Computational Linguistics

Spring 2023

108.413A (002)

Course Information

- Instructor: Sangah Lee (Dept. of Linguistics, Seoul National University)
  (sanalee@snu.ac.kr)
- TA: Minji Kang (mnjkng@snu.ac.kr)
- Lecture: Tue, Thu 11:00-12:15 (bldg. 14, room 208)
- Textbooks: slides and supplementary materials provided
  + Jurafsky and Martin (2023 draft), “Speech and Language Processing”
  https://web.stanford.edu/~jurafsky/slp3/
- If you haven’t taken the pre-requisite course [Language and Computer], make sure that you should be familiar with Python and language modeling: some traditional topics including n-grams, Naive Bayes, and Logistic Regression.

Objectives

- Introduction to fundamental notions and theories on CL and NLP
  - current issues on deep learning models and the Transformer mechanism
  - large-scale language models based on them

- Development of programming and research abilities
  - understanding deep learning models and transformer mechanism
  - dealing with issues of CL and NLP
  - using Python-based tools (e.g. PyTorch)

Requirements

- Grade Policies: Relative Grading (A-F)

Item	Attendance	Assignment	Midterm	Final	Total
Rate (%)	10	30	30	30	100

- If you have a valid reason for absence, please submit:
  the relevant documents and the Attendance Acknowledgment Request Form (uploaded on the eTL page)
- Midterm: paper test
- Final: project presentation
- Assignments: Python programming exercises, a taken-home quiz
- Students majoring in engineering could get separate grades.

Syllabus

Week	Date	Topic
0	3/2 (Thu)	Course Introduction
1	3/7, 3/9	Basics of Text Processing and PyTorch
2	3/14, 3/16	PyTorch (Neural Networks)
3	3/21, 3/23	PyTorch (Neural Networks)
4	3/28, 3/30	Statistical Language Model (N-gram)
5	4/4, 4/6	N-gram and Entropy
6	4/11, 4/13	Text Classification
7	4/18, 4/20	Word Embedding
8	4/25, 4/27	Midterm Exam

Syllabus

Week	Date	Topic
9	5/2, 5/4	Sequence-to-Sequence Model (Encoder-Decoder)
10	5/9, 5/11	Attention
11	5/16, 5/18	Transformer
12	5/23, 5/25	Transformer-based Pre-trained Models
13	5/30, 6/1	HuggingFace Transformer
14	6/8(Thu), 6/13(Tue)	Various NLP Tasks based on Transformer
15	6/15 (Thu)	Final Project Presentations

Environment Settings

- Make sure that you have environments for Python programming:
  - Google Colab https://colab.research.google.com/
  - Jupyter Notebook https://jupyter.org/

- Assignment codes will be basically based on .ipynb.
  - .py forms are also allowed: but the paths should be correct!

Supplementaries

- Take a look over the Python libraries below!
  - NLTK (Natural Language Toolkit) https://www.nltk.org/
  - SpaCy https://spacy.io/
  - textacy https://textacy.readthedocs.io/en/latest/
  - csv https://docs.python.org/3/library/csv.html
  - json https://docs.python.org/3/library/json.html

- And a Python data structure “Classes” (Object Oriented Programming) https://docs.python.org/3/tutorial/classes.html

Computational Linguistics

- A subfield of linguistics and computer science
  - concerned with the interaction of human language and computers
- Includes:
  - the analysis of written texts and spoken discourse
  - the translation of text and speech from one language into another
  - the use of human languages for communication between computers and people
  - the modeling and testing of linguistic theories
- Statistical analysis of written texts and spoken discourse
  - analysis on corpus: relative frequencies or collocation

    of letters, sounds, morphemes, words, …

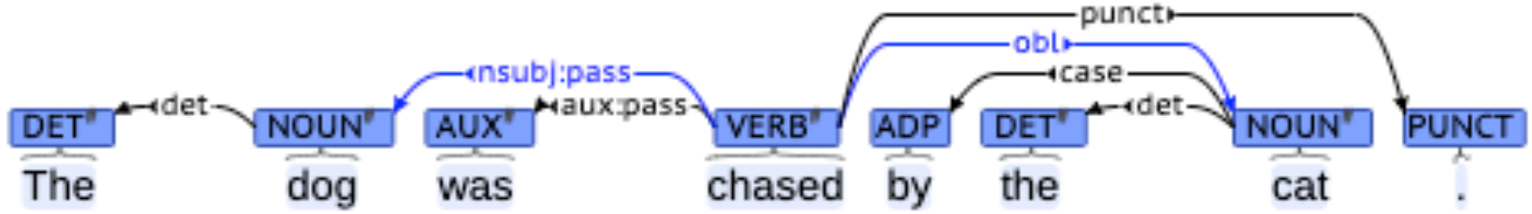
Fromkin et al. (2018). "An Introduction to Language". Cengage Learning Australia.

https://medium.com/dair-ai/building-your-first-wordcloud-with-google-colaboratory-and-python-b3c8c1be5883

Computational Linguistics

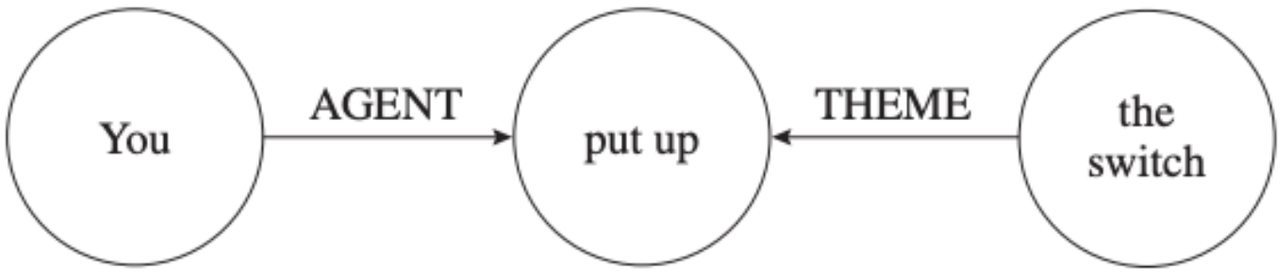
- The interaction between language and computers in all dimensions
  - Computational Phonetics and Phonology

    - Speech Recognition, Speech Synthesis (Text-to-Speech)
  - Computational Morphology: processing of word structures
  - Computational Syntax



https://universaldependencies.org/

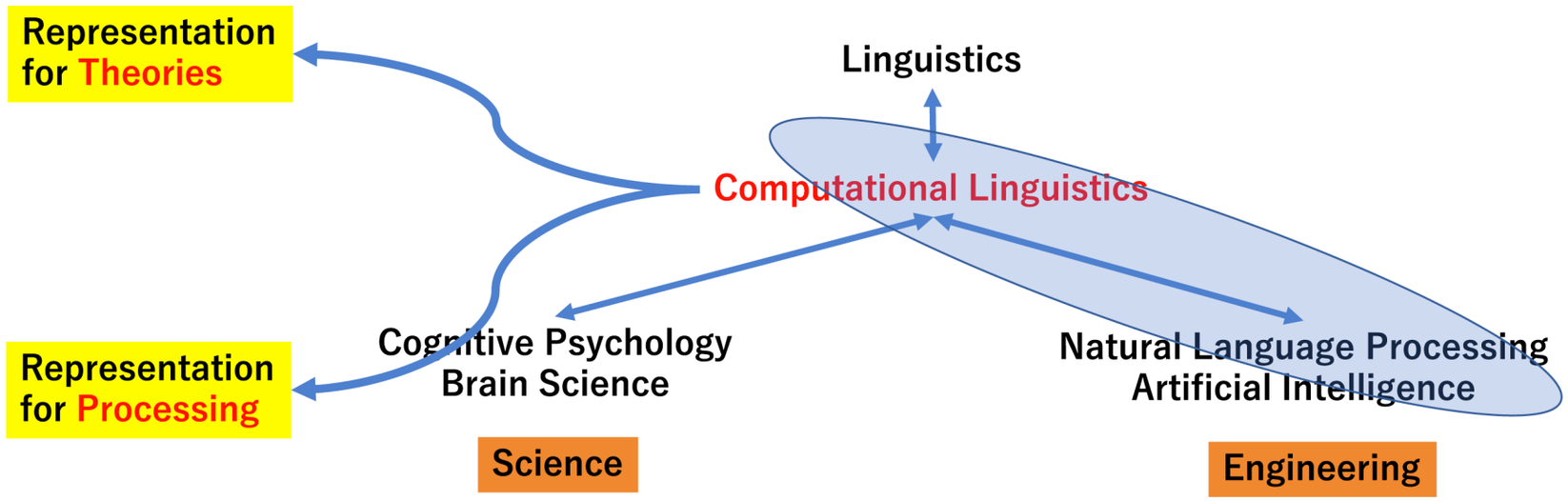
- Computational Semantics: speech understanding and generation



- Computational Pragmatics: sentence disambiguation, coreference resolution, …

Natural Language Processing

- A schematic view of research disciplines: CL and NLP



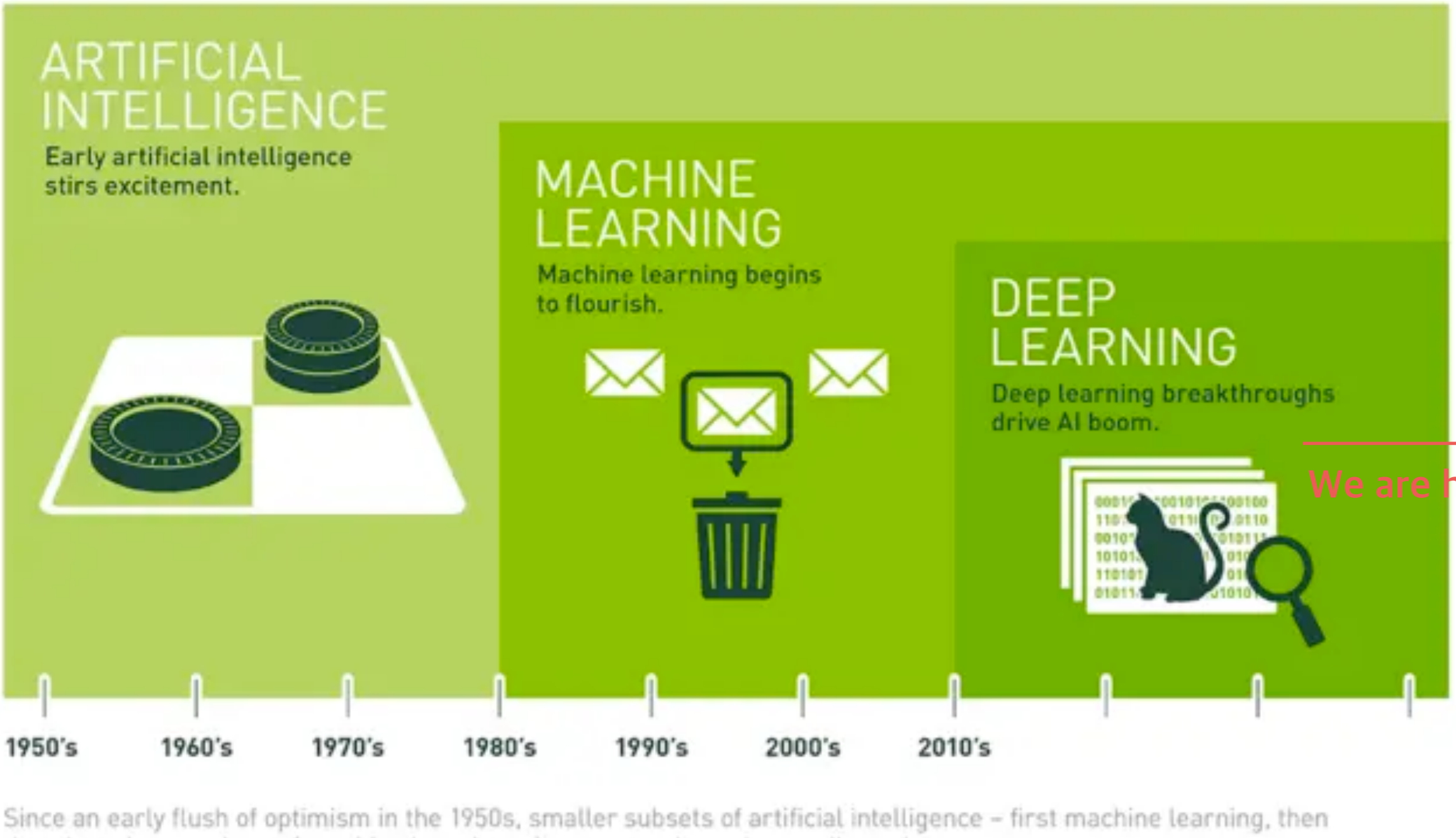
how language is processed in our minds or our brains

how computer systems should be designed to process language efficiently and effectively

- NLP may be included either in CL and other fields as their subfield.
  (Any other fields can deal with and utilize language data.)
- And this kind of research can be performed through…

Tsujii (2021). "Natural Language Processing and Computational Linguistics". Computational Linguistics 2021. 47 (4): pp. 707-727.

Deep Learning?

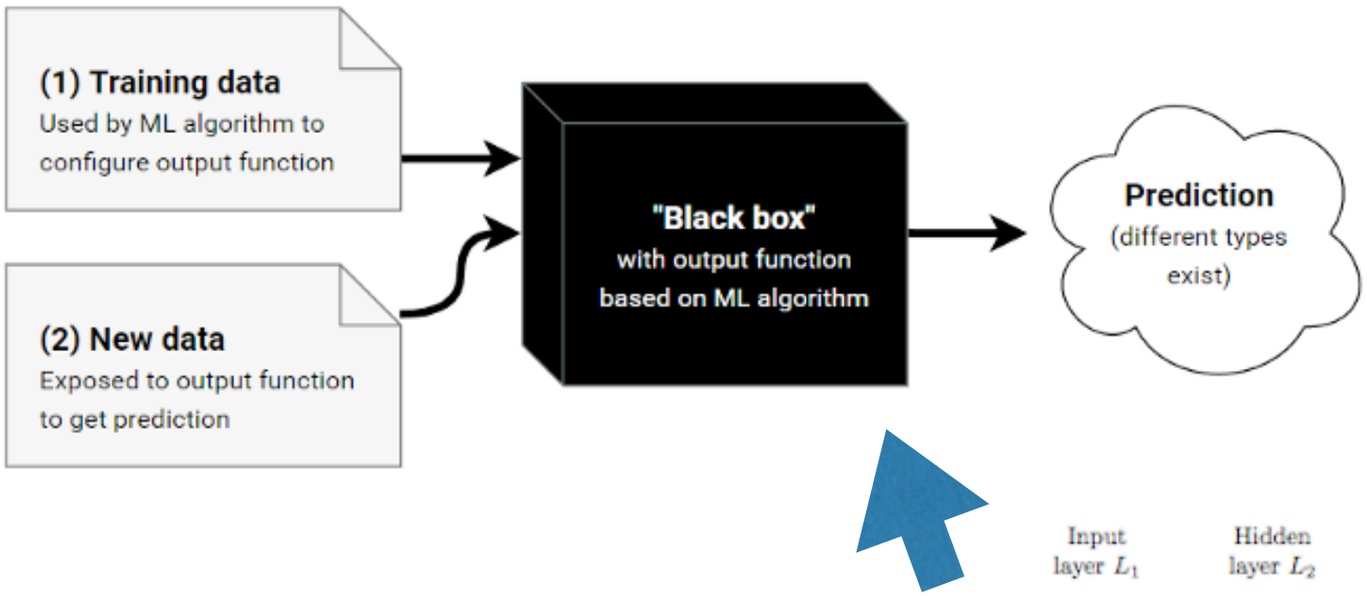


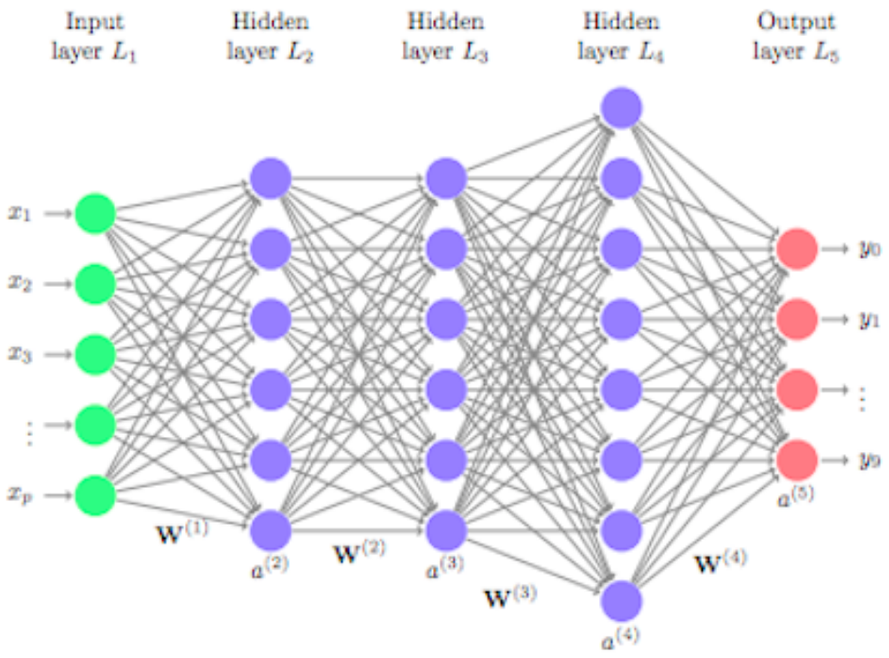
Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/

Deep Learning

- 데이터를 입력받고 그 안의 패턴과 정보를 분석하는 모델이 심층 구조로 이루어진 것





뉴런들의 상호작용을 모사한 것

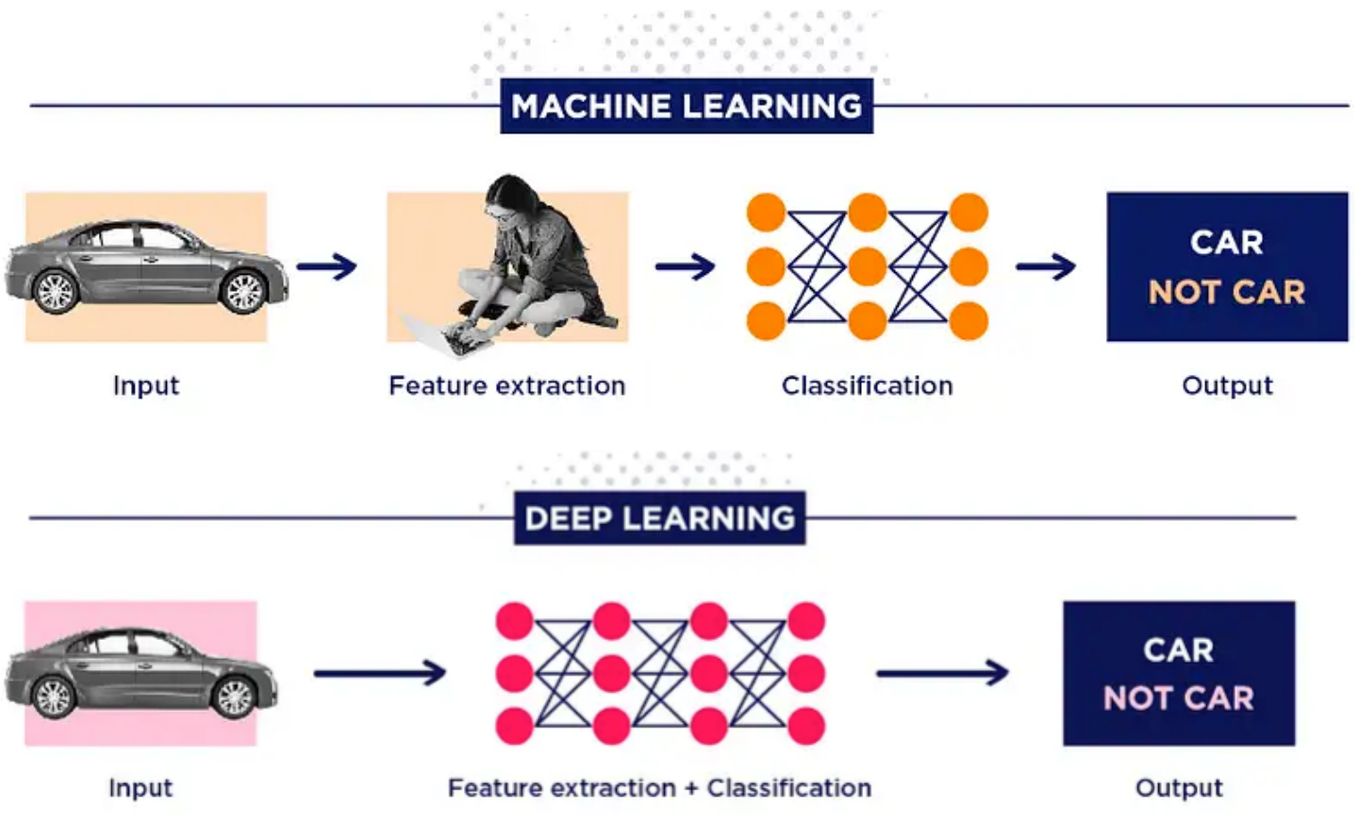
=> 더 복잡한, 추상적인 연산 기대

https://towardsdatascience.com/its-deep-learning-times-a-new-frontier-of-data-a1e9ef9fe9a8

Deep Learning

- No/Less hand-crafted features required
- 기계학습 모델과 달리 사람이 직접 자질 구성을 하지 않는다.

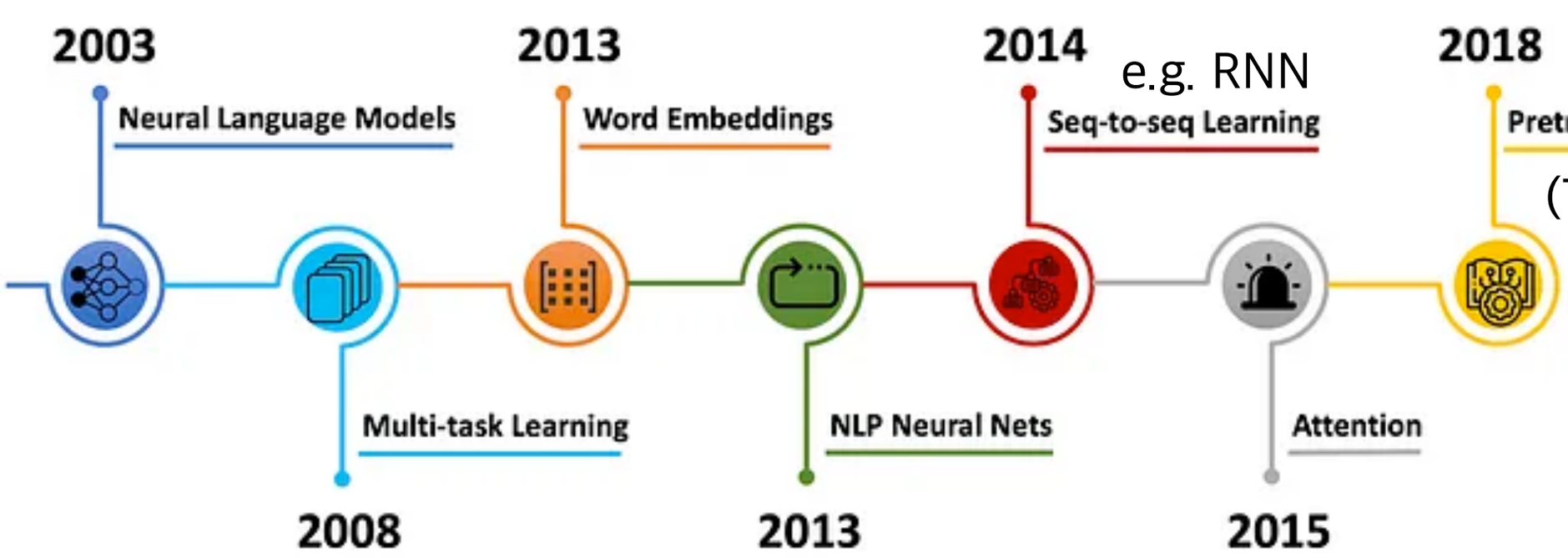
  심층학습 구조 안에서 모델이 데이터로부터 스스로 자질을 구성한다.



https://www.iteratorshq.com/blog/machine-learning-vs-deep-learning-the-ultimate-comparison/

Deep Learning

- In NLP:
  - Neural models automatically learn low-dimensional continuous vectors from data as task-specific features.
  - capturing semantic meanings of words, phrases, and sentences, …
  - CNN, RNN, LSTM, … models for various NLP tasks



(better context processing)

Han et al. (2021). "Pre-Trained Models: Past, Present and Future". arXiv:2106.07319.
https://medium.com/@antoine.louis/a-brief-history-of-natural-language-processing-part-2-f5e575e8e37

PyTorch

PyTorch
https://pytorch.org/

- An open-source library for machine learning and deep learning applications
  - can use GPU accelerators -> faster!
  - 기본 자료 구조: Tensor (Numpy의 array와 유사, 호환)
  - 여러 기계학습 모델, 딥러닝 모델, 학습과 검증 등에 필요한 요소 등이 구현되어 있음

```
from torch.utils.data import DataLoader

train_dataloader = DataLoader(training_data, batch_size=64, shuffle=True)
test_dataloader = DataLoader(test_data, batch_size=64, shuffle=True)
```

데이터 로드, 구조화

```
import torch
from torch import nn
```

```
class NeuralNetwork(nn.Module):
    def __init__(self):
        super().__init__()
        self.flatten = nn.Flatten()
        self.linear_relu_stack = nn.Sequential(
            nn.Linear(28*28, 512),
            nn.ReLU(),
            nn.Linear(512, 512),
            nn.ReLU(),
            nn.Linear(512, 10),
        )

    def forward(self, x):
        x = self.flatten(x)
        logits = self.linear_relu_stack(x)
        return logits
```

class를 이용한 모델 정의

```
model = NeuralNetwork()
```

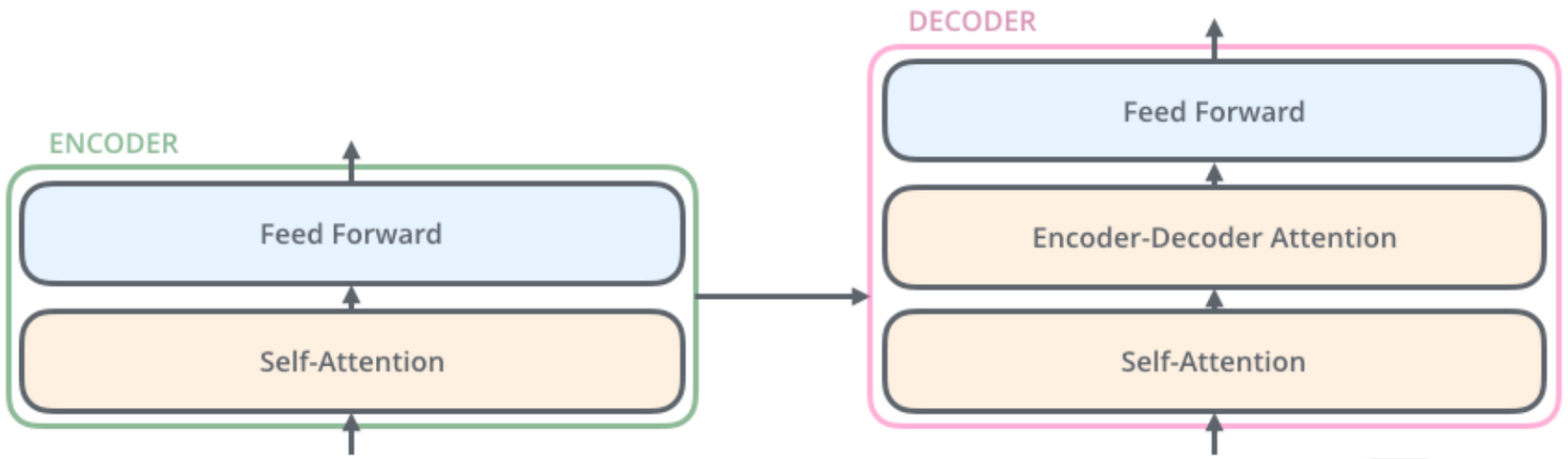
정의한 모델 객체 생성

```
logits = model(X)
```

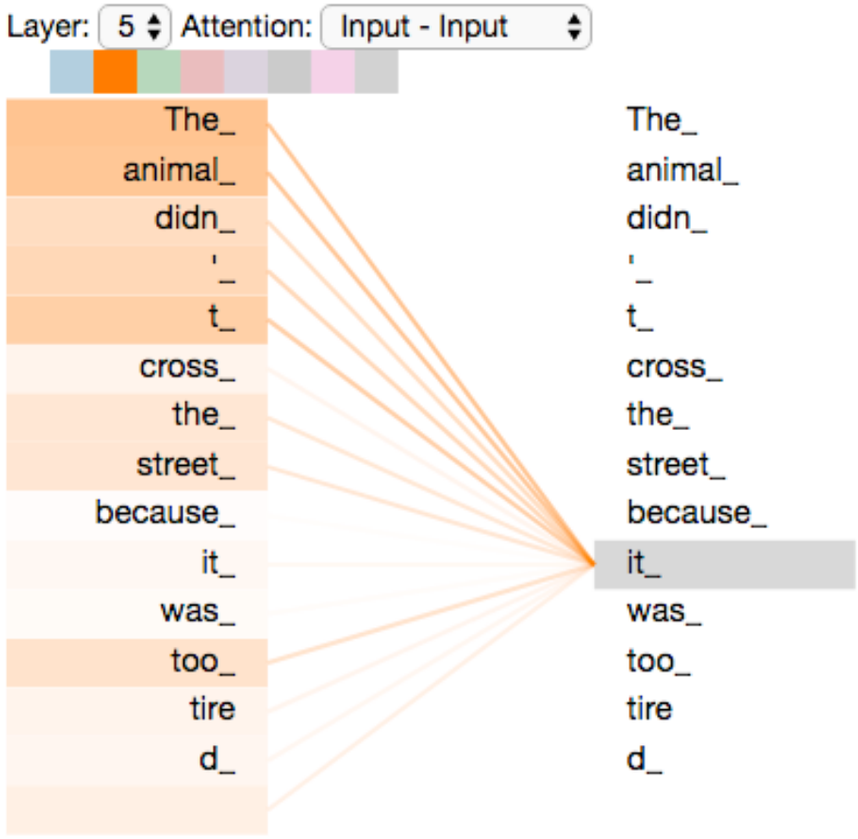
모델에 input 넣고 계산

Transformers

- Using very deep neural models and attention



- Effectively modeling semantic meanings of language
  - e.g. Focusing on relationship between words based on semantic meaning and contexts



https://jalammar.github.io/illustrated-transformer/

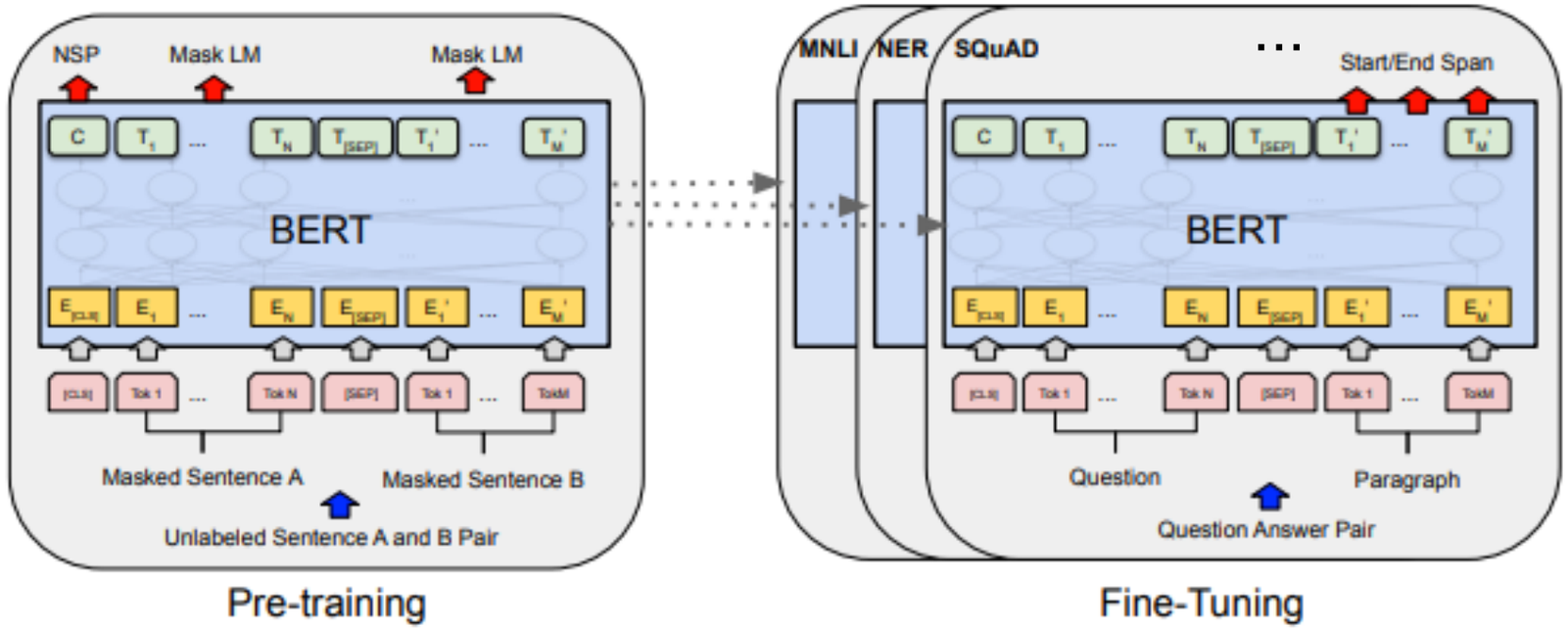
Pre-Trained Models

- Such deep and complex calculations are performed and **freezed** for future use

  ===> Pre-Trained (Language) Models: PTM, PLM (사전학습모델)

  - Pre-trained models: BERT, GPT, …
  - Usually provided through HuggingFace

tasks:
Natural Language Inference
Named Entity Recognition
Question Answering
…



import and load a given BERT model

let the model perform specific task

Devlin et al. (2019). “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). pp. 4171-4186.

And you will like this

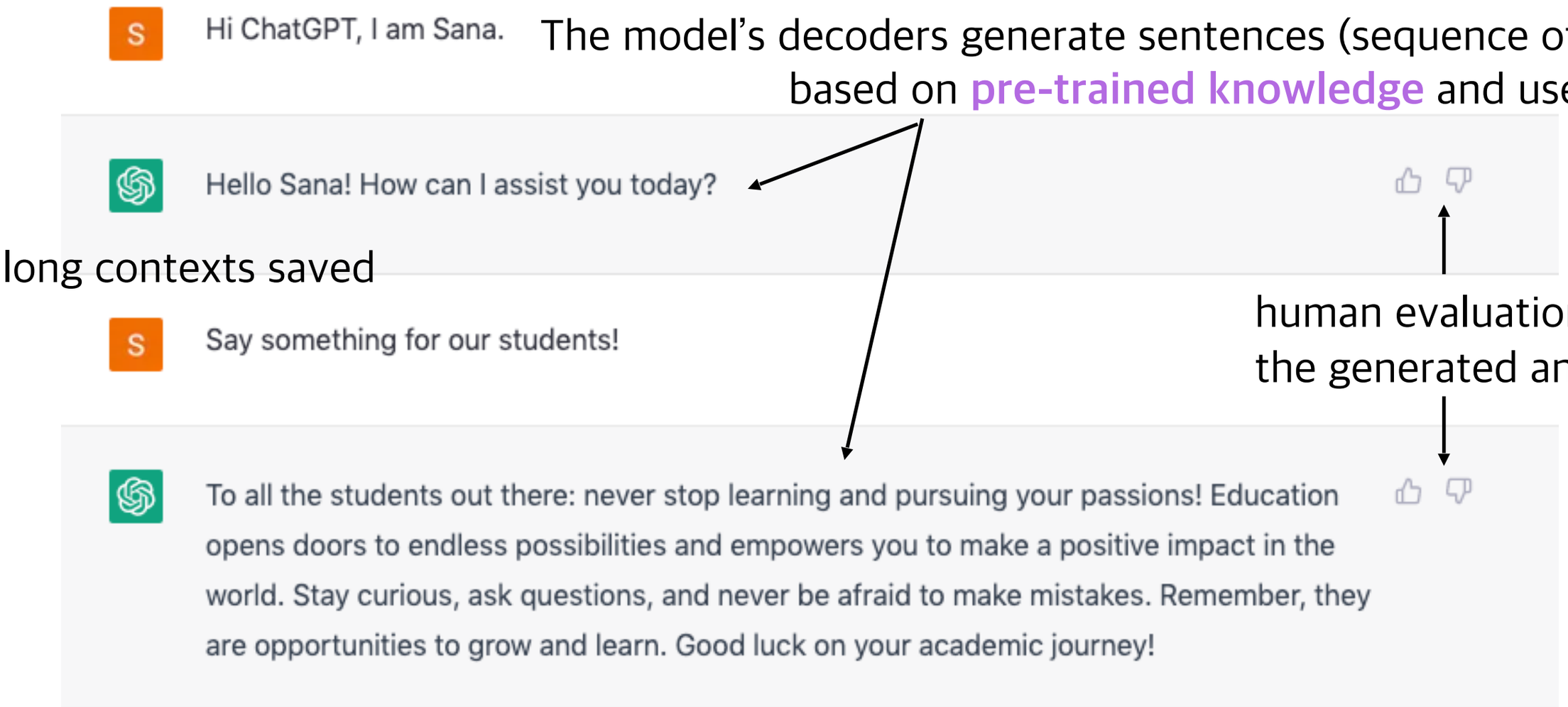
- ChatGPT: one of the PTMs especially working for dialogues

https://openai.com/blog/chatgpt/

The model's decoders generate sentences (sequence of words) based on **pre-trained knowledge** and user inputs

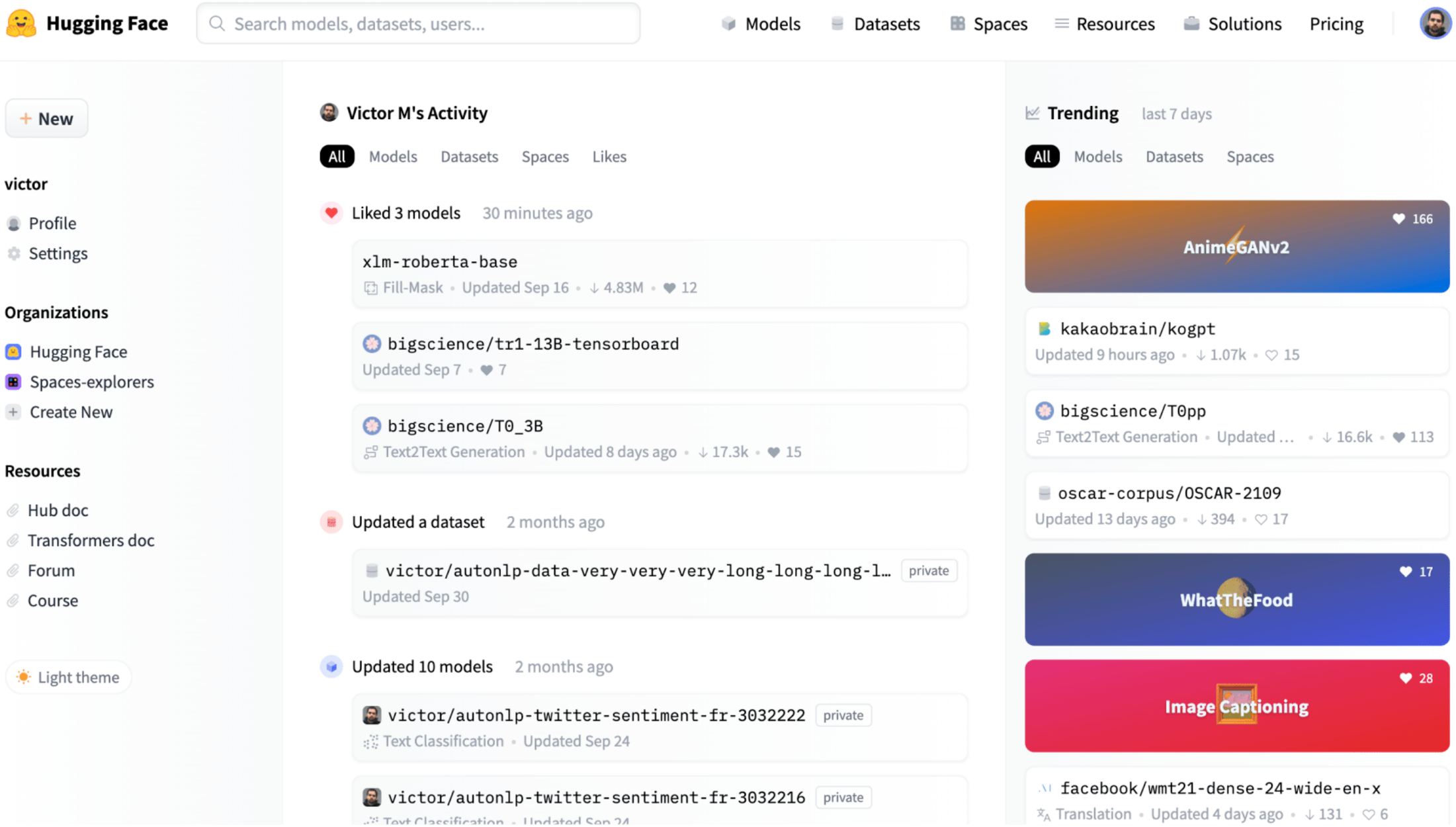
long contexts saved

human evaluation about the generated answers



HuggingFace

- A platform and a library providing various models, datasets and codes



HuggingFace

- A platform and a library providing various models, datasets and codes

snunlp / **KR-BERT-char16424** ♡ like 4

PyTorch | JAX | Transformers | Korean | arxiv:2008.03979 | bert

Model card | Files and versions | Community

KoRean based Bert pre-trained (KR-BERT)

This is a release of Korean-specific, small-scale BERT models with comparable or better performances developed by Computational Linguistics Lab at Seoul National University, referenced in KR-BERT: A Small-Scale Korean-Specific Language Model.

Vocab, Parameters and Data

	Mulitlingual BERT (Google)	KorBERT (ETRI)	KoBERT (SKT)	KR-BERT character	KR-BERT sub-character
vocab size	119,547	30,797	8,002	16,424	12,367
parameter size	167,356,416	109,973,391	92,186,880	99,265,066	96,145,233

- README.md
- config.json
- flax_model.msgpack
- pytorch_model.bin
- tokenizer_config.json
- vocab.txt

HuggingFace

- You can import and use the pre-trained models!

```
from transformers import BertConfig, BertModel, BertForPreTraining, BertTokenizer

tokenizer = BertTokenizer.from_pretrained('snunlp/KR-BERT-char16424', do_lower_case=False)

sentence = '내가 만든 쿠키 너를 위해 구웠지'
tkns = tokenizer.tokenize(sentence)
print(tkns)
```

Sentence Tokenization

```
['내가', '만든', '쿠', '##키', '너', '##를', '위해', '구', '##웠', '##지']
```

```
model = BertModel.from_pretrained('snunlp/KR-BERT-char16424')
inputs = tokenizer(sentence, return_tensors="pt")
output = model(**inputs)
print(output)
```

Sentence Encoding

```
BaseModelOutputWithPoolingAndCrossAttentions(last_hidden_state=tensor([[[ 0.3934, -0.2802, -0.1499,  ..., -0.0381, -0.4774,
         [ 0.6975,  0.1270, -0.7793,  ...,  1.0838, -0.1161, -0.8736],
         [ 0.7390,  0.1325, -0.5397,  ...,  0.5206,  0.2594, -0.8449],
         ...,
         [-0.2334,  0.1795, -0.6226,  ...,  0.2216, -0.3278,  0.3870],
         [-0.3575, -0.1823, -0.7299,  ...,  0.0212,  0.0282, -0.7853],
         [ 0.4552, -1.0330, -0.0385,  ...,  0.1119, -0.7874, -0.1191]]],
       grad_fn=<NativeLayerNormBackward0>), pooler_output=tensor([[ 2.6916e-01, -2.0484e-01,  1.6539e-01,  2.5938e-01,  2.9
          1.7813e-01,  2.4731e-01,  1.4999e-01, -3.3565e-01, -4.1460e-01,
         -1.6348e-01, -6.1204e-01,  3.5326e-01, -4.8477e-01,  1.6353e-01,
         -7.5884e-02, -9.3399e-02, -4.9610e-02,  7.0147e-02,  8.9095e-01,
         -3.7699e-02,  3.6045e-01, -7.8517e-01, -3.9698e-01,  1.5543e-01,
          4.0353e-01, -1.2426e-01,  1.9852e-01,  2.1575e-01, -8.6640e-02,
          4.6102e-02,  3.1457e-01,  8.0098e-04, -2.7198e-01,  2.2101e-01,
         -3.1642e-01,  1.1386e-03,  1.4578e-01, -8.6948e-02,  4.3643e-02,
```

For Next Week

- Article:
  https://medium.com/@ageitgey/natural-language-processing-is-fun-9a0bff37854e
- Library: try PyTorch in your environment!
  https://pytorch.org/