



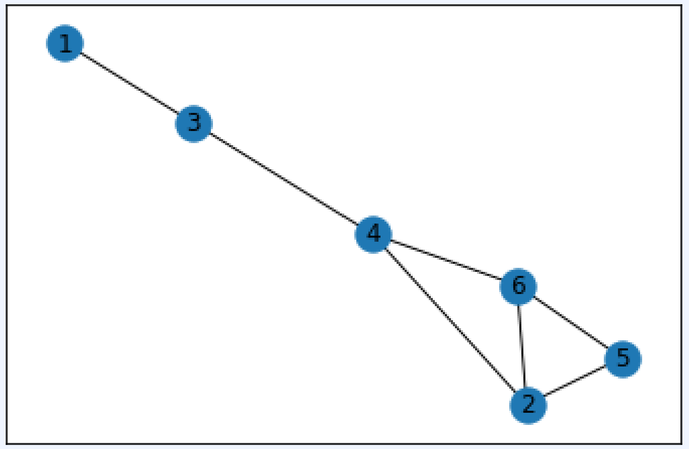
SNA를 위한 파이썬 패키지

➢ NetworkX

[IN]:

```
#그래프 시각화
import matplotlib.pyplot as plot
nx.draw_networkx(g)
plot.show()
```

[OUT]:



SNA를 위한 파이썬 패키지

➢ NetworkX

[IN]:
```
#연결중심성
nx.degree_centrality(g)
```

[OUT]:
```
{1: 0.2, 2: 0.6000000000000001, 3: 0.4, 4: 0.6000000000000001, 5:
0.4, 6: 0.6000000000000001}
```

[IN]:
```
#매개중심성
nx.betweenness_centrality(g)
```

[OUT]:
```
{1: 0.0, 2: 0.15000000000000002, 3: 0.4, 4: 0.6000000000000001,
5: 0.0, 6: 0.15000000000000002}
```

[IN]:
```
#근접중심성
nx.closeness_centrality(g)
```

[OUT]:
```
{1: 0.38461538461538464, 2: 0.625, 3: 0.5555555555555556, 4:
0.7142857142857143, 5: 0.45454545454545453, 6: 0.625}
```

Quiz

Q **다음 중 핵심어(keyword) 분석 효과로 알맞지 않은 것은?**

① 텍스트의 주제가 무엇인지 짐작 가능

② 텍스트가 서로 어느 정도 비슷한지 파악

③ SNA와 같은 관계망은 핵심어 분석에 적합하지 않음

④ 검색 엔진에서 검색결과의 우선 순위를 결정

해설

▶ SNA를 활용하여 단어와 단어 간의 관계망을 분석하여 연결중심성, 매개중심성, 근접중심성을 기준으로 핵심어를 파악할 수 있다.

Quiz

Q **다음 중 의미 연결망 분석(Semantic Network Analysis)에 대한 설명으로 옳지 않은 것은?**

① 사회 연결망 분석 기법을 단어의 관계에 적용하여 텍스트의 의미 구조를 파악하려는 분석 기법이다.

② 문서의 저자가 강조하고자 하는 것이 무엇인지, 어떤 어조를 띄고 있는지 추측할 수 있다.

③ 연결이 많이 된 단어만 핵심어로 판단하고, 매개자 역할이나 거리는 고려하지 않는다.

④ 문서를 구성하고 있는 단어를 노드로 구성하고 노드와 노드를 연결한다.

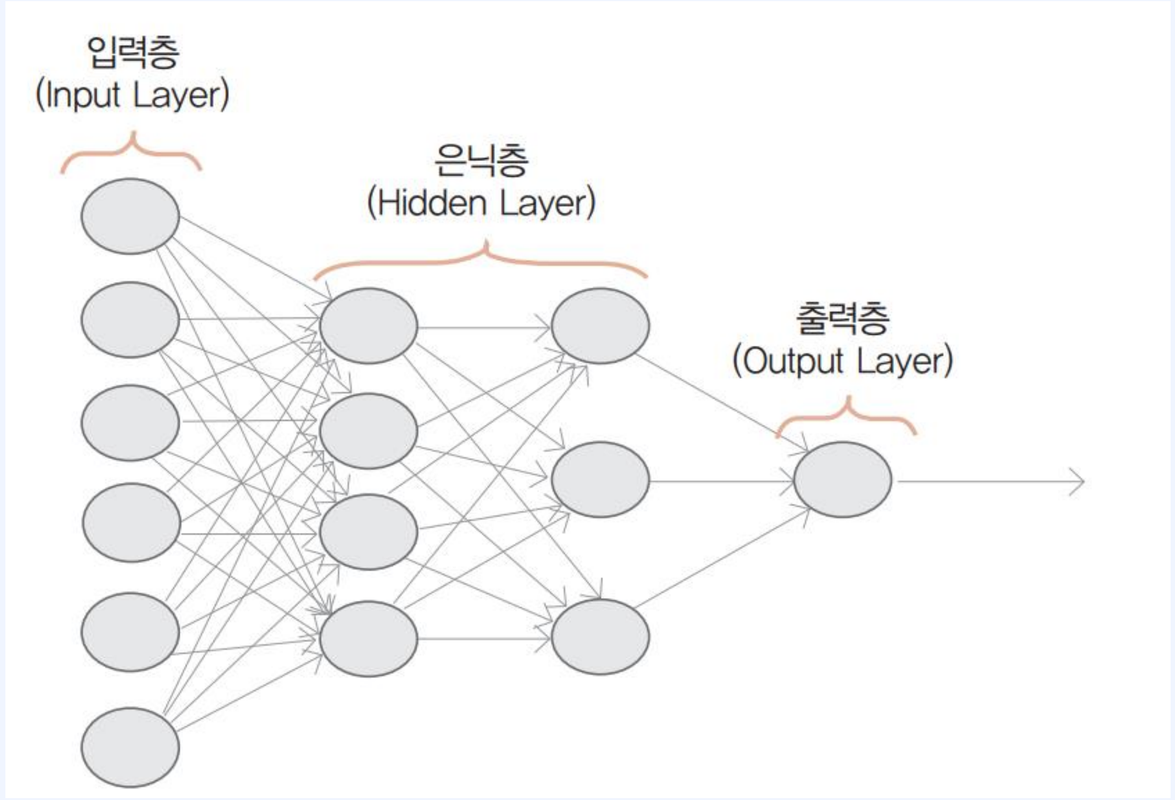
해설

▶ SNA를 활용하여 단어와 단어 간의 관계망을 분석하여 연결중심성(연결이 많이 된 단어만 핵심어로 판단), 매개중심성(매개자 역할), 근접중심성(단어 간의 거리)을 기준으로 핵심어를 파악할 수 있다.

NLP

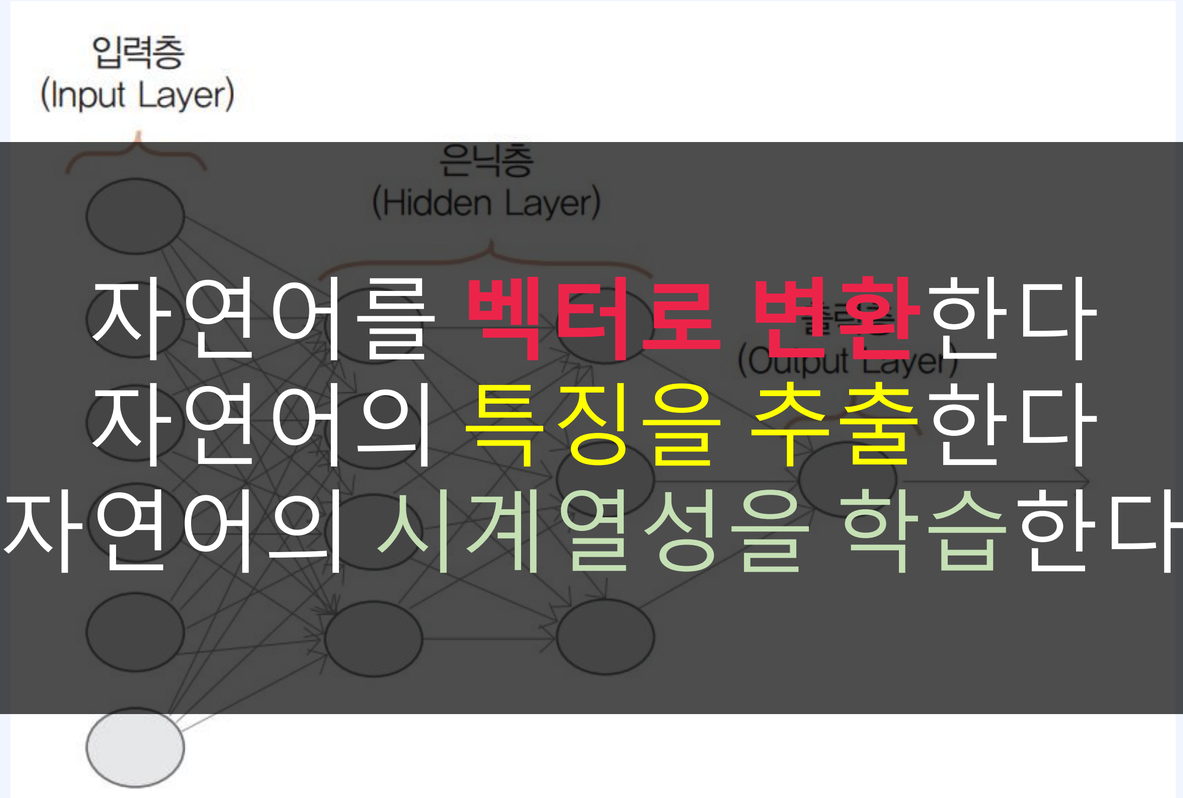
자연어 처리 분야에서 사용하는 신경망의 이해

기본 신경망



이미지출처: 파이썬으로 시작하는 머신러닝+딥러닝(아이리포)

자연어 + 인공신경망

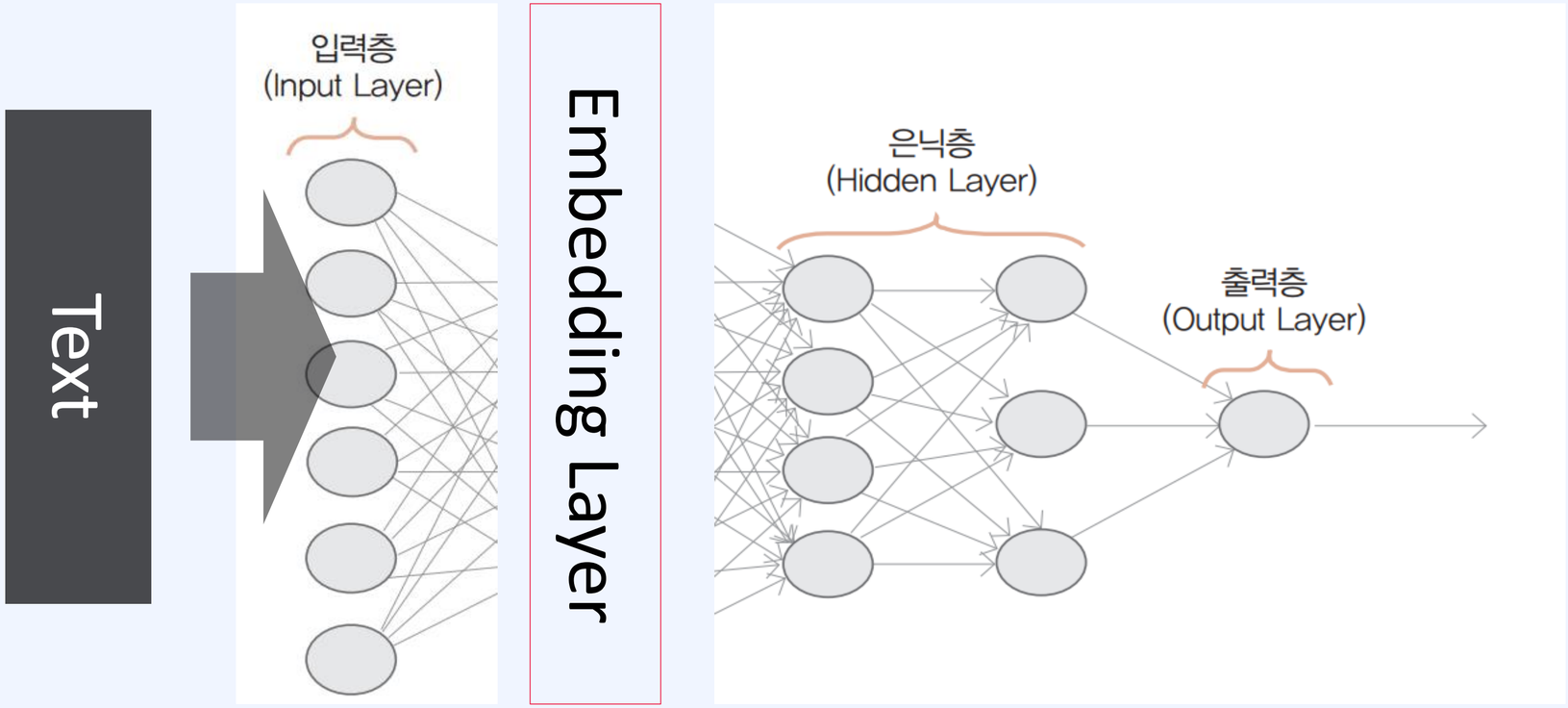


자연어를 **벡터로 변환**한다
자연어의 **특징을 추출**한다
자연어의 **시계열성을 학습**한다

이미지출처: 파이썬으로 시작하는 머신러닝+딥러닝(아이리포)

자연어 + 인공신경망

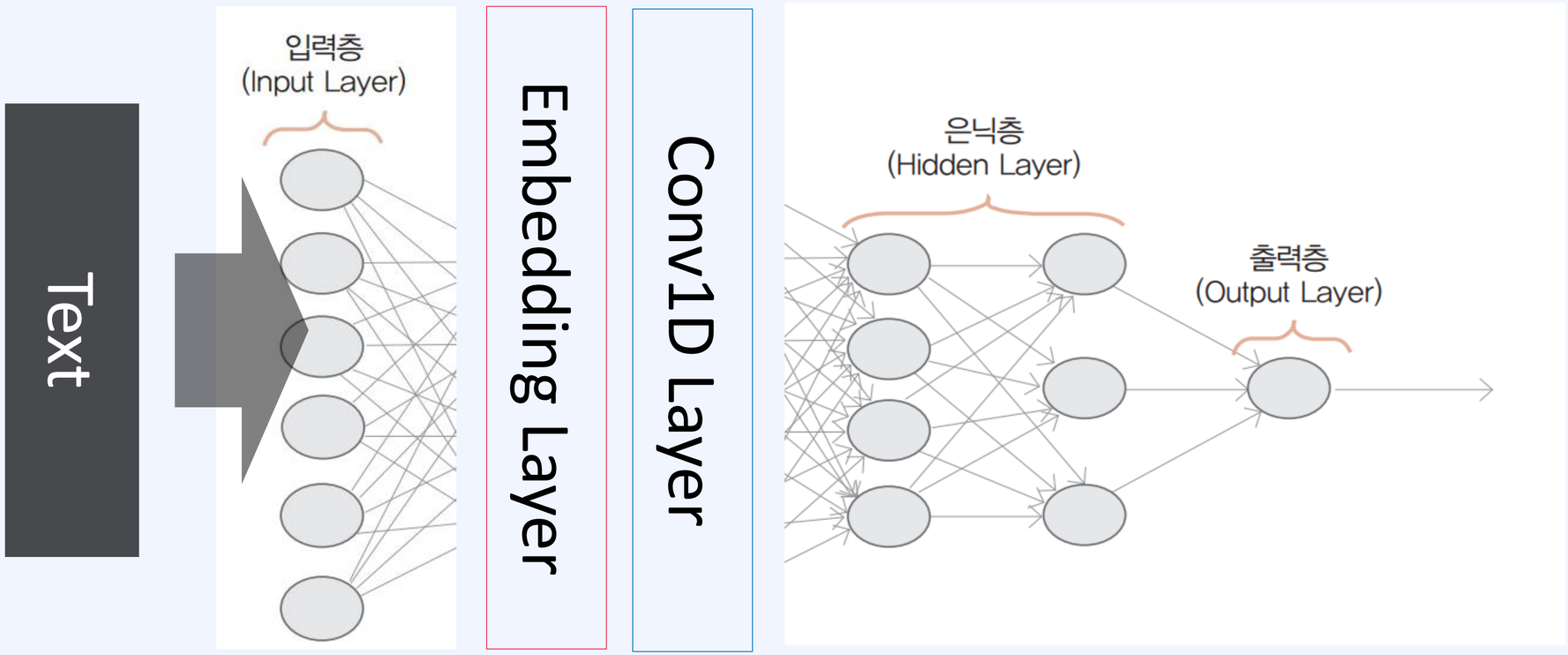
➢ 자연어를 **벡터로 변환**한다 → Embedding Layer



이미지출처: 파이썬으로 시작하는 머신러닝+딥러닝(아이리포)

자연어 + 인공신경망

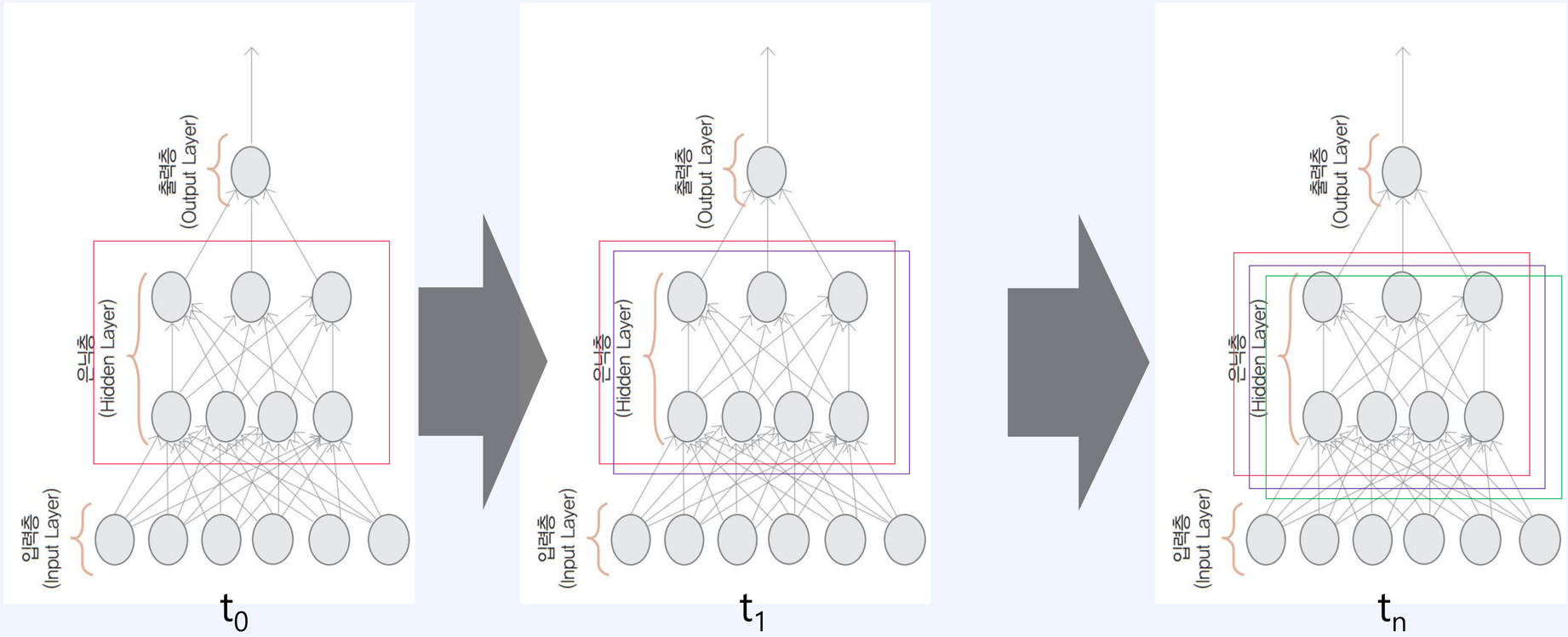
➢ 자연어의 **특징을 추출**한다 → Conv1D Layer



이미지출처: 파이썬으로 시작하는 머신러닝+딥러닝(아이리포)

자연어 + 인공신경망

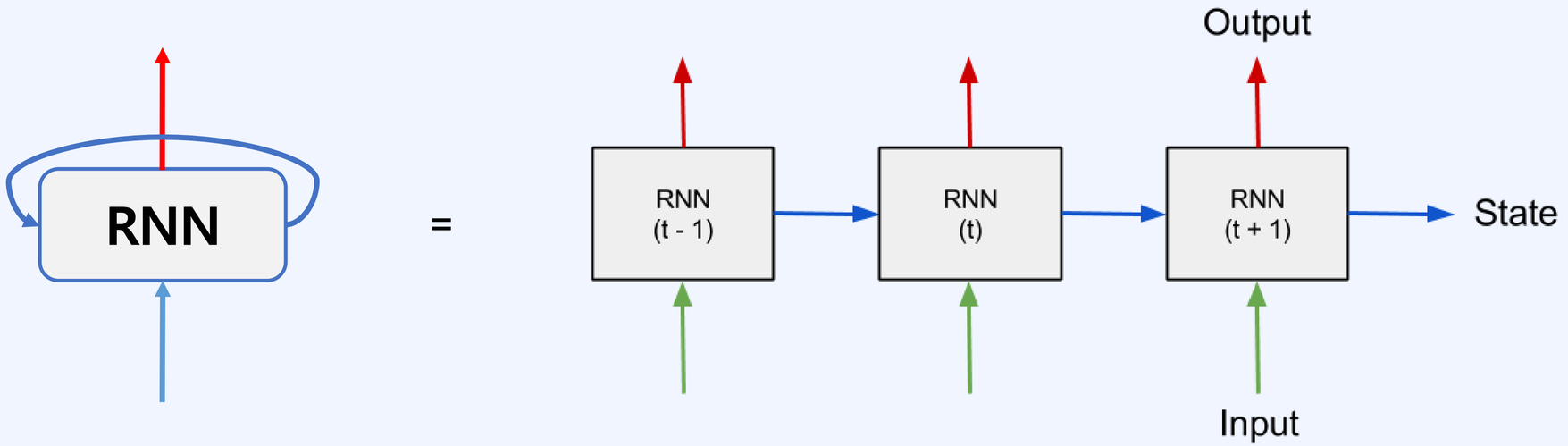
➢ 자연어의 **시계열성을 학습**한다 → RNN



이미지출처: 파이썬으로 시작하는 머신러닝+딥러닝(아이리포)

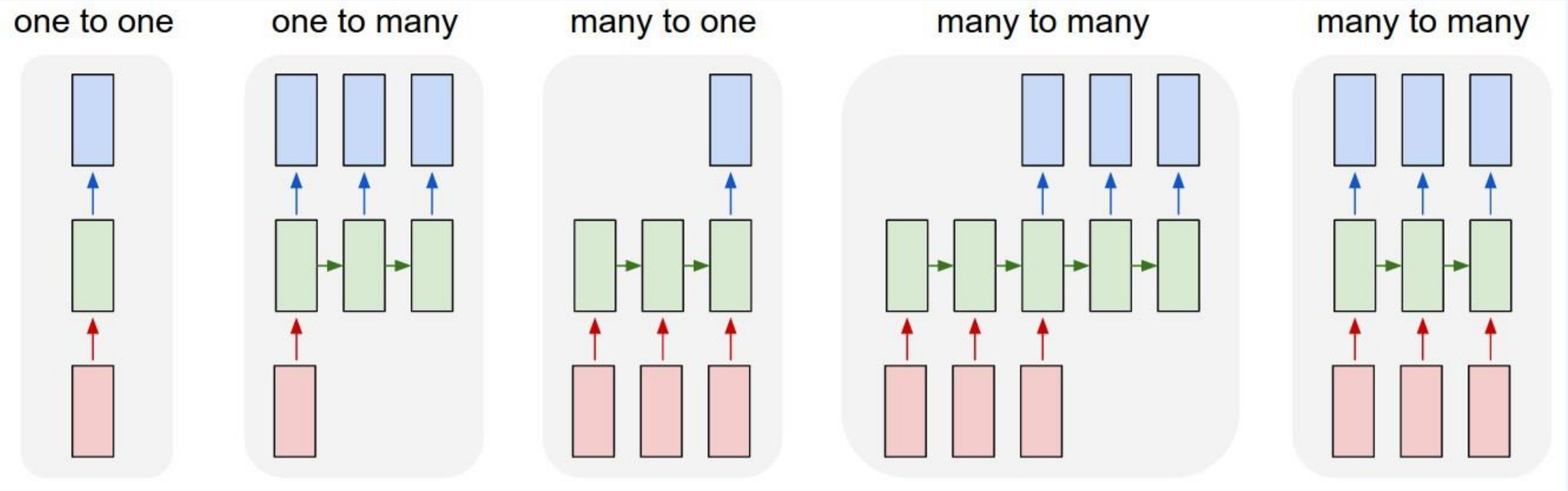
자연어 + 인공신경망

➢ 자연어의 **시계열성을 학습**한다 → RNN



이미지 출처: https://www.kdnuggets.com/2017/04/build-recurrent-neural-network-tensorflow.html

RNN



이미지 출처: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

RNN

➢ RNN(Recurrent Neural Network)

- 유닛 사이의 연결이 **방향성 있는 사이클(Directed Cycle)**을 형성하며 자신을 가리키는 반복 가중치 구조(Recurrent Weight)를 포함하는 신경망 알고리즘

- 연속된 데이터 상에서 이전 순서의 hidden node의 값을 저장하고, 다음 순서의 입력 데이터로 학습할 때 사용

- Sequential Data 학습
  - 과거학습의 정보를 잃지 않고 연속적인 정보의 흐름을 학습에 반영 ($y_{k-1} \rightarrow y_k \rightarrow y_{k+1}$)

Text classifier code example

```python
# A integer input for vocab indices.
inputs = tf.keras.Input(shape=(None,), dtype="int64")

# Next, we add a layer to map those vocab indices into a space of dimensionality
# 'embedding_dim'.
x = layers.Embedding(max_features, embedding_dim)(inputs)
x = layers.Dropout(0.5)(x)

# Conv1D + global max pooling
x = layers.Conv1D(128, 7, padding="valid", activation="relu", strides=3)(x)
x = layers.Conv1D(128, 7, padding="valid", activation="relu", strides=3)(x)
x = layers.GlobalMaxPooling1D()(x)

# We add a vanilla hidden layer:
x = layers.Dense(128, activation="relu")(x)
x = layers.Dropout(0.5)(x)

# We project onto a single unit output layer, and squash it with a sigmoid:
predictions = layers.Dense(1, activation="sigmoid", name="predictions")(x)

model = tf.keras.Model(inputs, predictions)

# Compile the model with binary crossentropy loss and an adam optimizer.
model.compile(loss="binary_crossentropy", optimizer="adam", metrics=["accuracy"])
```

https://keras.io/examples/nlp/text_classification_from_scratch/

Text classifier code example

```
# Input for variable-length sequences of integers
inputs = keras.Input(shape=(None,), dtype="int32")
# Embed each integer in a 128-dimensional vector
x = layers.Embedding(max_features, 128)(inputs)
# Add 2 bidirectional LSTMs
x = layers.Bidirectional(layers.LSTM(64, return_sequences=True))(x)
x = layers.Bidirectional(layers.LSTM(64))(x)
# Add a classifier
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs, outputs)
model.summary()
```

https://keras.io/examples/nlp/bidirectional_lstm_imdb/

Summary

➢자연어 처리 분야에서 사용하는 신경망의 이해

- 자연어를 **벡터로 변환**한다 → _____
- 자연어의 **특징을 추출**한다 → _____
- 자연어의 **시계열성을 학습**한다 → _____

Summary

➢자연어 처리 분야에서 사용하는 신경망의 이해

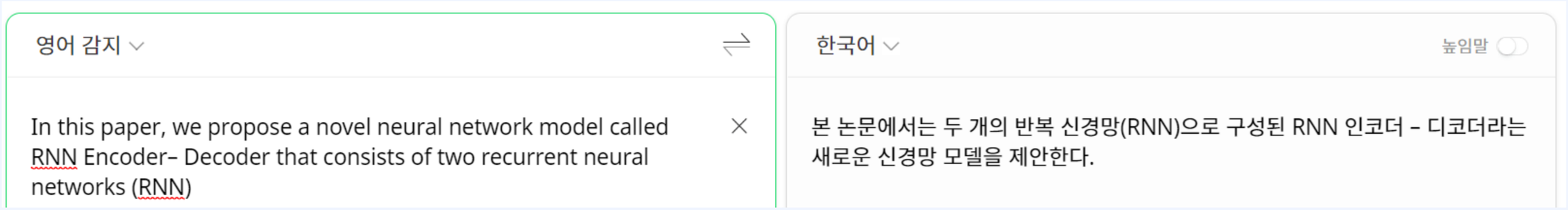
- 자연어를 **벡터로 변환**한다 →Embedding Layer
- 자연어의 **특징을 추출**한다 → Conv1D Layer
- 자연어의 **시계열성을 학습**한다 → RNN

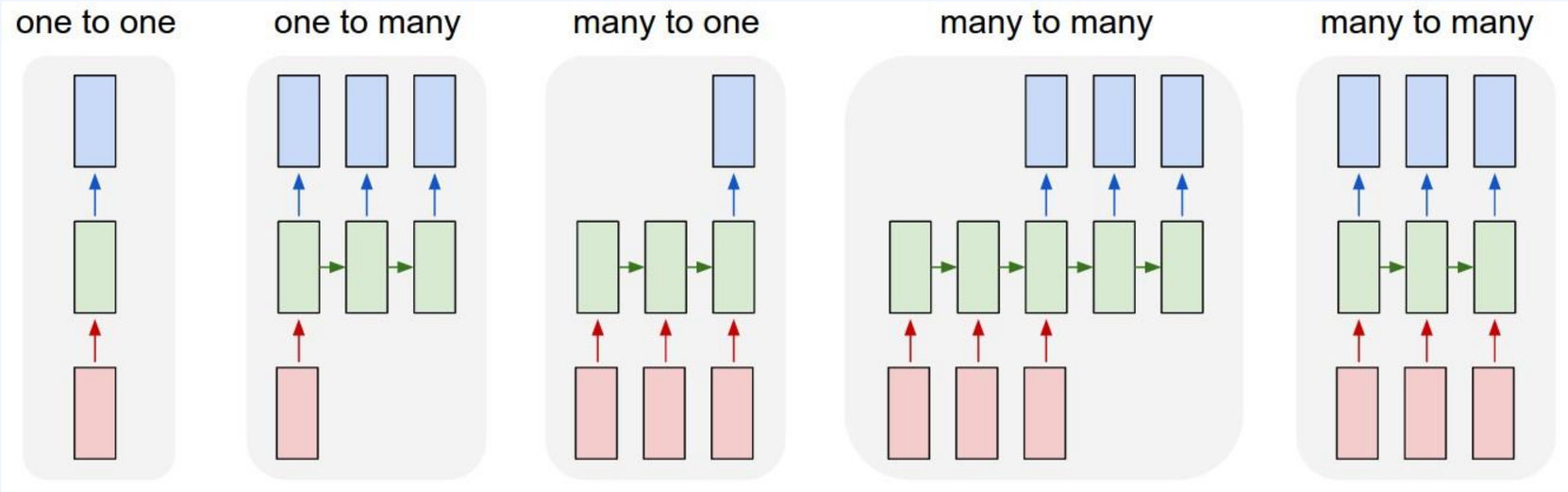
NLP

Seq2Seq

번역?!

➢ many to many task

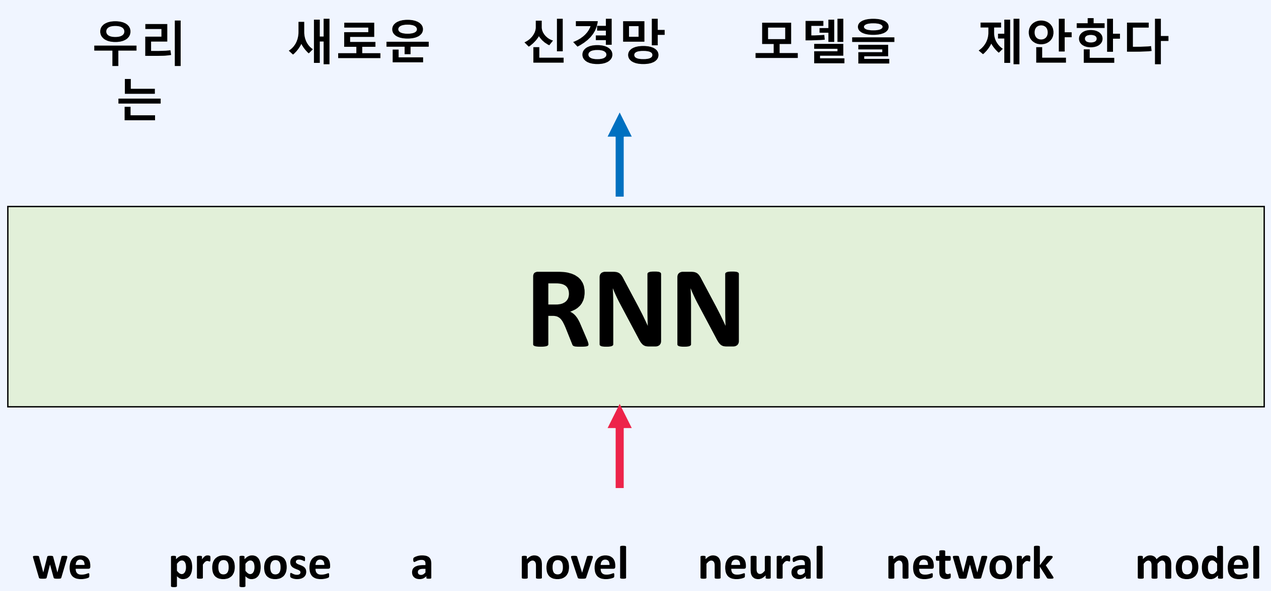




이미지 출처: 파파고, http://karpathy.github.io/2015/05/21/rnn-effectiveness/

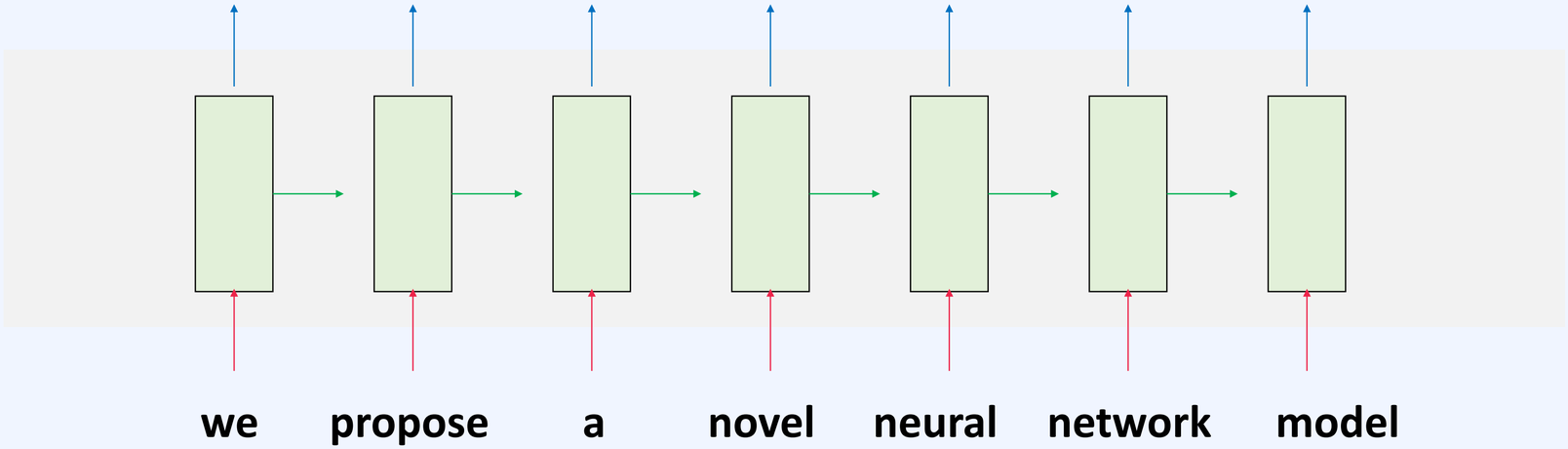
번역?!

➢ many to many task



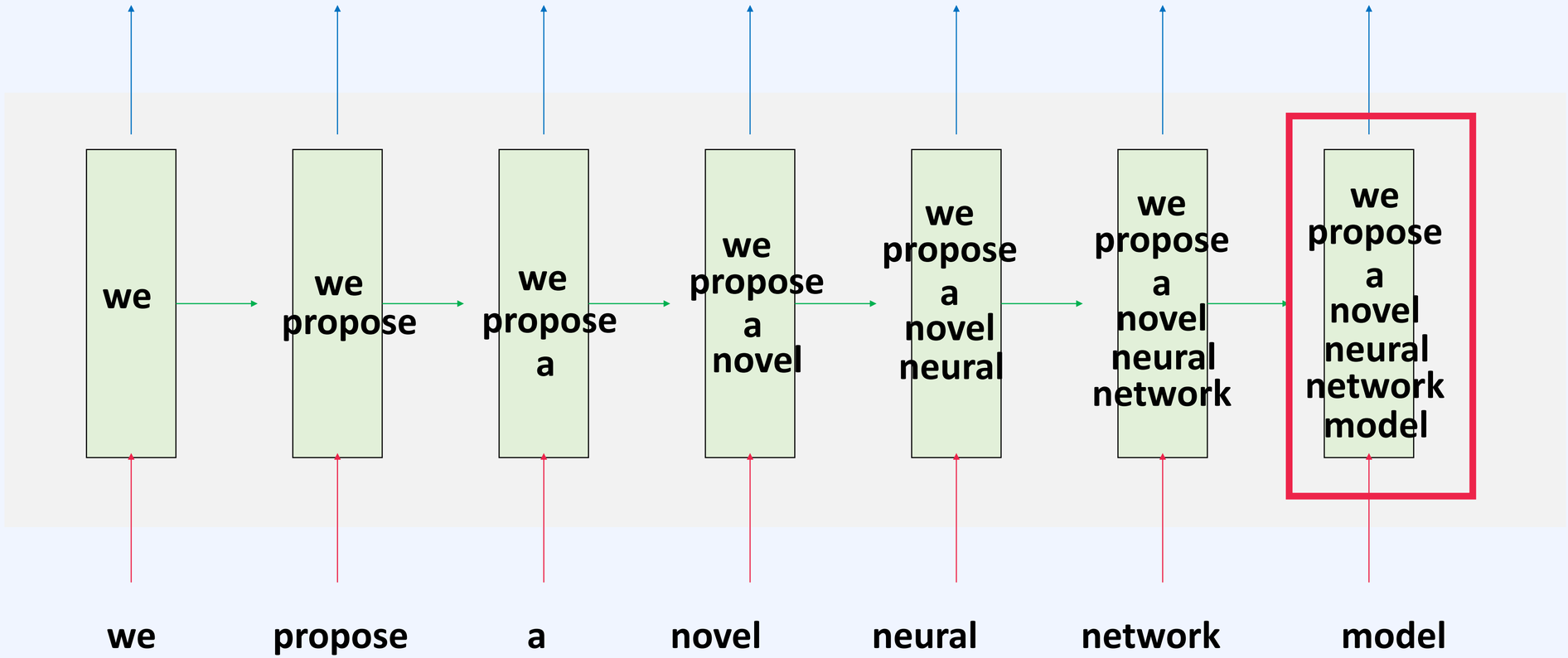
번역?!

➢ many to many task



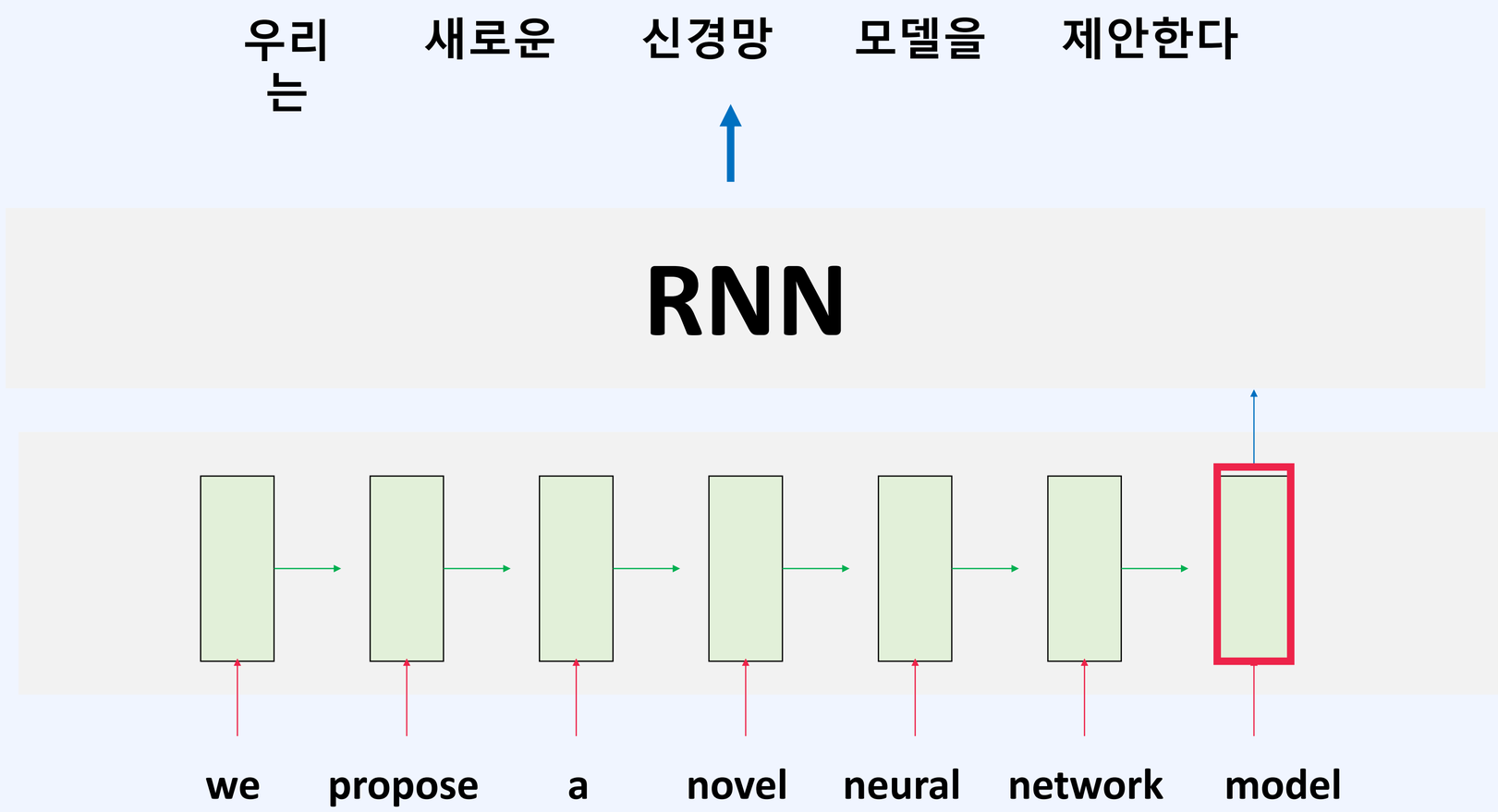
번역?!

➢ many to many task



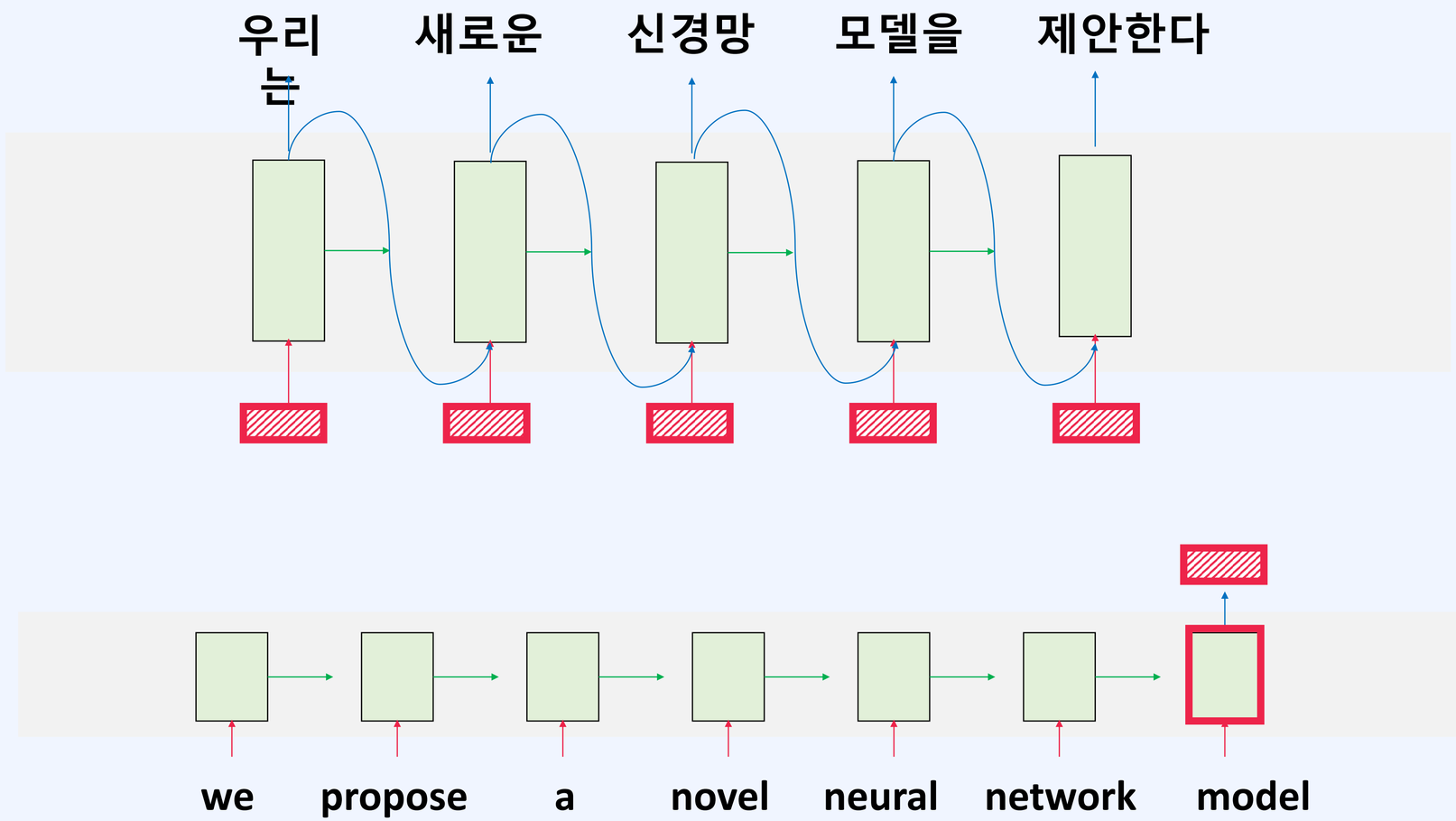
번역?!

➢ many to many task



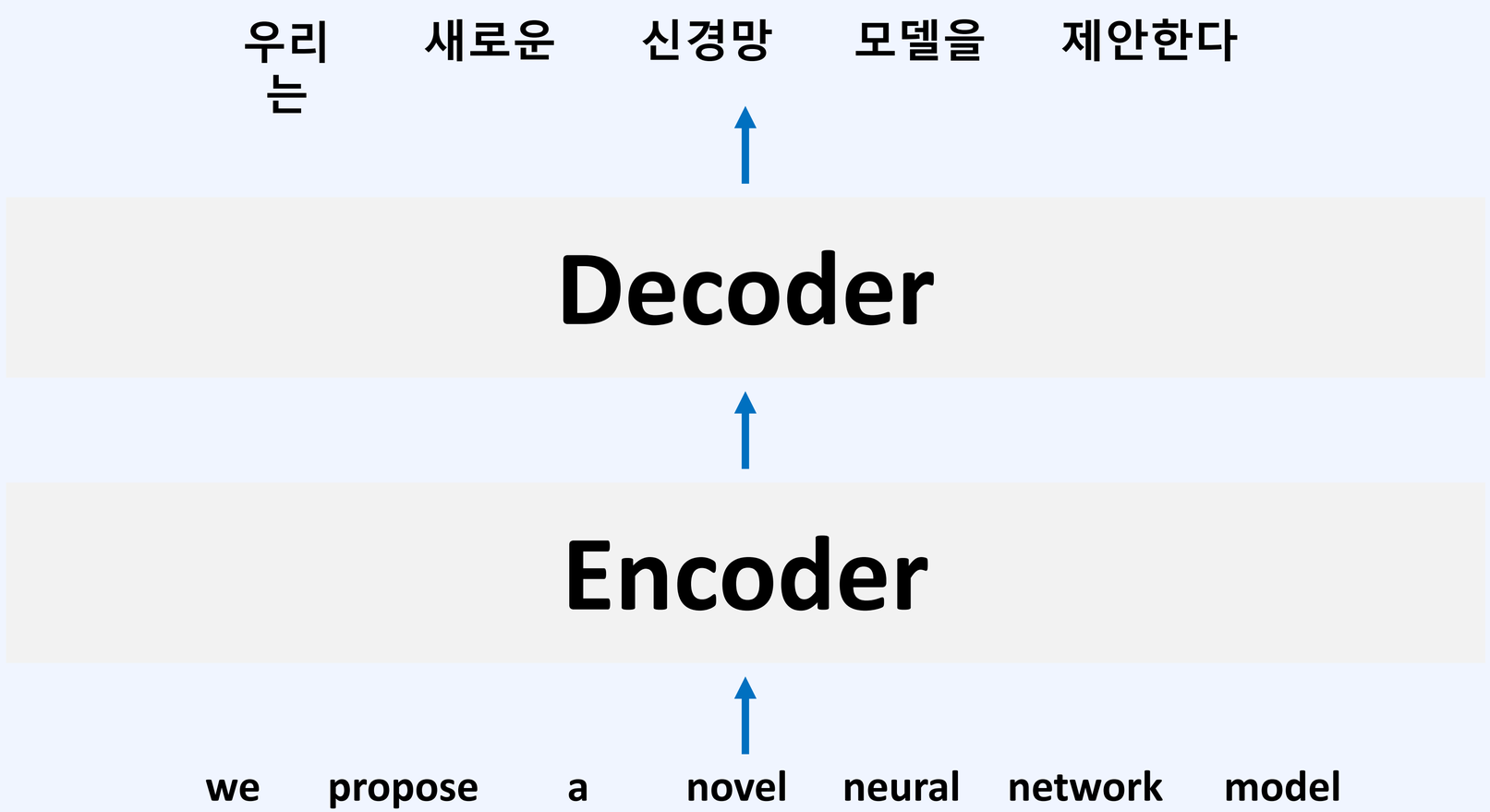
번역?!

➢ many to many task



번역?!

➢ many to many task



Sequence to Sequence(Seq2Seq)

- Paper: Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation(Kyunghyun Cho et al., 2014)

In this paper, we propose a novel neural network model called RNN Encoder–Decoder that consists of two recurrent neural networks (RNN). One RNN encodes a sequence of symbols into a fixed-length vector representation, and the other decodes the representation into another sequence of symbols. The encoder and de-

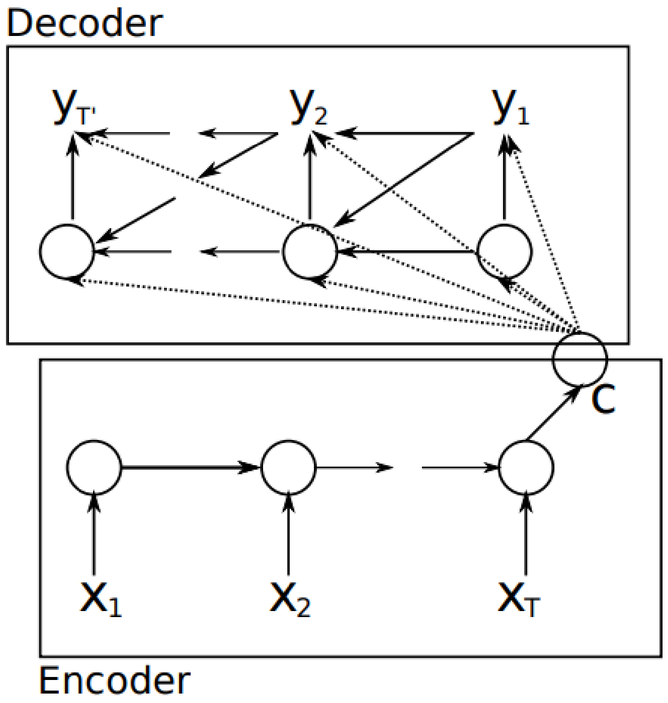


Figure 1: An illustration of the proposed RNN Encoder–Decoder.

Summary

- Seq2Seq
  - Encoder + Decoder
  - Encoder
    - variable length source sequence
    - fixed length vector
  - Decoder
    - fixed length vector의 conditional probability를 maximize
    - variable length target sequence

Quiz

Q **seq2seq 에 대한 설명으로 옳지 않은 것은?**

① seq2seq 모델은 인코더-디코더 구조로 이루어진다.

② 인코더는 차원을 축소하여 하나의 context vector로 데이터를 압축하여 출력한다.

③ seq2seq 언어모델은 문장을 이해하기 위해 문법 계층 구조 정보를 사용한다.

④ 시퀀스 데이터를 입력 받아 다른 도메인의 시계열 데이터로 출력한다..

해설

▶ seq2seq는 문장의 문법 계층 구조 정보가 아닌 시퀀스 데이터를 사용한다.

Quiz

Q **seq2seq 에 대한 설명으로 옳지 않은 것은?**

① 인코더는 여러 개의 벡터를 입력으로 받아 문장을 함축하는 벡터로 만들어낸다.

② 문장이 길어질수록 압축 성능이 떨어진다.

③ 인코더, 디코더로 구성된 구조이다.

④ 디코더는 주어진 문장을 차원 축소하여 잠재공간의 어떤 하나의 점에 투영하는 작업을 수행한다.

해설

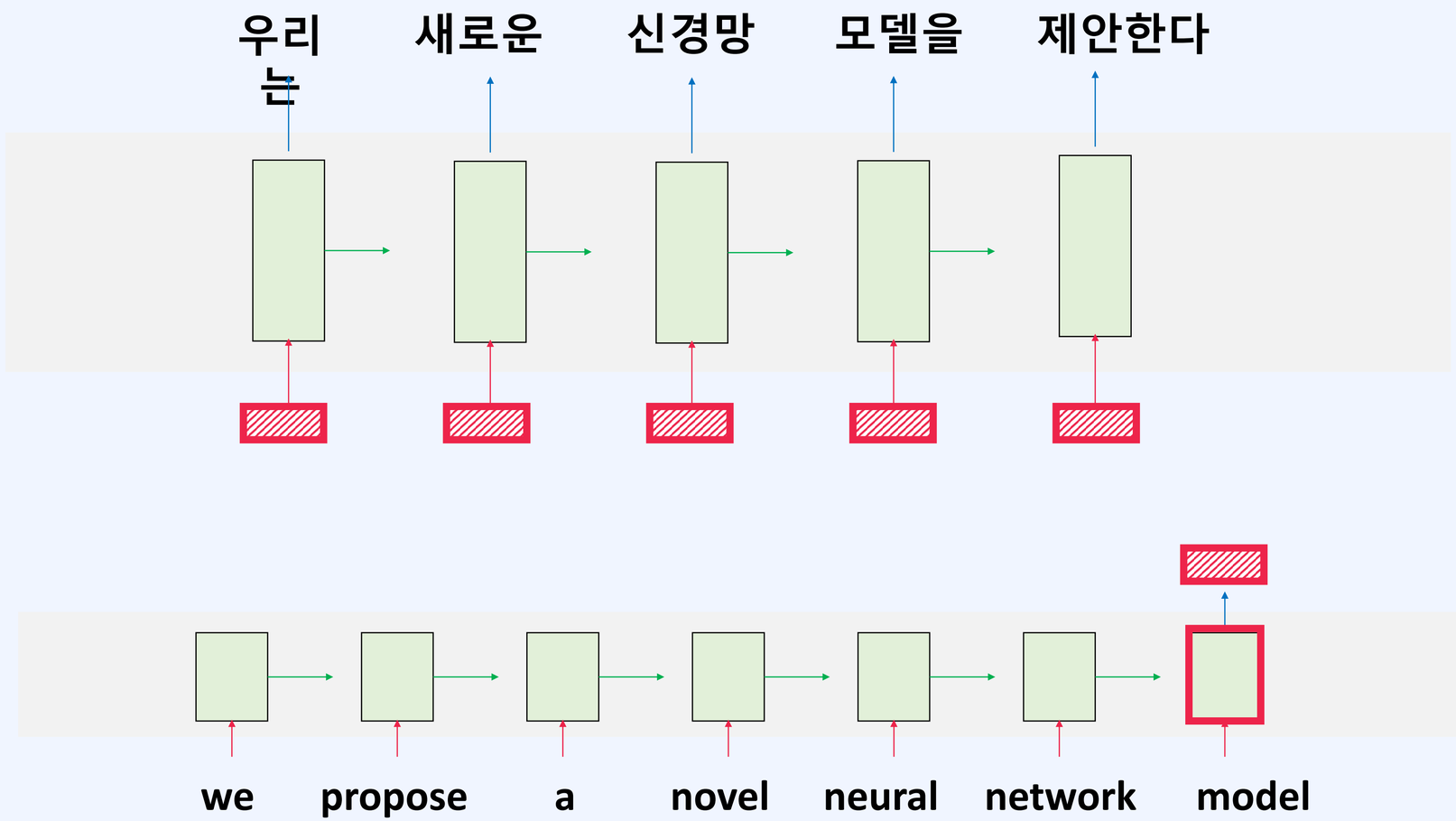
▶ ④는 인코더에 대한 설명이다. 디코더는 인코더의 결과와 이전 time-step까지 번역하여 생성한 단어들에 기반하여 단어를 생성한다.

NLP

Attention

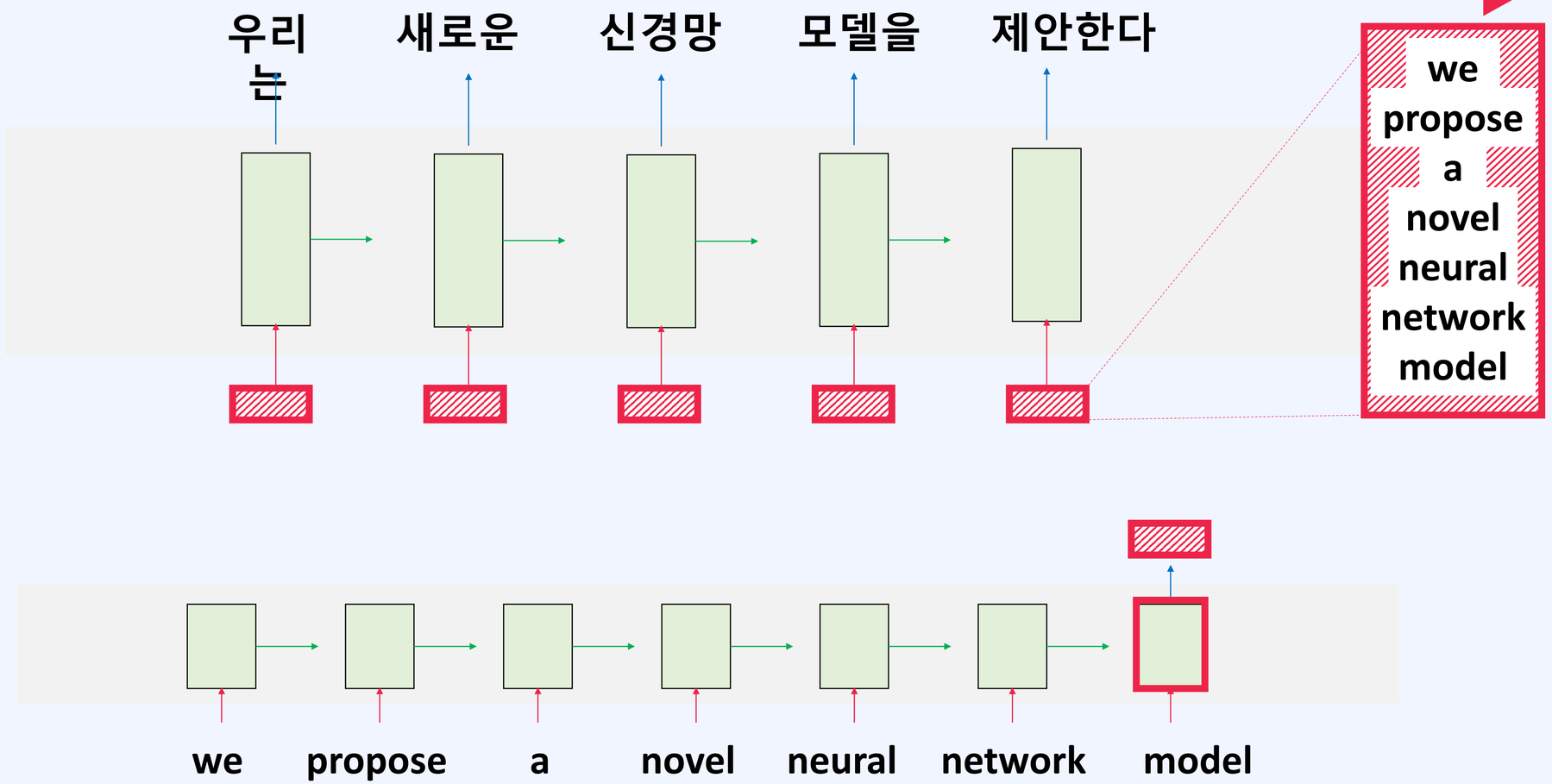
Seq2Seq

➢ Static context vector



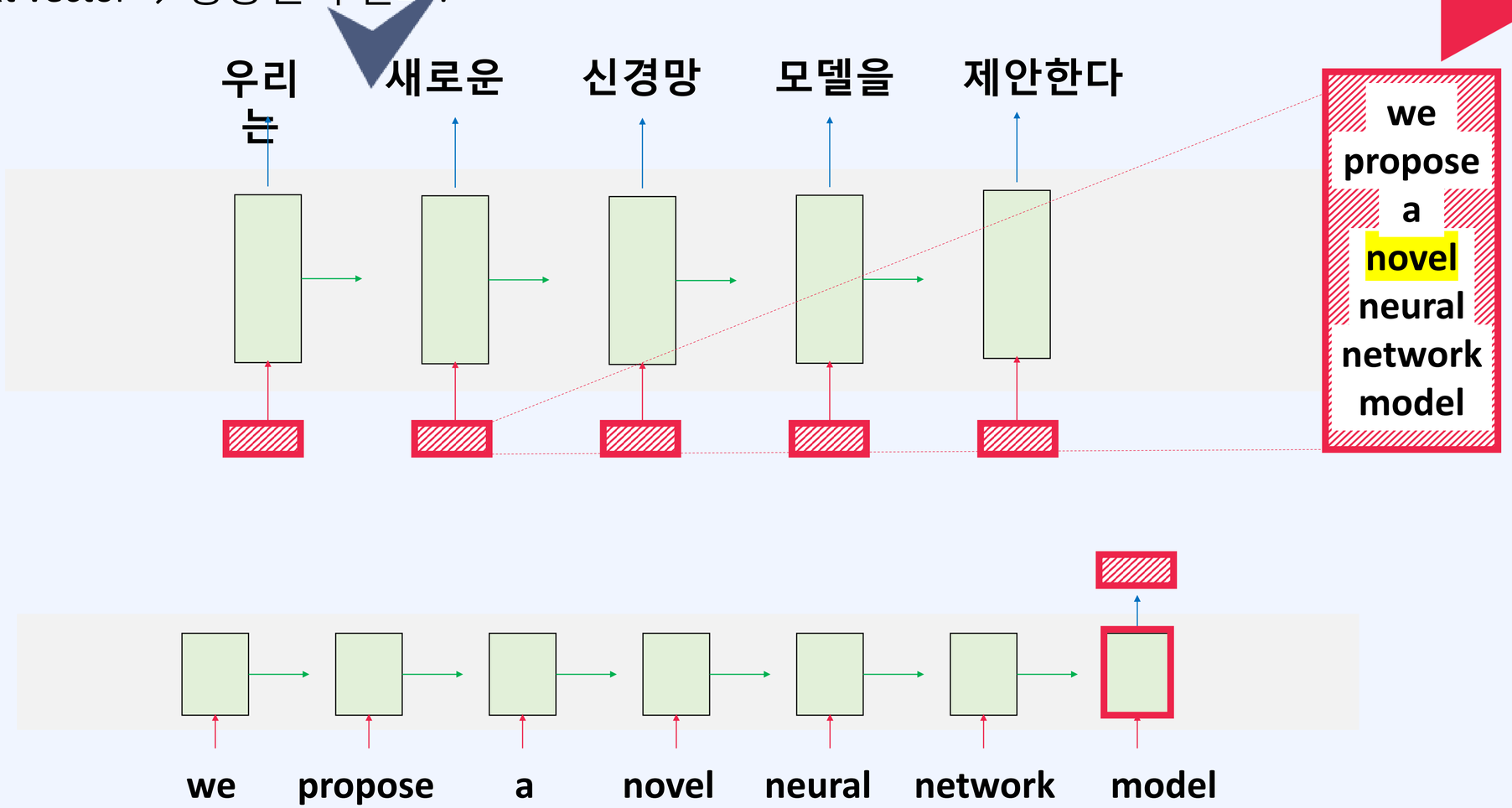
Seq2Seq

➢ 문장이 길어지니까 어렵네...



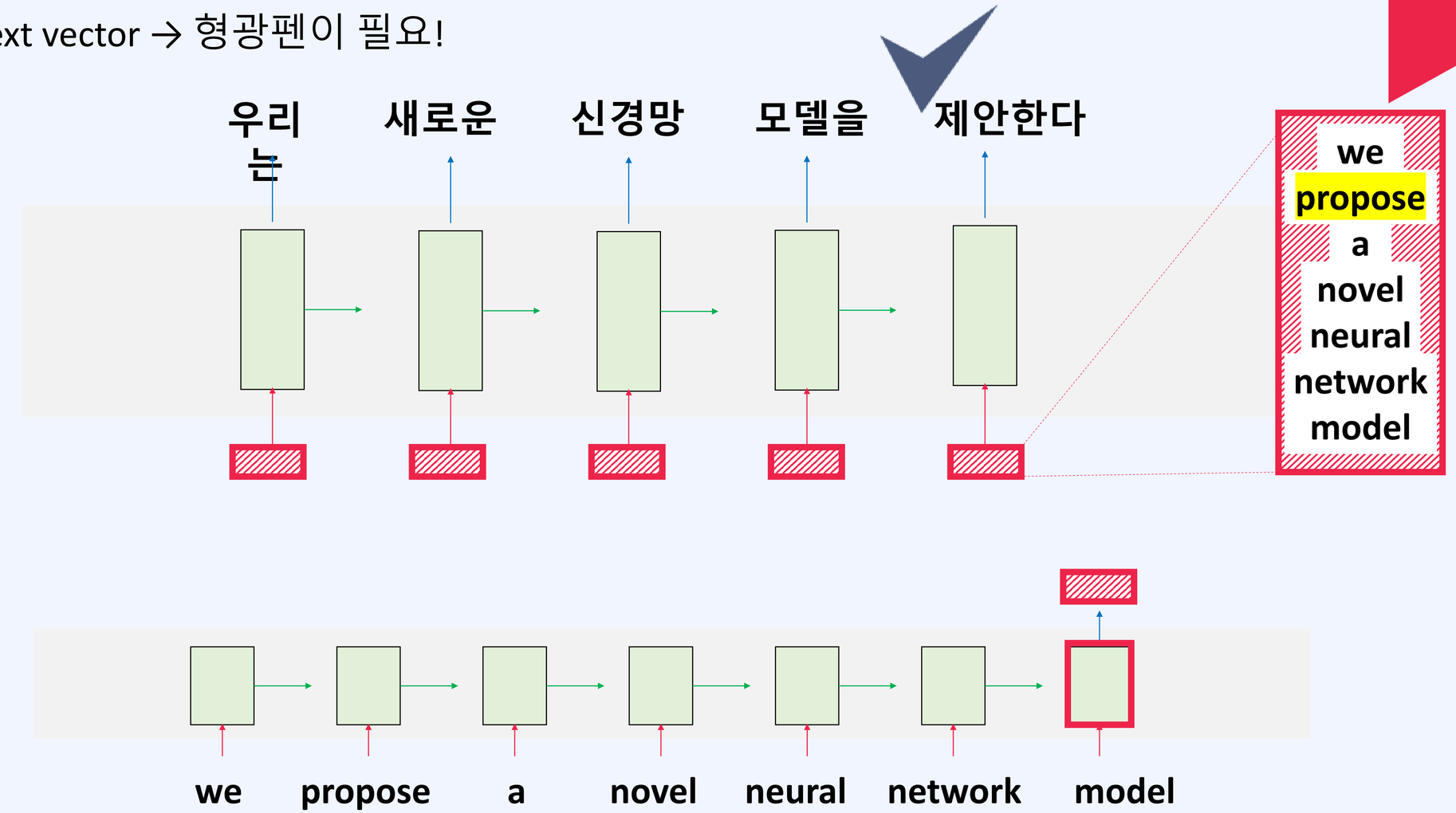
Attention

➢ Static context vector → 형광펜이 필요!



Attention

➢ Static context vector → 형광펜이 필요!



Attention

- Paper: Neural Machine Translation by Jointly Learning to Align and Translate(Dzmitry Bahdanau et al., ICLR 2015)

Neural machine translation is a recently proposed approach to machine translation. Unlike the traditional statistical machine translation, the neural machine translation aims at building a single neural network that can be jointly tuned to maximize the translation performance. The models proposed recently for neural machine translation often belong to a family of encoder–decoders and encode a source sentence into a fixed-length vector from which a decoder generates a translation. In this paper, we conjecture that the use of a fixed-length vector is a bottleneck in improving the performance of this basic encoder–decoder architecture, and propose to extend this by allowing a model to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word, without having to form these parts as a hard segment explicitly. With this new approach, we achieve a translation performance comparable to the existing state-of-the-art phrase-based system on the task of English-to-French translation. Furthermore, qualitative analysis reveals that the (soft-)alignments found by the model agree well with our intuition.

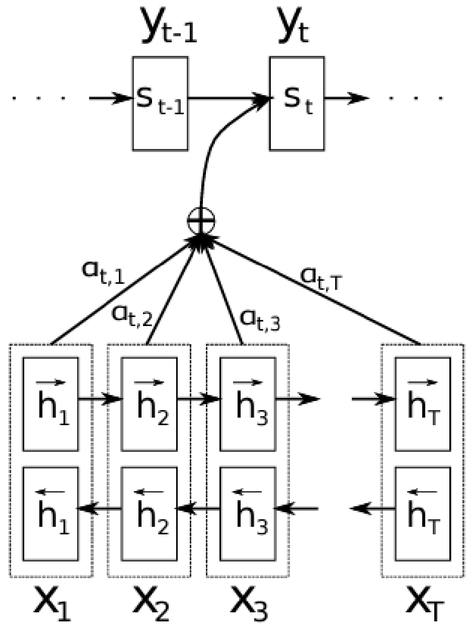


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

Attention

- Paper: Neural Machine Translation by Jointly Learning to Align and Translate(Dzmitry Bahdanau et al., ICLR 2015)

Decoder: context vector, s_{t-1}, y_{t-1}

Context vector: weighted sum of Hidden state vector

Encoder: Bidirectional RNN

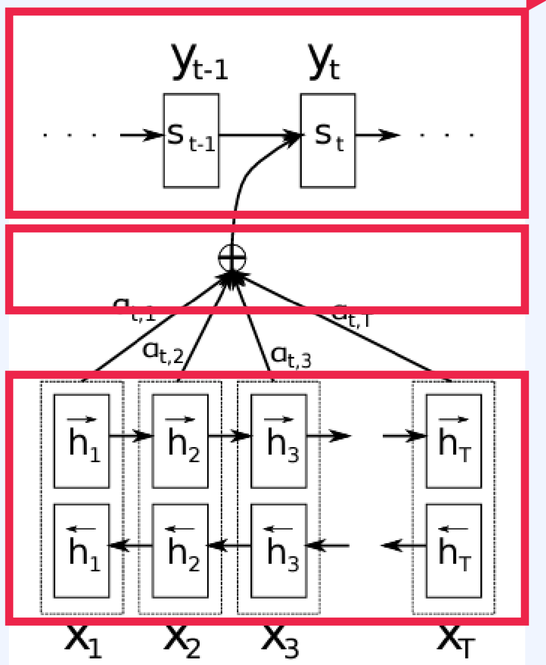


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

Attention

➢ Attention Score

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

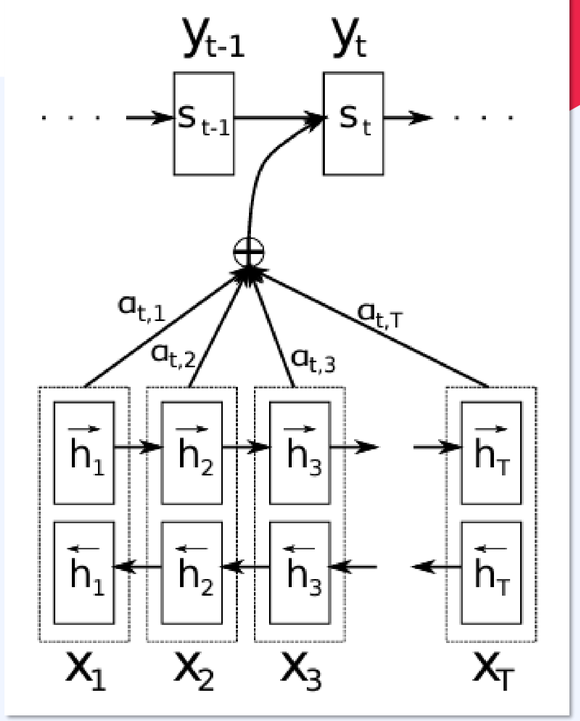
$$e_{ij} = a(s_{i-1}, h_j)$$

디코더의 hidden state

$$s_i = f(s_{i-1}, y_{i-1}, c_i)$$

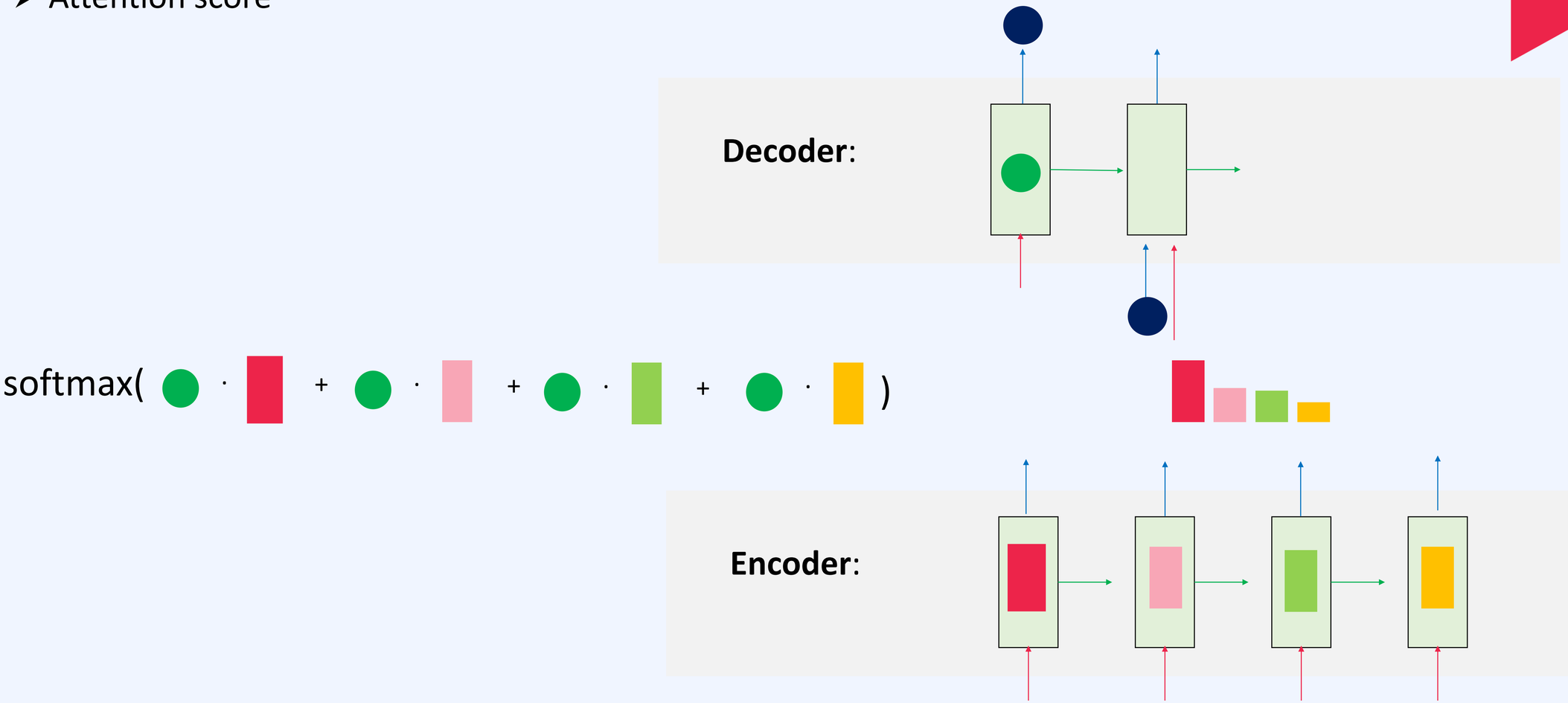
인코더의 hidden state

$$h_t = f(x_t, h_{t-1})$$



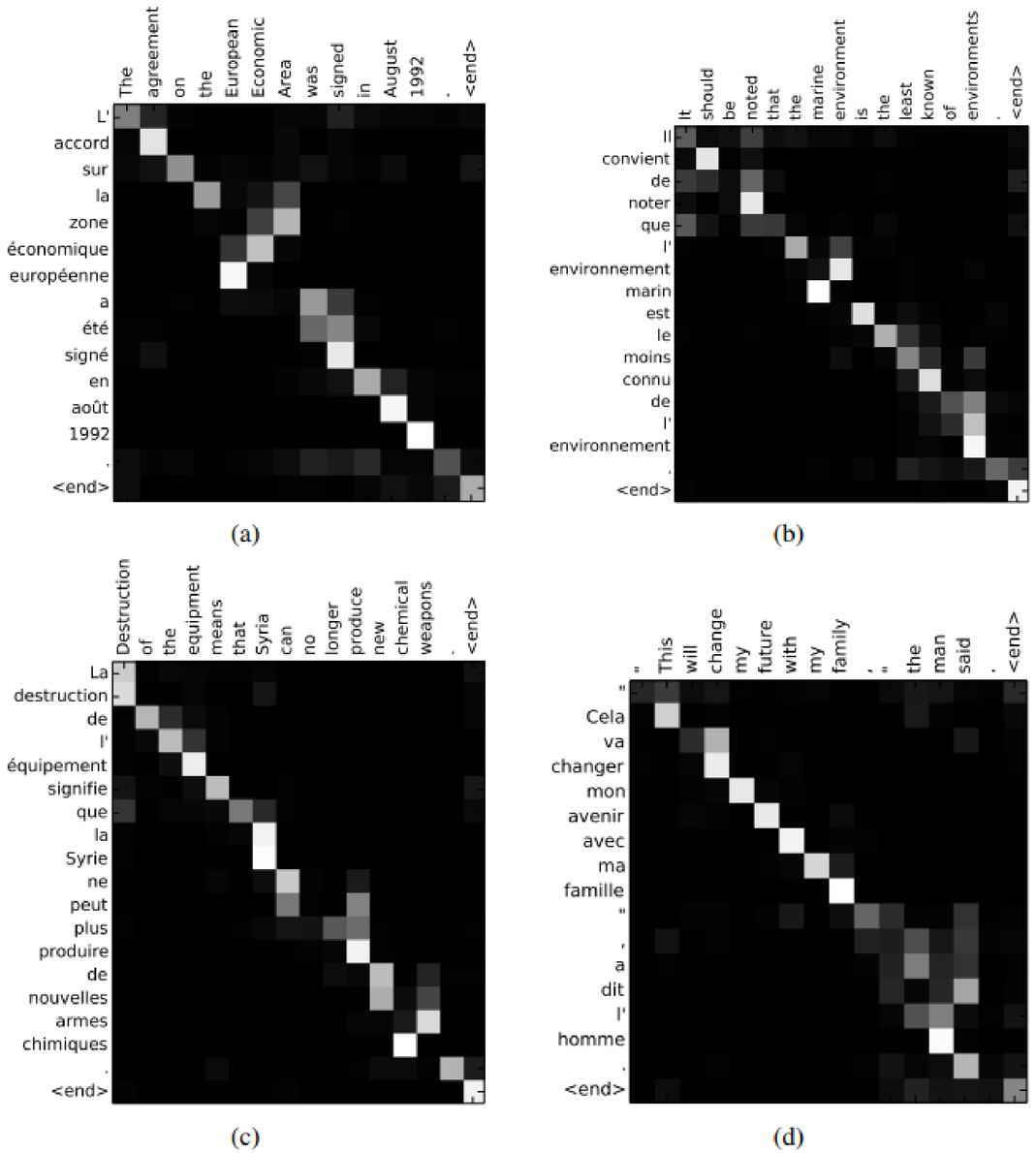
Attention

➢ Attention score



Attention

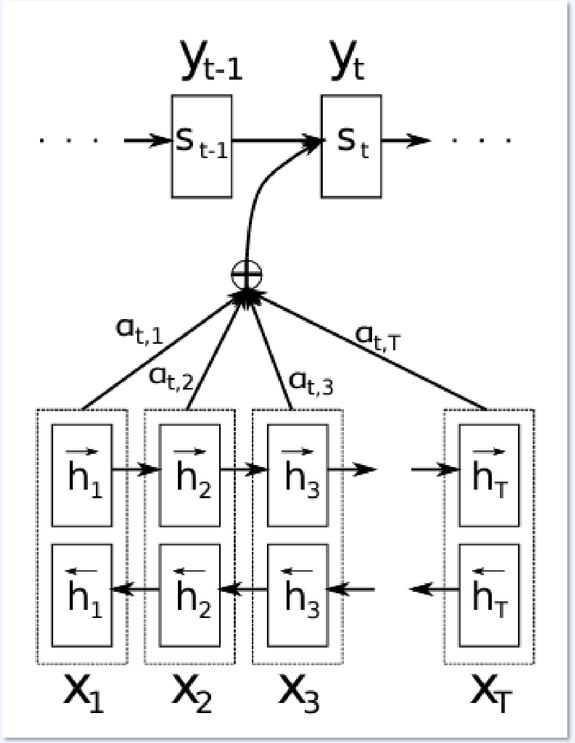
- Machine Translation
  - English - French



Summary

- Attention
  - seq2seq에서 context vector가 fix된 것을 보완
  - 타겟 단어와 중요한 관계를 가지는 부분을 자동(동적)으로 찾음

- Attention Score
  - h의 s에 대한 중요도



Quiz

Q 인코더-디코더 기반의 언어모델 중 하나로, 쿼리와 비슷한 값을 가진 키를 찾아서 그 값을 얻는 과정을 수행하는 구조는?

① Attention

② FastText

③ Word2Vec

④ Seq2Seq

해설

▶ Attention은 쿼리와 비슷한 킷값을 찾아서 비슷한 정도에 따라 weight를 정하고, 각 key의 value값을 weight만큼 가져와서 모두 더하는 연산을 수행하는 언어모델이다.

NLP

Transformer and so on…

Transformer

➢ Paper

- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin, Attention is all you need,

- RNN의 근본적인 한계: Long-term dependency, Gradient Vanishing

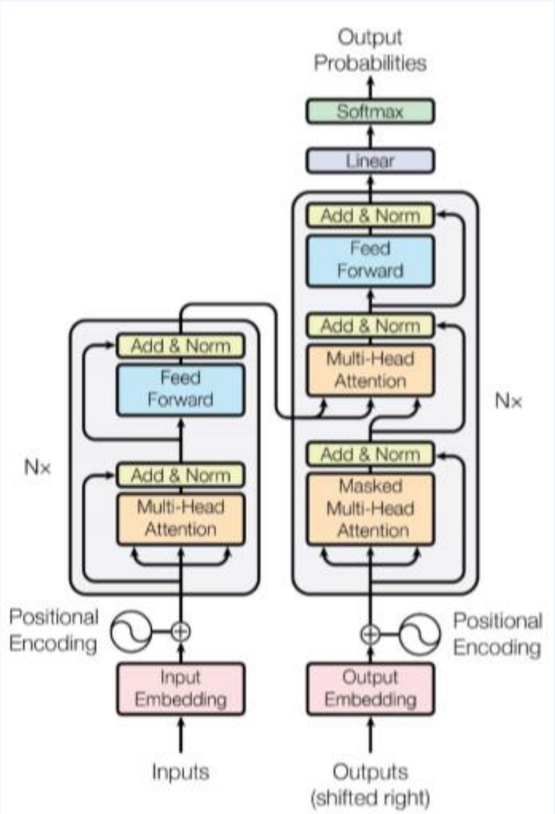


Figure 1: The Transformer - model architecture.

Transformer

- Model Architecture
  - Encoder and Decoder stacks
  - Attention
    - Scaled dot-product attention
    - Multi-head attention
  - Position wise feed forward networks
  - Embeddings and Softmax
  - Positional Encoding

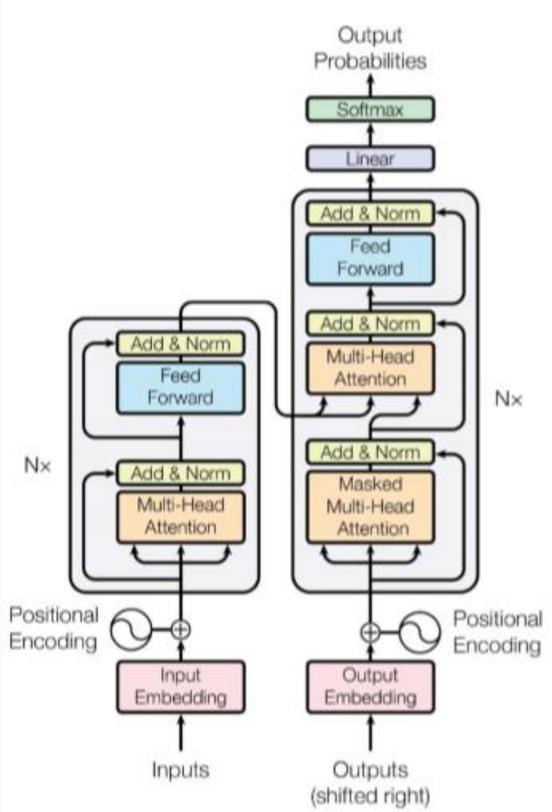


Figure 1: The Transformer - model architecture.

Transformer

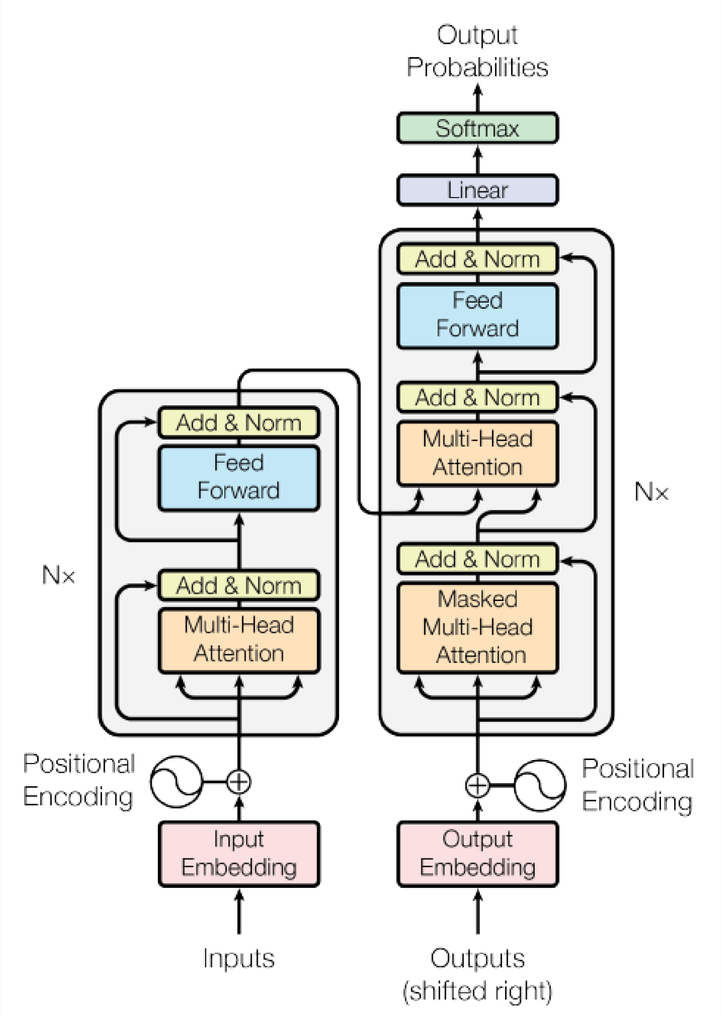
➢ Model Architecture
- **Encoder and Decoder stacks**
- Attention
  - Scaled dot-product attention
  - Multi-head attention
- Position wise feed forward networks
- Embeddings and Softmax
- Positional Encoding

- **Encoder와 Decoder를 각각 6개 사용**
- **Encoder**
  - 두 파트로 구성(Multi-head self attention, Position wise fully connected feed forward network)
  - 각 파트에 residual connection 사용
  - 모든 파트의 Output dimension은 512

- **Decoder**
  - 6개의 독립적인 구조
  - 세 파트로 구성(Masked Multi-Head Attention, Multi-head self attention, Position wise fully connected feed forward network)
  - 각 파트에 residual connection 사용
  - 모든 파트의 Output dimension은 512

Transformer

➢ Model Architecture



- **Encoder와 Decoder를 각각 6개 사용**
- **Encoder**
  - 두 파트로 구성(Multi-head self attention, Position wise fully connected feed forward network)
  - 각 파트에 residual connection 사용
  - 모든 파트의 Output dimension은 512

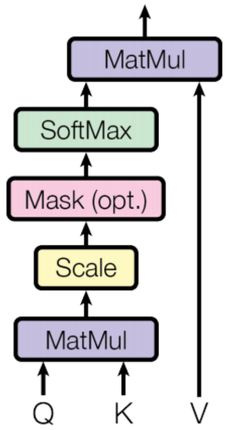
- **Decoder**
  - 6개의 독립적인 구조
  - 세 파트로 구성(Masked Multi-Head Attention, Multi-head self attention, Position wise fully connected feed forward network)
  - 각 파트에 residual connection 사용
  - 모든 파트의 Output dimension은 512

Transformer

➢ Model Architecture
- Encoder and Decoder stacks
- Attention
  - **Scaled dot-product attention**
  - Multi-head attention
- Position wise feed forward networks
- Embeddings and Softmax
- Positional Encoding

- **dot-product attention**
  - Attention score(q, k) = q · k
- **scaled dot-product attention**
  - Attention score(q, k) = q · k / $\sqrt{n}$

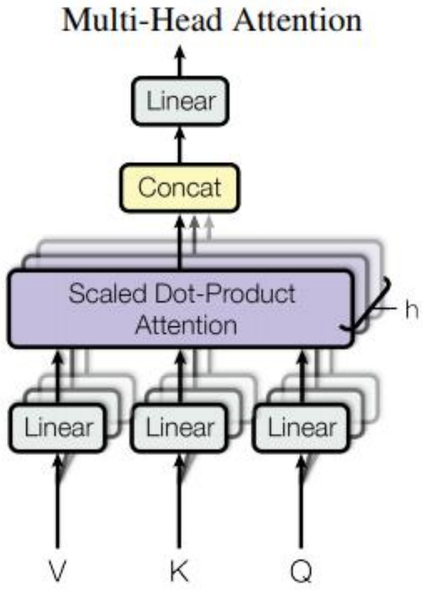
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



Transformer

➢ Model Architecture
- Encoder and Decoder stacks
- Attention
  - Scaled dot-product attention
  - **Multi-head attention**
- Position wise feed forward networks
- Embeddings and Softmax
- Positional Encoding

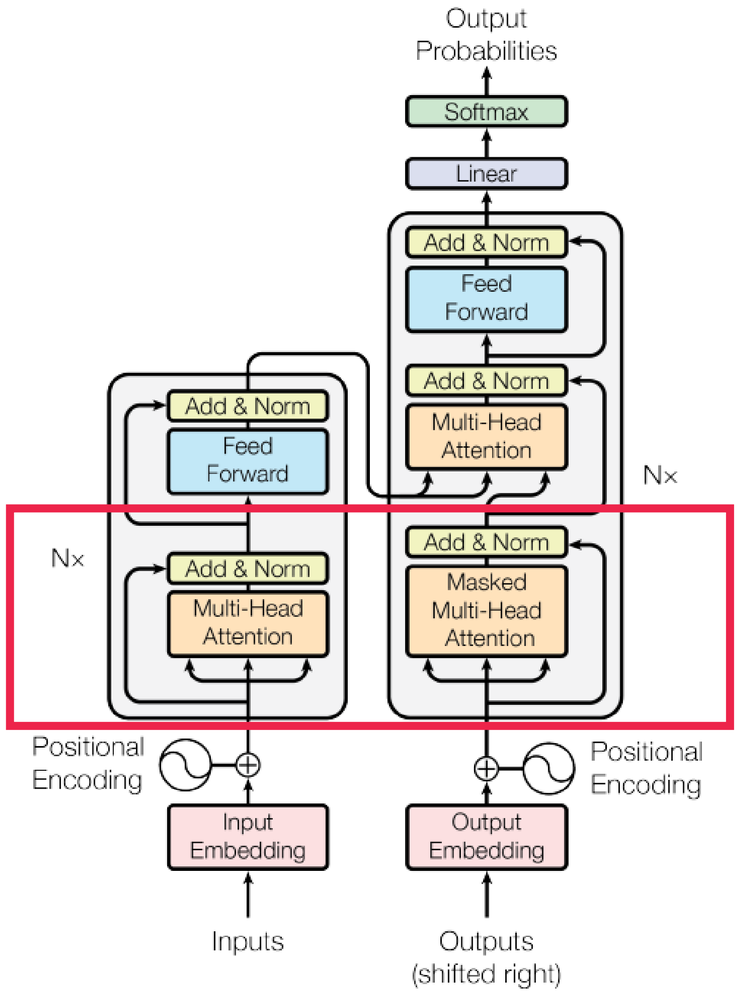
- **Attention을 병렬적으로 수행**



Transformer

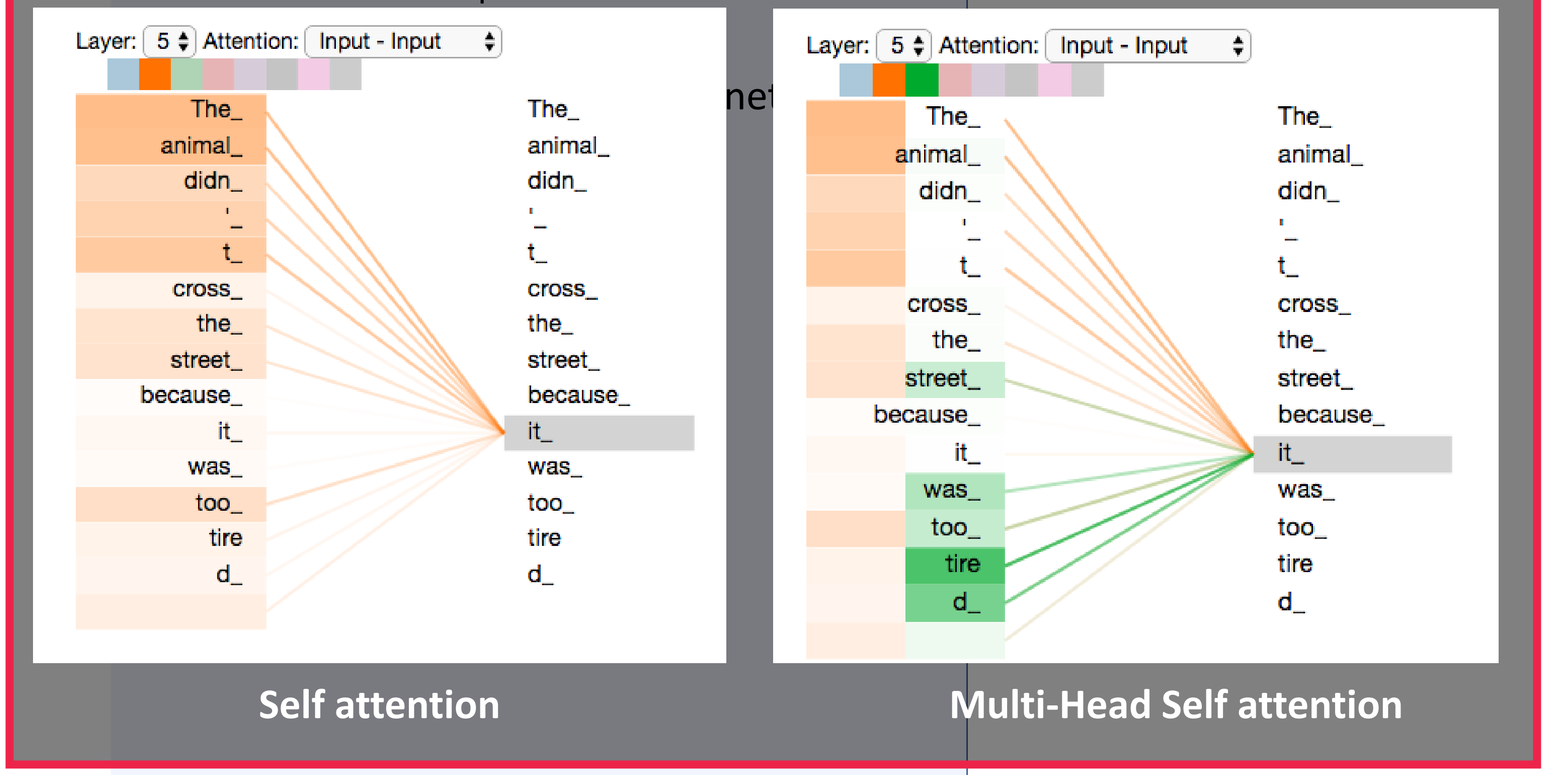
- Model Architecture
  - Encoder and Decoder stacks
  - **Attention**
    - Scaled dot-product attention
    - Multi-head attention
  - Position wise feed forward networks
  - Embeddings and Softmax
  - Positional Encoding

- **Self attention**

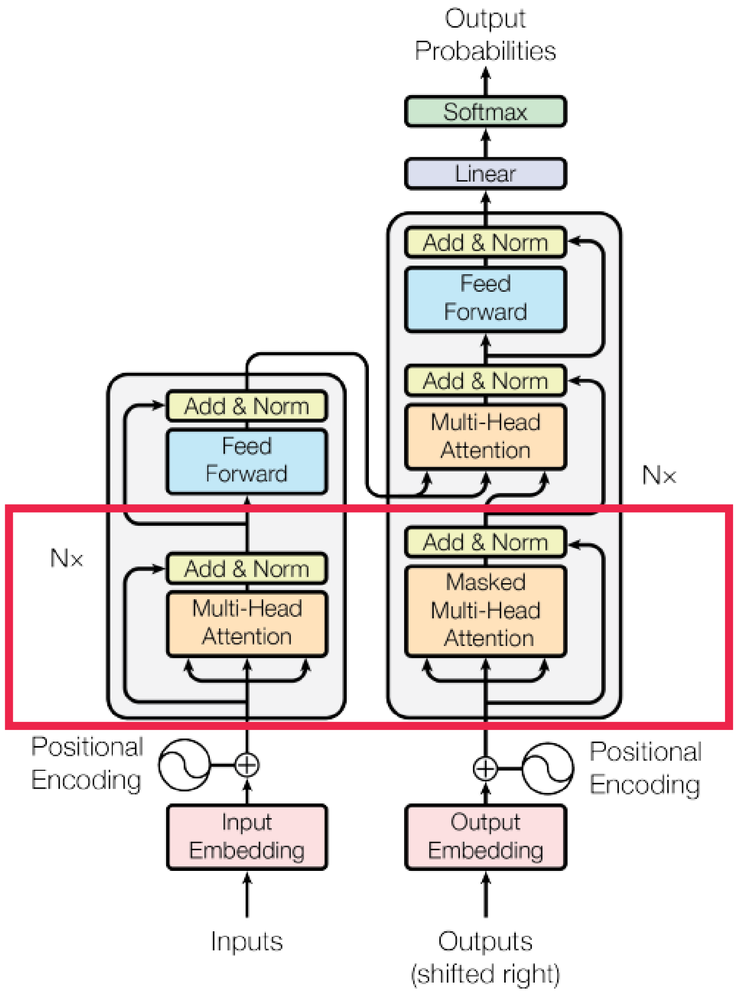


Transformer

- Model Architecture
  - Encoder and Decoder stacks
  - **Attention**
    - Scaled dot-product attention



- **Self attention**

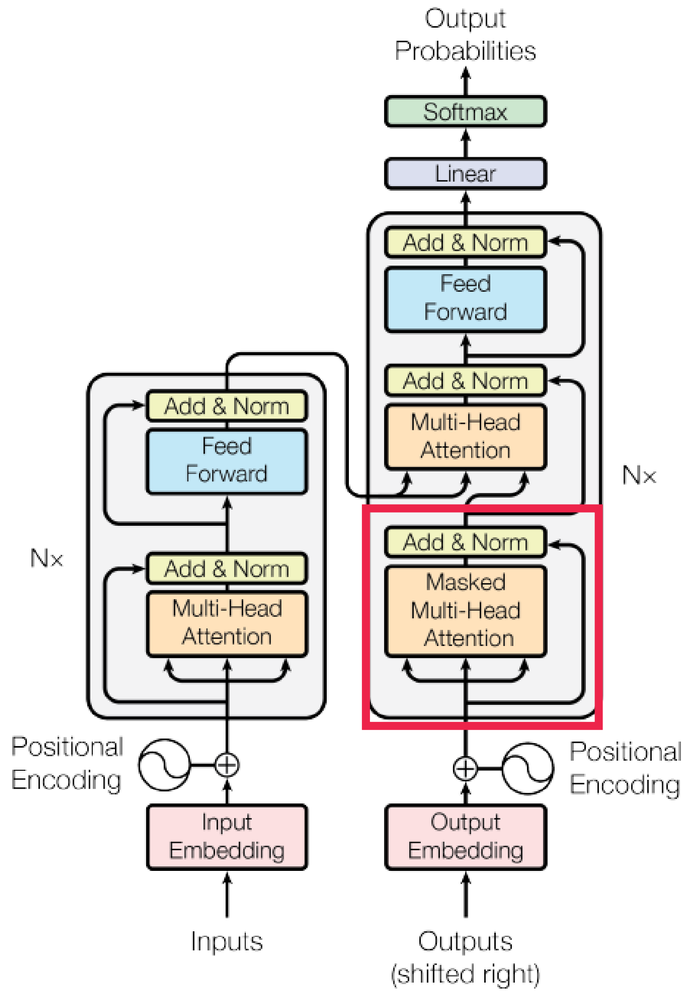


https://www.kaggle.com/code/residentmario/transformer-architecture-self-attention/notebook

Transformer

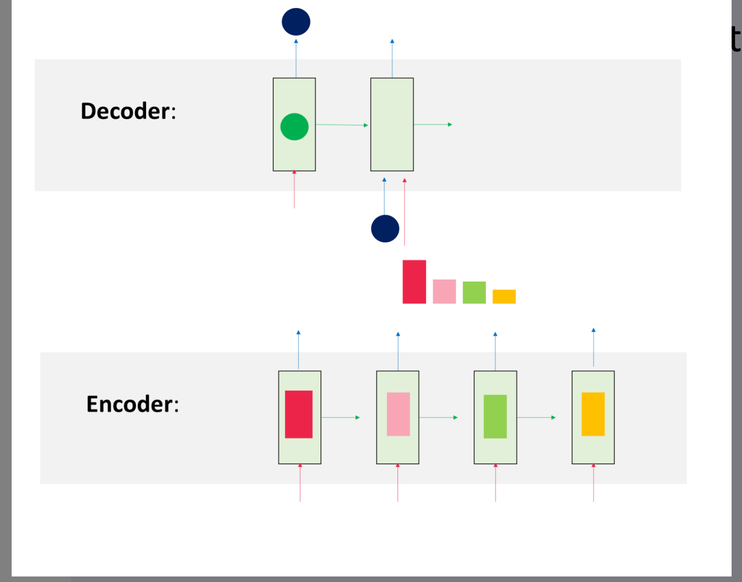
- Model Architecture
  - Encoder and Decoder stacks
  - **Attention**
    - Scaled dot-product attention
    - Multi-head attention
  - Position wise feed forward networks
  - Embeddings and Softmax
  - Positional Encoding

- **Masked Self attention**

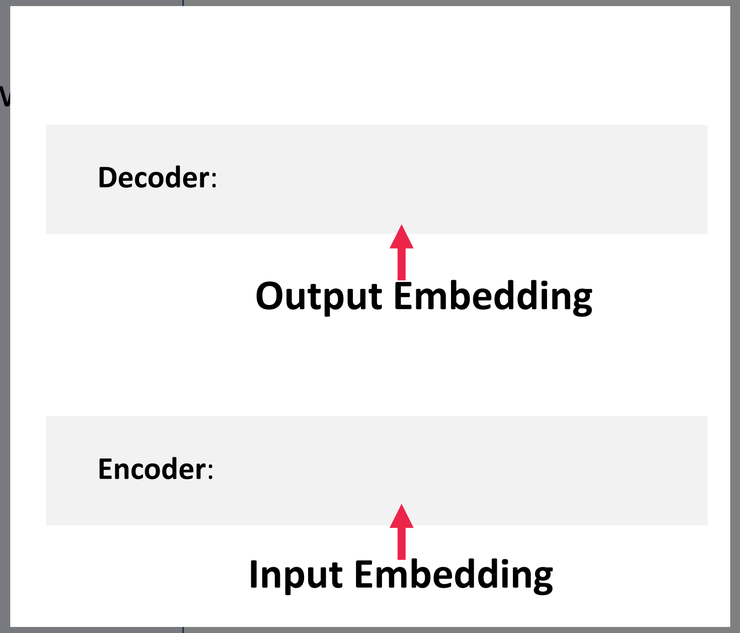


Transformer

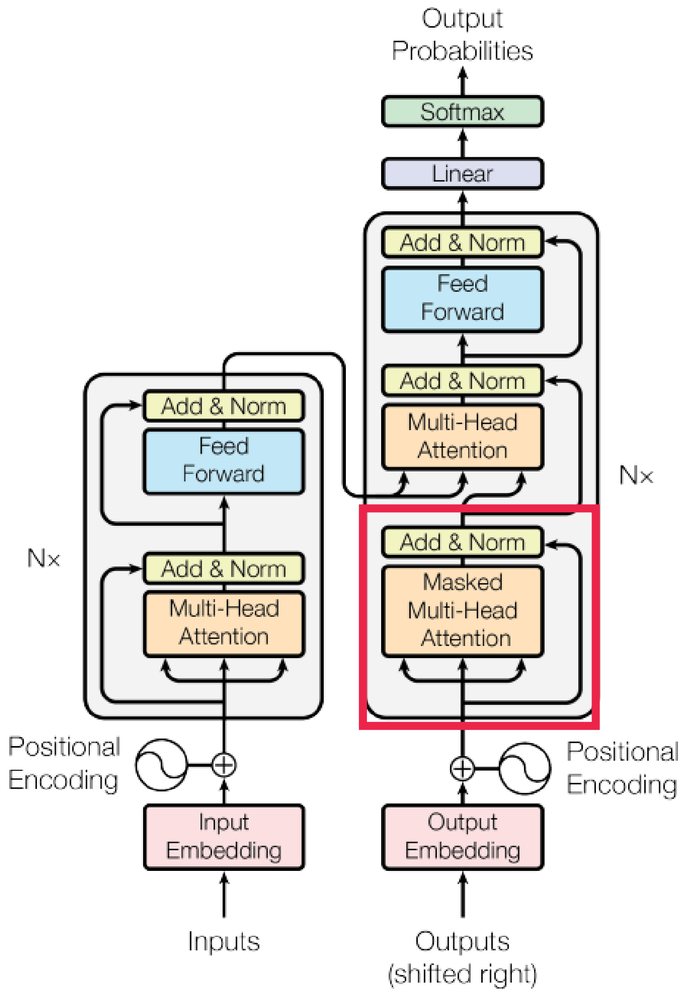
- Model Architecture
  - Encoder and Decoder stacks
  - ~~Attention~~
    - Scaled dot-product attention
    - **Masked Self attention**



Attention



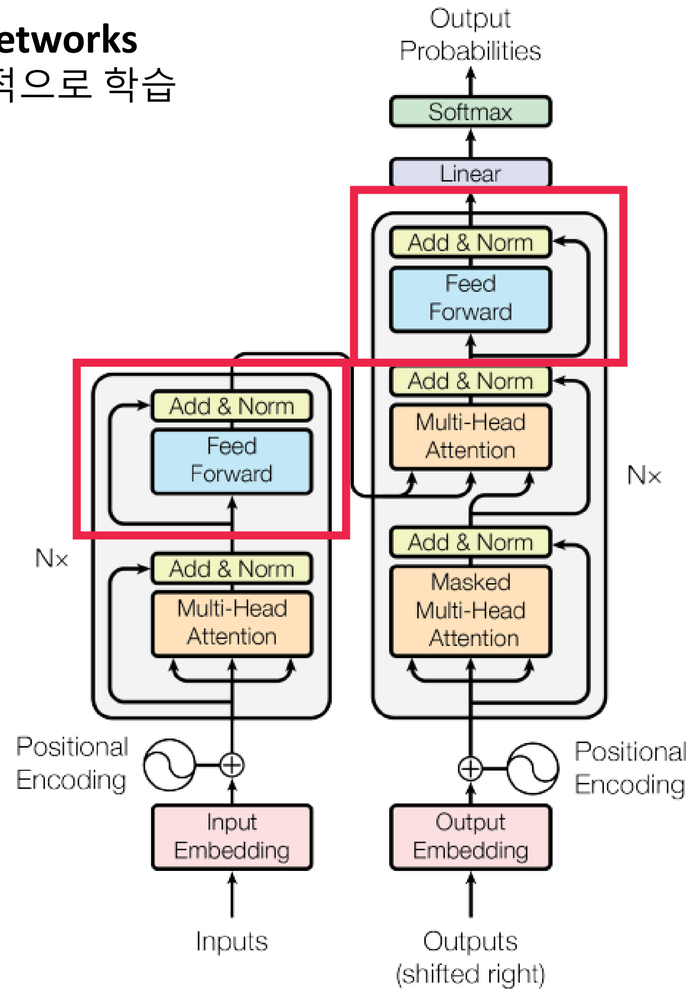
Transformer



Transformer

- Model Architecture
  - Encoder and Decoder stacks
  - Attention
    - Scaled dot-product attention
    - Multi-head attention
  - **Position wise feed forward networks**
  - Embeddings and Softmax
  - Positional Encoding

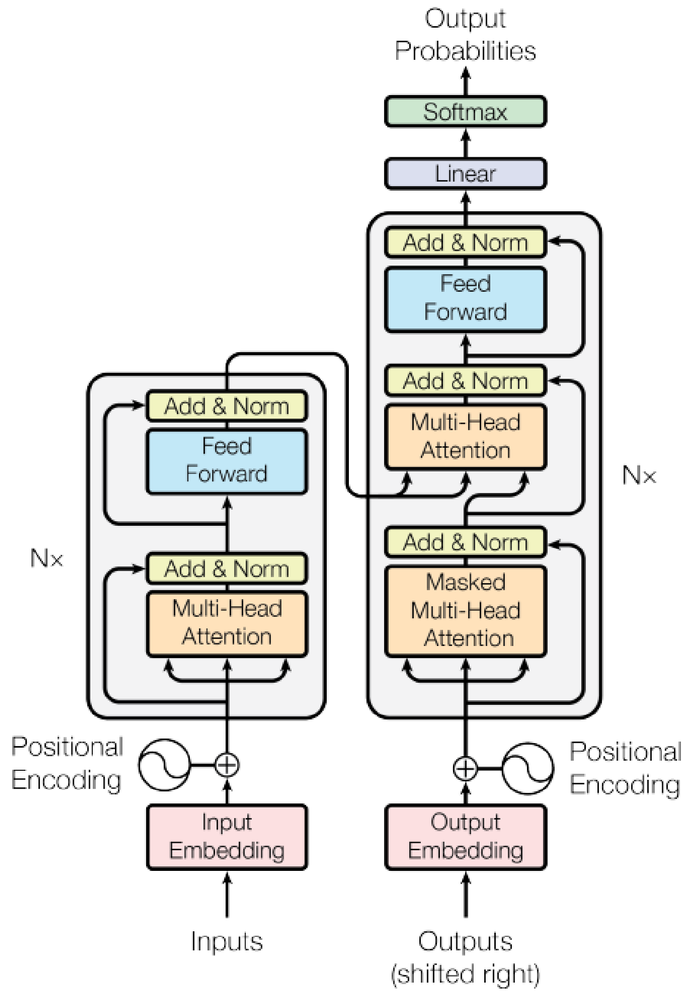
- **Position wise feed forward networks**
  - 문장의 각 위치를 서로 독립적으로 학습



Transformer

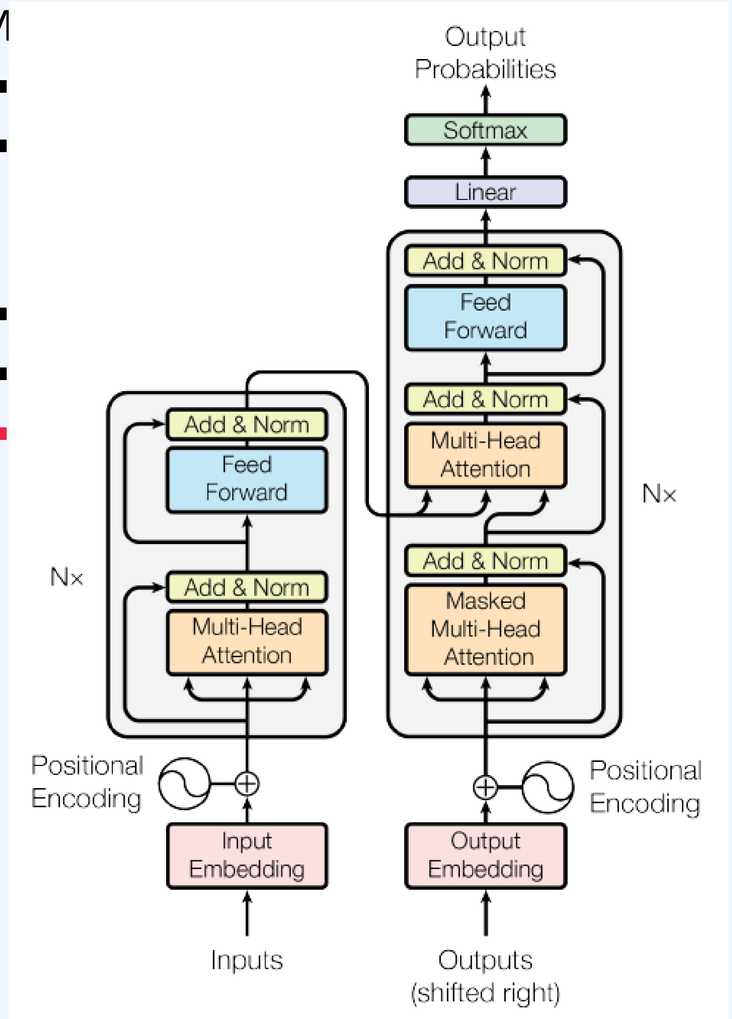
- Model Architecture
  - Encoder and Decoder stacks
  - Attention
    - Scaled dot-product attention
    - Multi-head attention
  - Position wise feed forward networks
  - **Embeddings and Softmax**
  - Positional Encoding

- **Embeddings and Softmax**



Transformer

➢ M



- RNN을 제거하였지만 단어의 위치 정보를 전달 할 수 있는 방법이 필요
- 입력 단어의 임베딩 벡터에 위치 정보를 더하여 사용
  - d = 512, (n = 10000)

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Positional Encoding Matrix with d=4, n=100

Sequence	Index of token, k	i=0	i=0	i=1	i=1
I	0	P_{00}=sin(0) = 0	P_{01}=cos(0) = 1	P_{02}=sin(0) = 0	P_{03}=cos(0) = 1
am	1	P_{10}=sin(1/1) = 0.84	P_{11}=cos(1/1) = 0.54	P_{12}=sin(1/10) = 0.10	P_{13}=cos(1/10) = 1.0
a	2	P_{20}=sin(2/1) = 0.91	P_{21}=cos(2/1) = -0.42	P_{22}=sin(2/10) = 0.20	P_{23}=cos(2/10) = 0.98
Robot	3	P_{30}=sin(3/1) = 0.14	P_{31}=cos(3/1) = -0.99	P_{32}=sin(3/10) = 0.30	P_{33}=cos(3/10) = 0.96

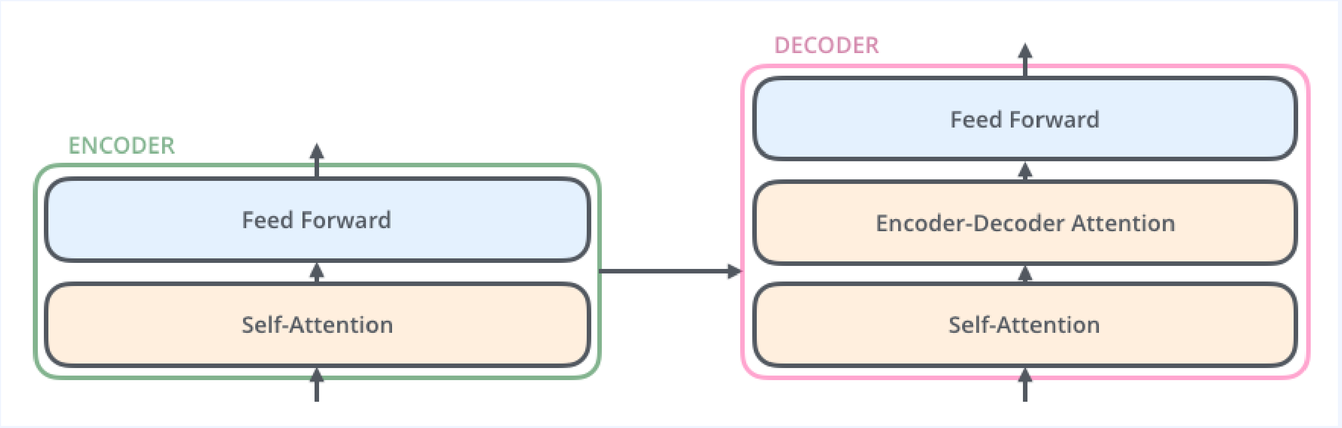
Positional Encoding Matrix for the sequence 'I am a robot'

https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1/

Summary – Transformer

- RNN을 encoder와 decoder에서 제거
  - RNN의 근본적인 한계: Long-term dependency, Gradient Vanishing
- Positional Encoding
- Attention
  - Encoder: Self-Attention(입력 문장을 잘 표현)
  - Decoder: Self + Masked (입력 문장을 잘 표현, 미래의 데이터는 숨기기)
  - Encoder – Decoder Attention

  - Scaled-dot production
  - Multi-head



https://www.kaggle.com/code/residentmario/transformer-architecture-self-attention/notebook

BERT

- BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Jacob Devlin, Ming-Wei Chang , Kenton Lee, Kristina Toutanova
  - bi-directional Transformer로 이루어진 언어모델
  - 잘 만들어진 BERT 언어모델 위에 1개의 classification layer만 부착하여 다양한 NLP task를 수행
  - 영어권에서 11개의 NLP task에 대해 SOTA(state of the art) 달성

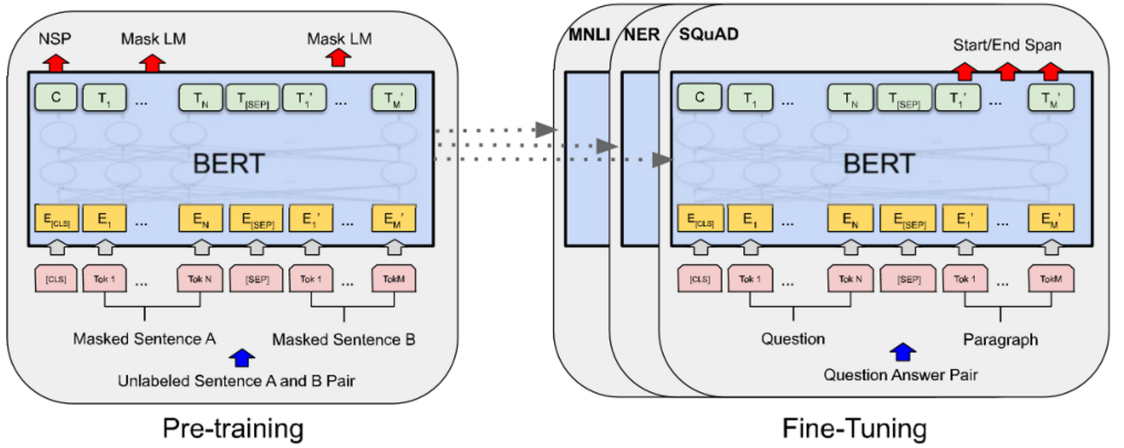
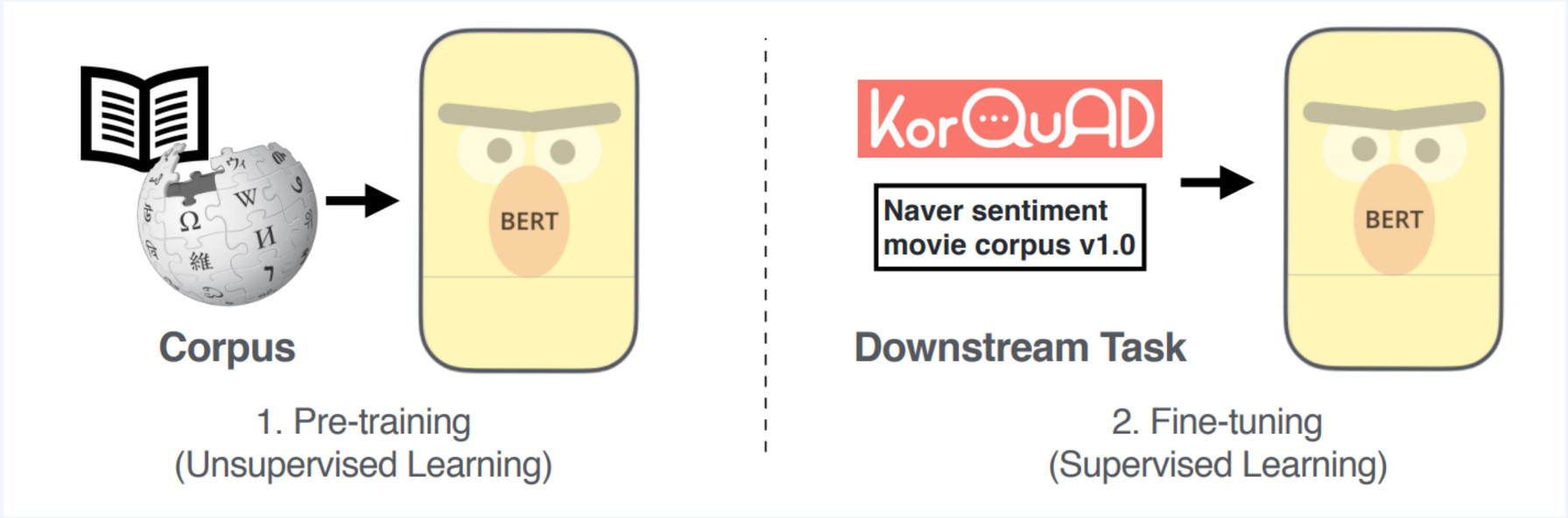


Figure 1: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. [CLS] is a special symbol added in front of every input example, and [SEP] is a special separator token (e.g. separating questions/answers).

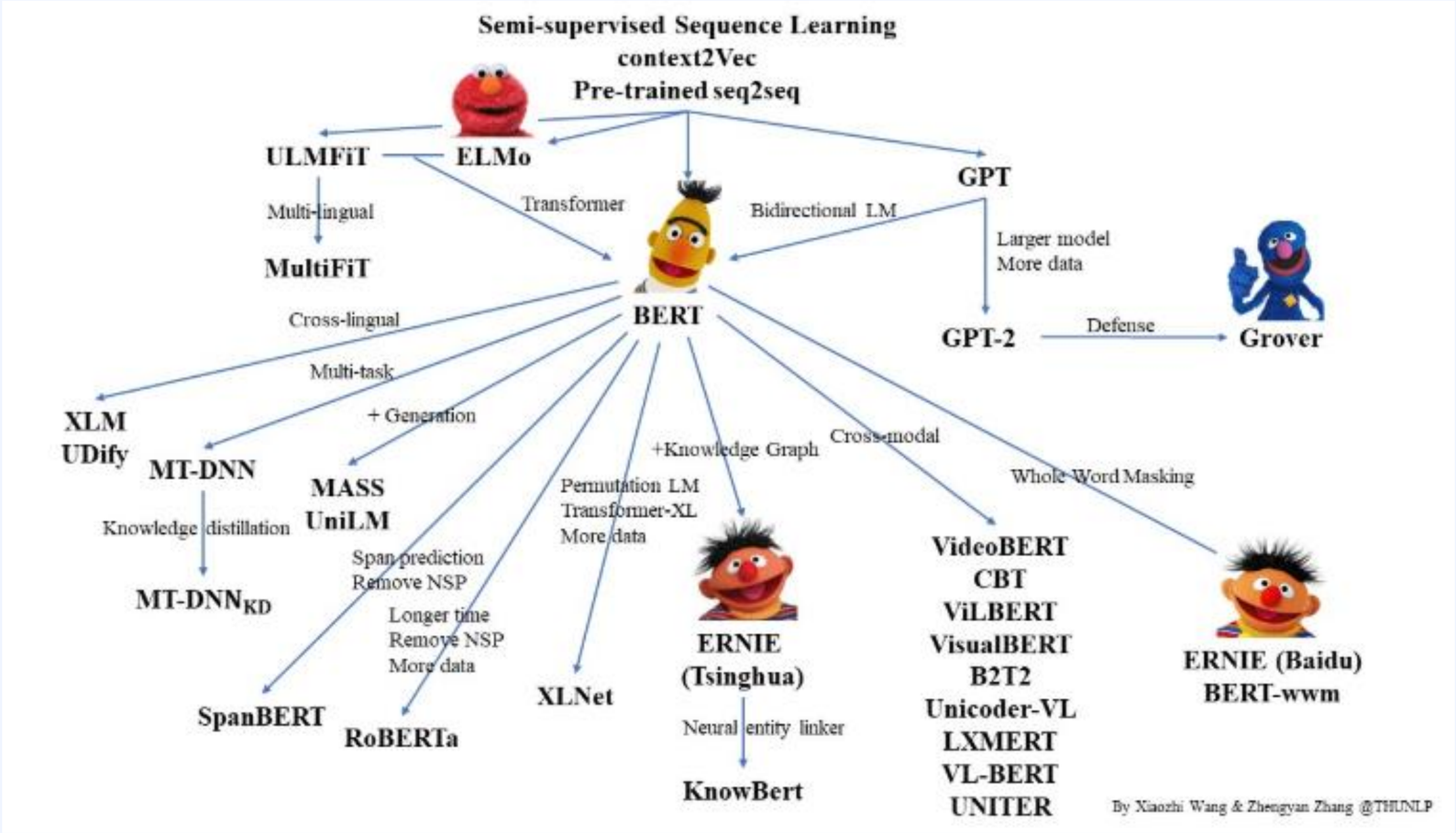
PLM

- Pre-training then fine-tuning
- 대량의 코퍼스를 Pre-training한 다음, 각 Task에 따라서 fine-tuning을 진행하는 모델



PLM

➢ BERT를 시작으로 계속해서 다양해짐

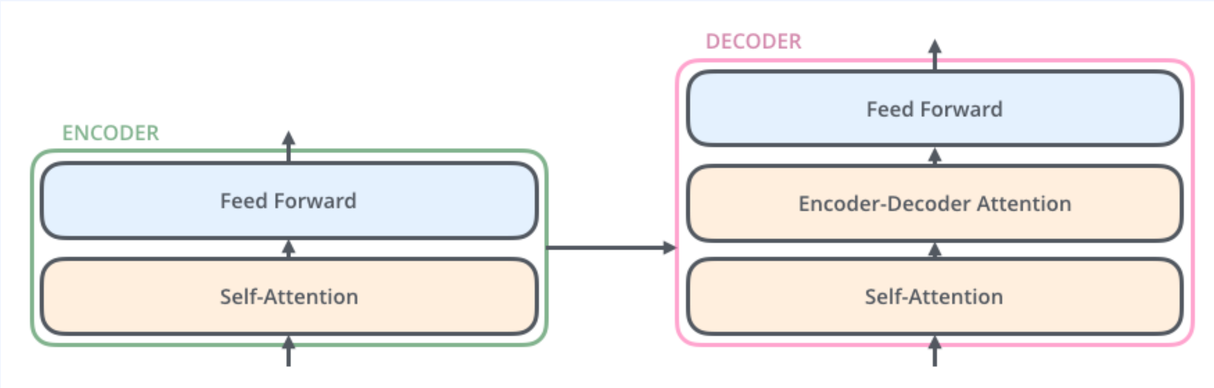


https://github.com/thunlp/PLMpapers

Summary

➢ Transformer는 RNN을 Long-term dependency, Gradient Vanishing 근본적인 한계로 지적하고, RNN은 제외하고 Attention 구조만 사용(Attention is all you need!)한 모델이다.

- Positional Encoding
- Position wise Feed forward
- Attention
  - Encoder
    - Multi head + Self
  - Decoder
    - Multi head + Self + Masked
    - Multi head + Encoder – Decoder Attention



➢ BERT는 대량의 코퍼스를 Pre-training한 다음, 각 Task에 따라서 fine-tuning을 진행하는 모델이다.