Classroom Activity

Deep Learning

Woohwan Jung

한양대학교
HANYANG UNIVERSITY

DSLAB
데이터사이언스 연구실

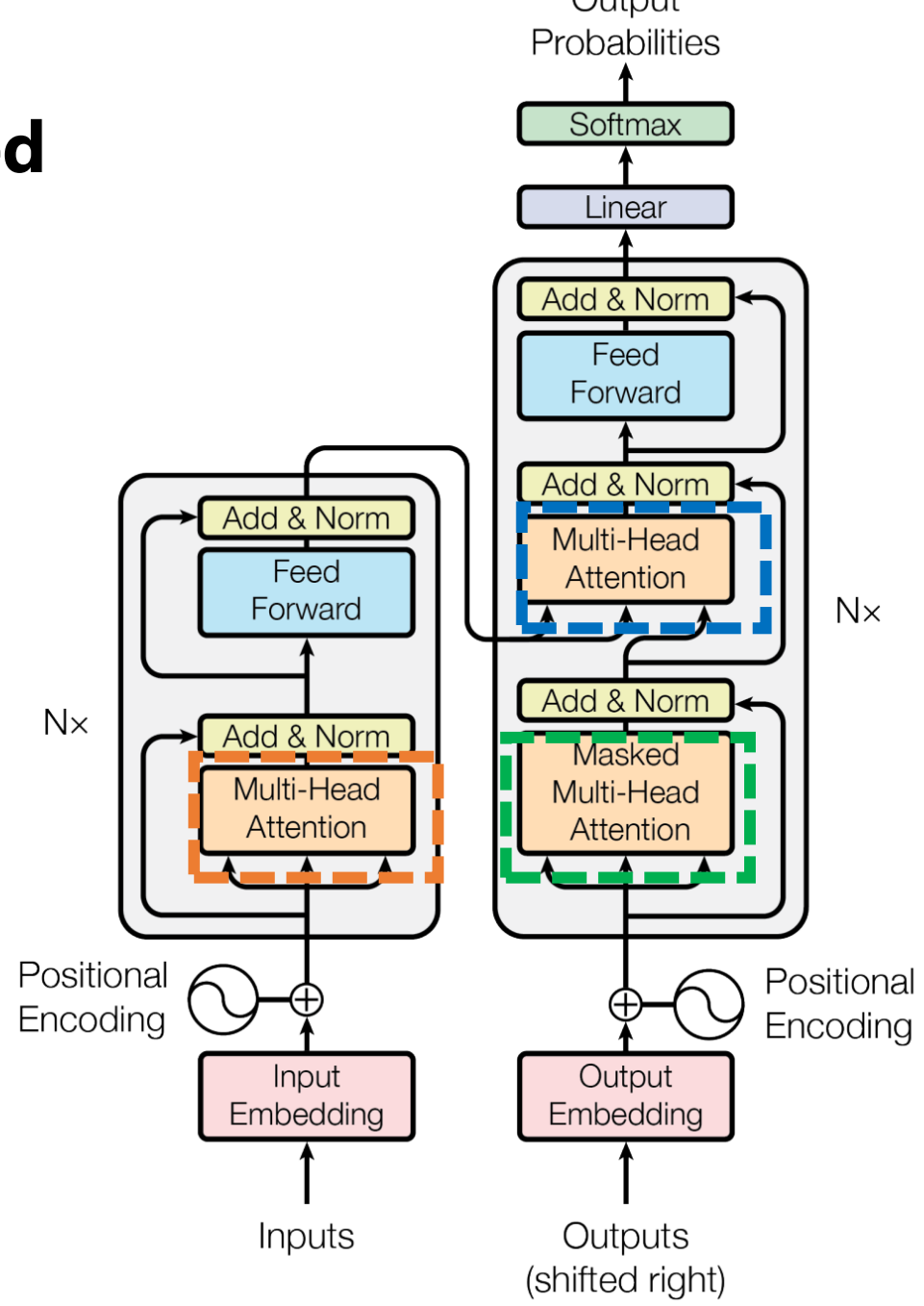
Attention Is All You Need

- Three attentions

Self-attention (encoder)

Self-attention (decoder)

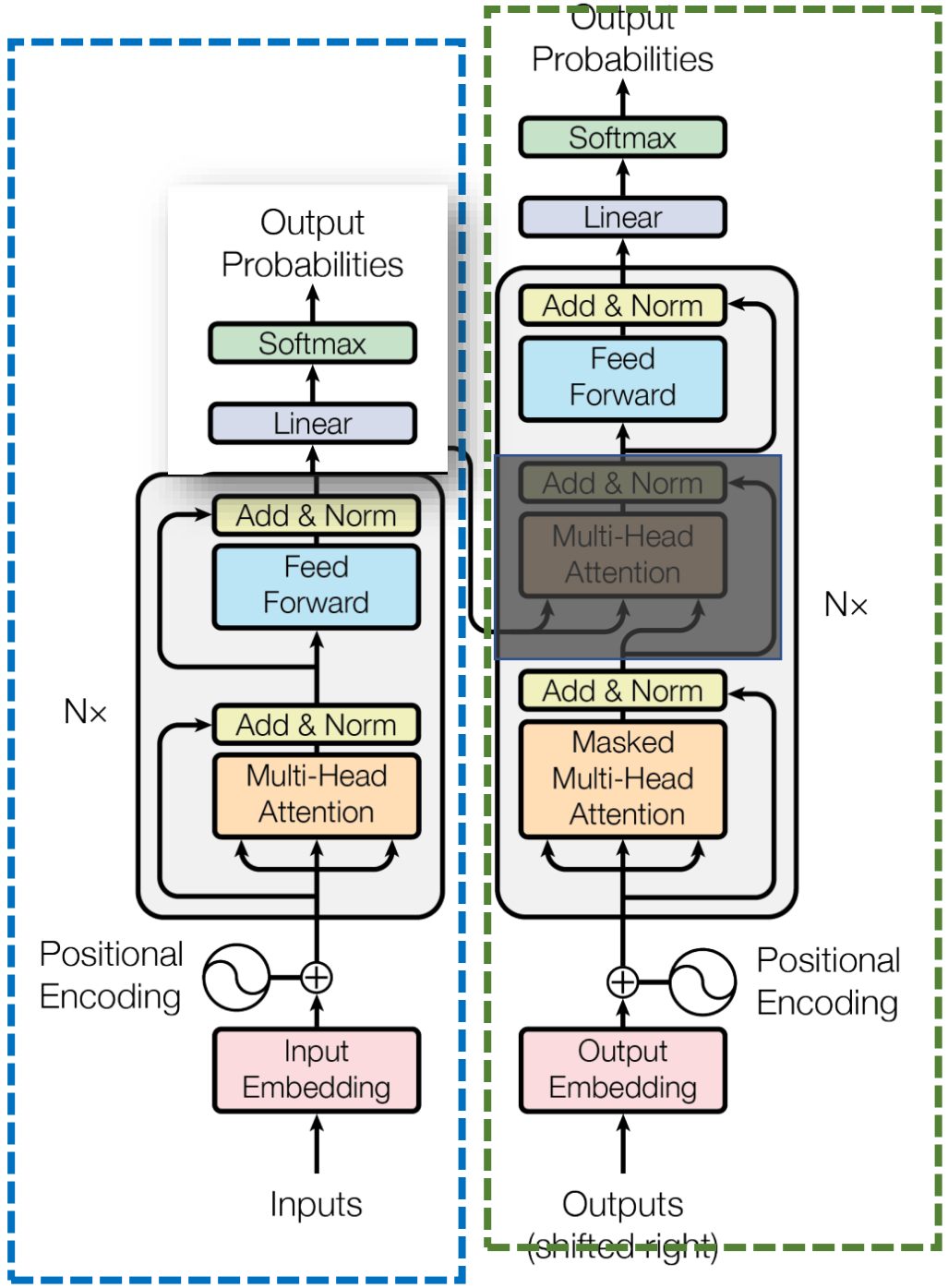
Encoder-decoder attention



Encoder + Big data

auto-encoder

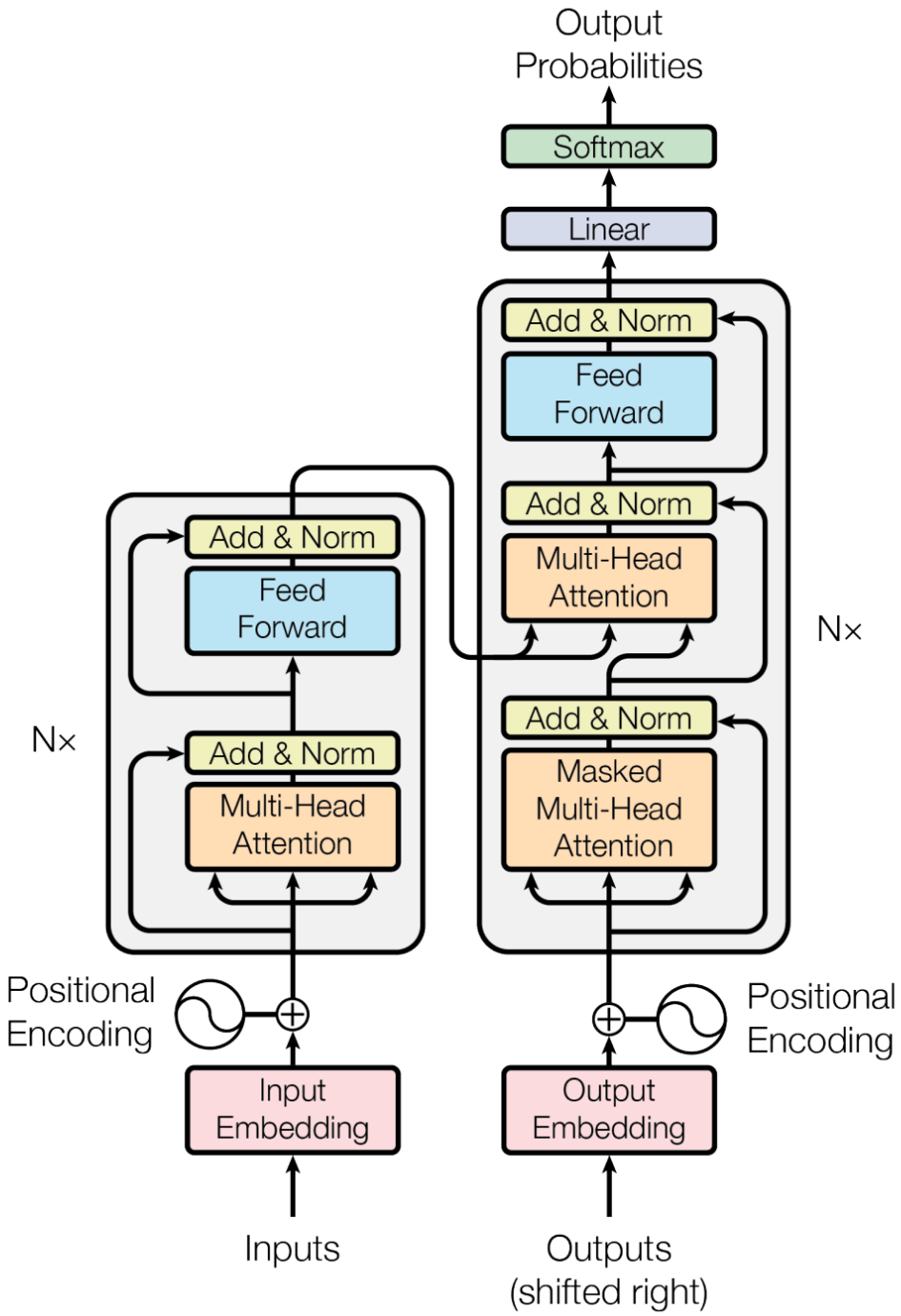
BERT (2018)
RoBERTa (2019)
ALBERT (2019)
DistilBERT (2019)
Reformer (2020)
Electra (2020)
...



Decoder + Big data

auto-regressive

GPT (2018)
GPT-2 (2019)
GPT-3 (2020)
GPT-4 (2023)
XLNet (2019)
...



Encoder + Decoder + Big data

BART (2019)
T5 (2020)
mBART (2020)
...

Outline

- GPT-1,2,3
- Homework
  Appendix: Sentiment analysis with BERT
- Artificial General Intelligence

Generative Pre-trained Transformers GPT-1,2,3

Deep Learning

Woohwan Jung

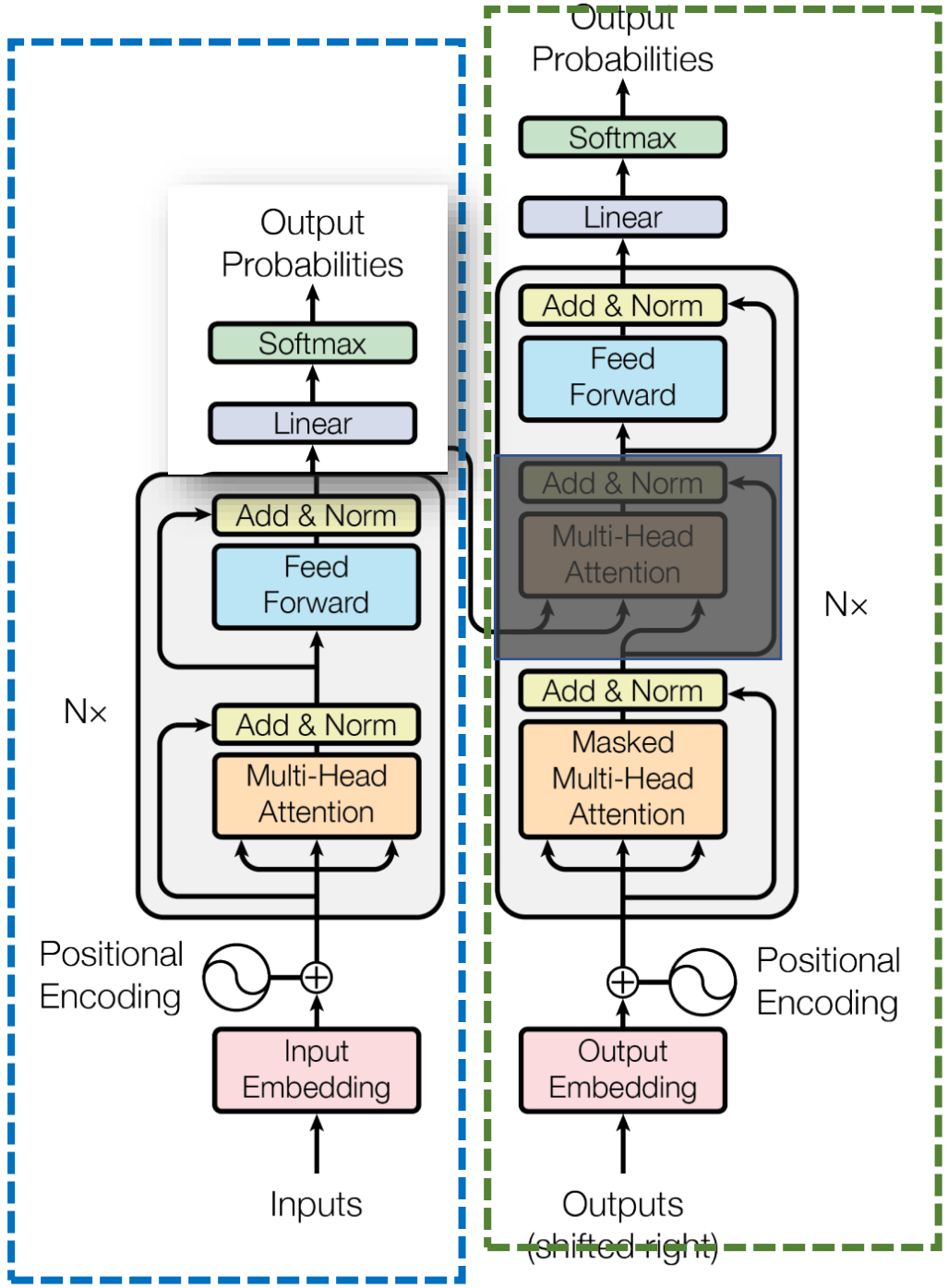
한양대학교 HANYANG UNIVERSITY

DSLAB 데이터사이언스 연구실

Encoder + Big data

auto-encoder

BERT (2018)
RoBERTa (2019)
ALBERT (2019)
DistilBERT (2019)
Reformer (2020)
Electra (2020)
...



Decoder + Big data

auto-regressive

GPT (2018)
GPT-2 (2019)
GPT-3 (2020)
GPT-4 (2023)
XLNet (2019)
...

Encoder + Decoder + Big data

BART (2019)
T5 (2020)
mBART (2020)
...

Outline

- GPT1 (2018)
- GPT2 (2019)
- GPT3 (2020)

- ChatGPT (2022)
- GPT4 (2023)

GPT-1

Improving Language Understanding by Generative Pre-Training

OpenAI 2018

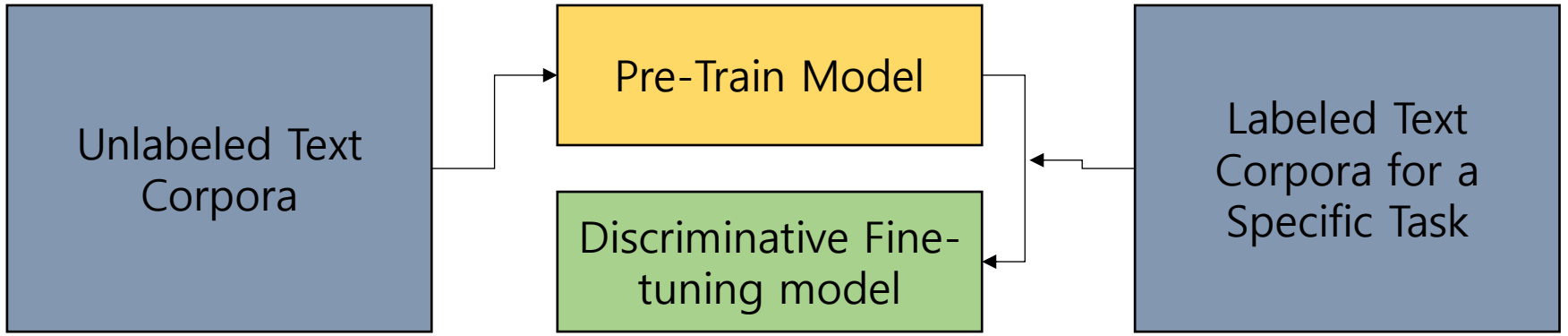
From Derek Hoiem’s slides

GPT1 (2018)

- Pre-cursor to BERT (2019) that we discussed last class
- Similar architecture and training procedures
  - 117M parameters in GPT1 vs. 340M for BERT Large
- Pre-training: Maximize data likelihood as a product of conditional probabilities, trained on Books Corpus
  - Predict each token based on the k tokens (the "context") that came before

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta)$$

- Fine-tuned for each task while also retaining the generative objective
- Achieved state-of-art in 9 out of 12 tasks



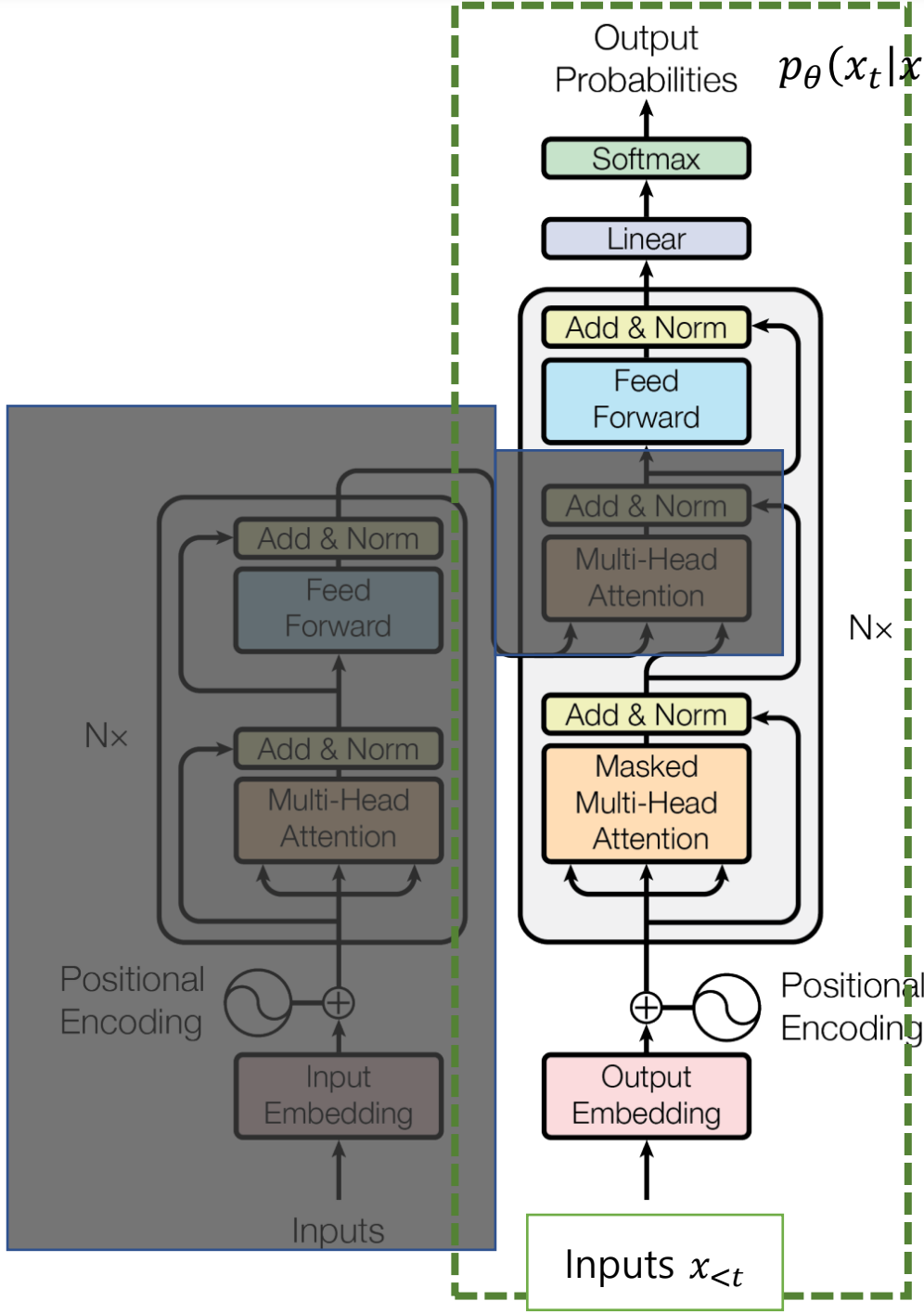
Autoregressive language modeling

- Next token prediction

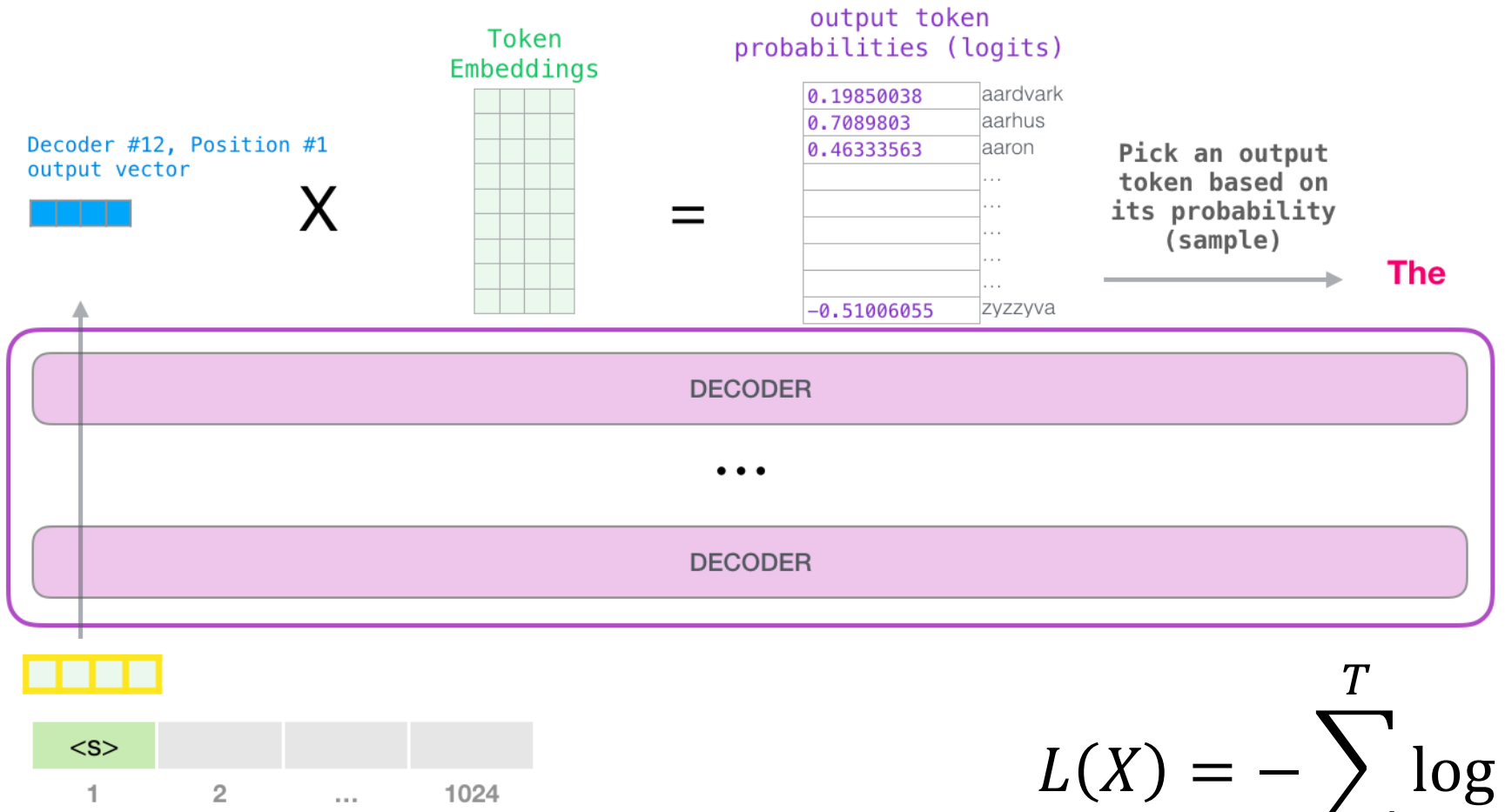
$$p(X) = \prod_{t=1}^{T} p(x_t | x_{<t})$$

- Loss function:
  - The sum of negative log-probabilities
  - = The sum of cross-entropy losses

$$L(X) = -\log p(X) = -\sum_{t=1}^{T} \log p_\theta(x_t | x_{<t})$$



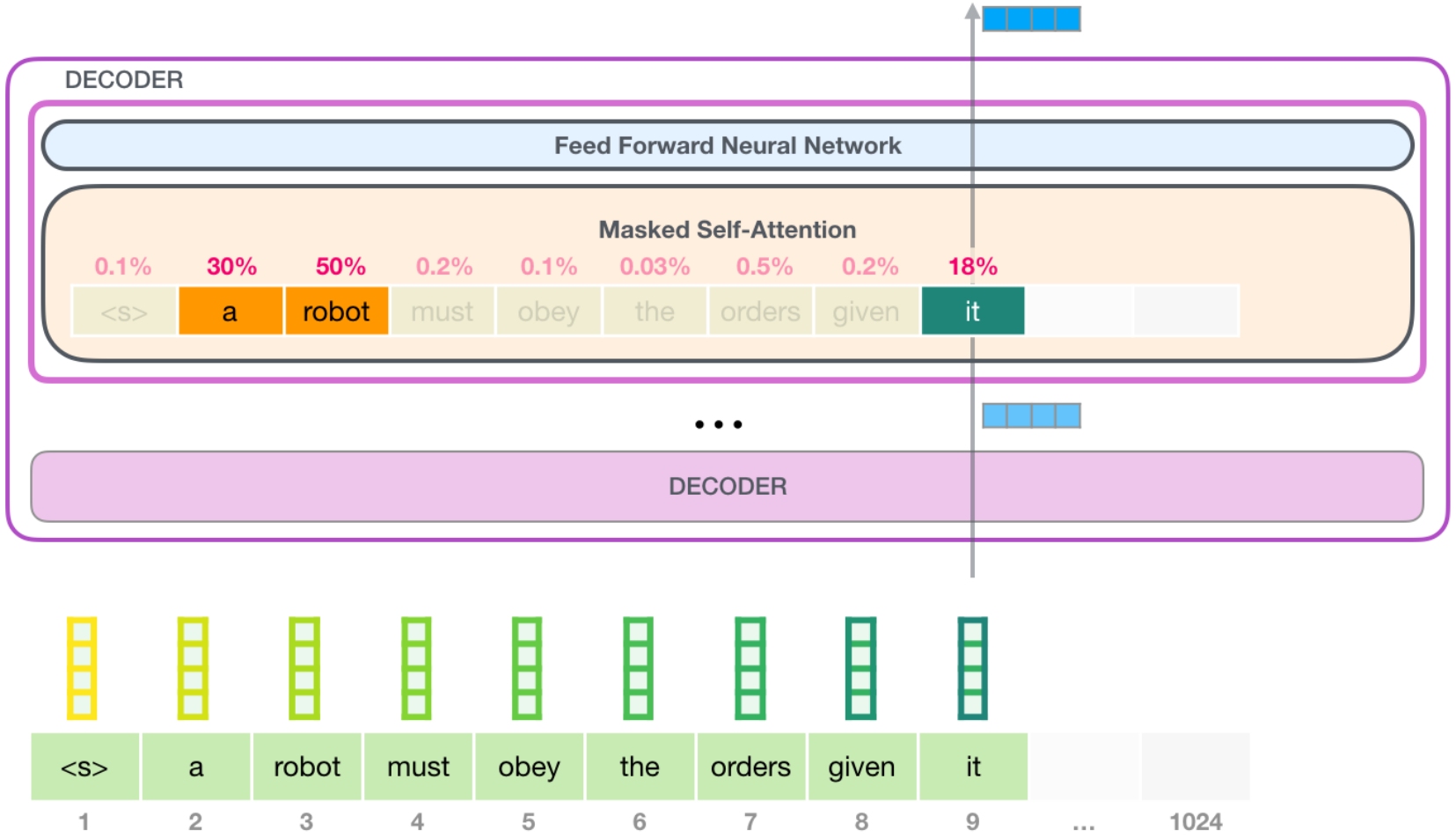
Next token prediction



$$L(X) = -\sum_{t=1}^{T} \log p_\theta(x_t | x_{<t})$$

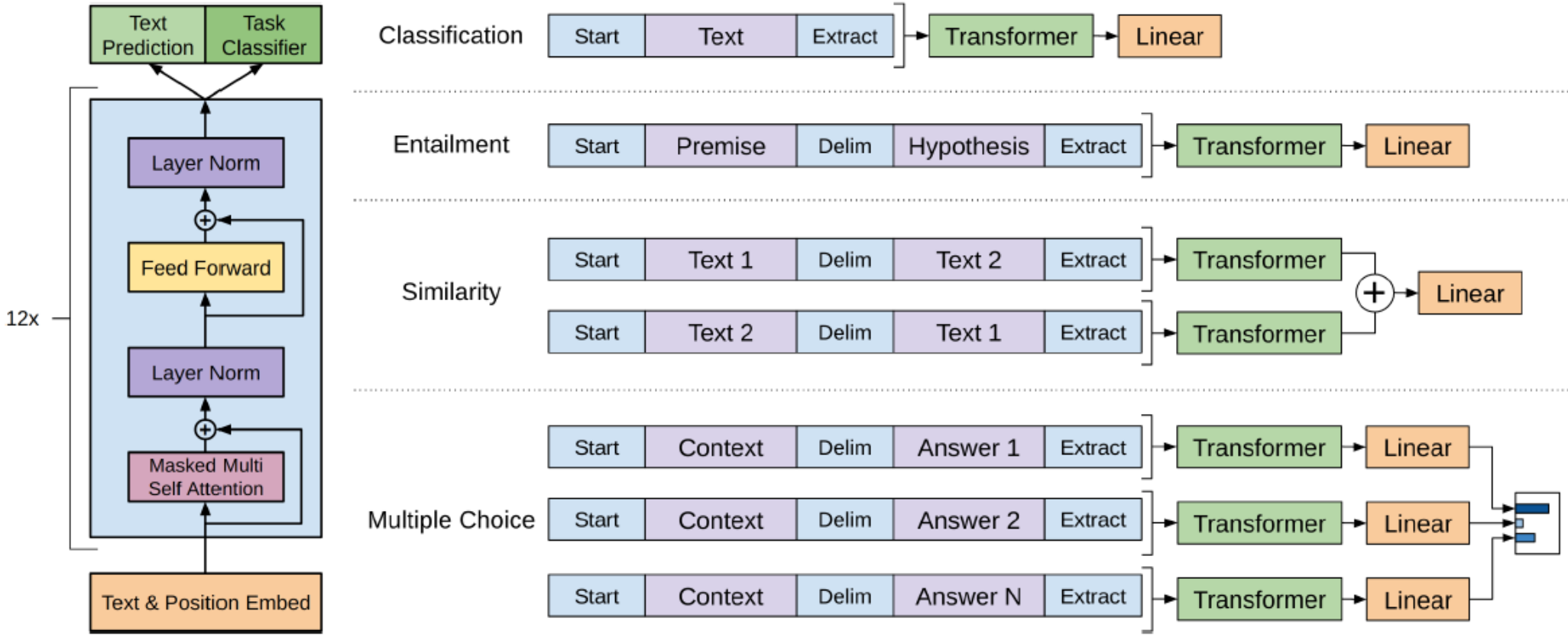
* https://jalammar.github.io/illustrated-gpt2/

Next token prediction

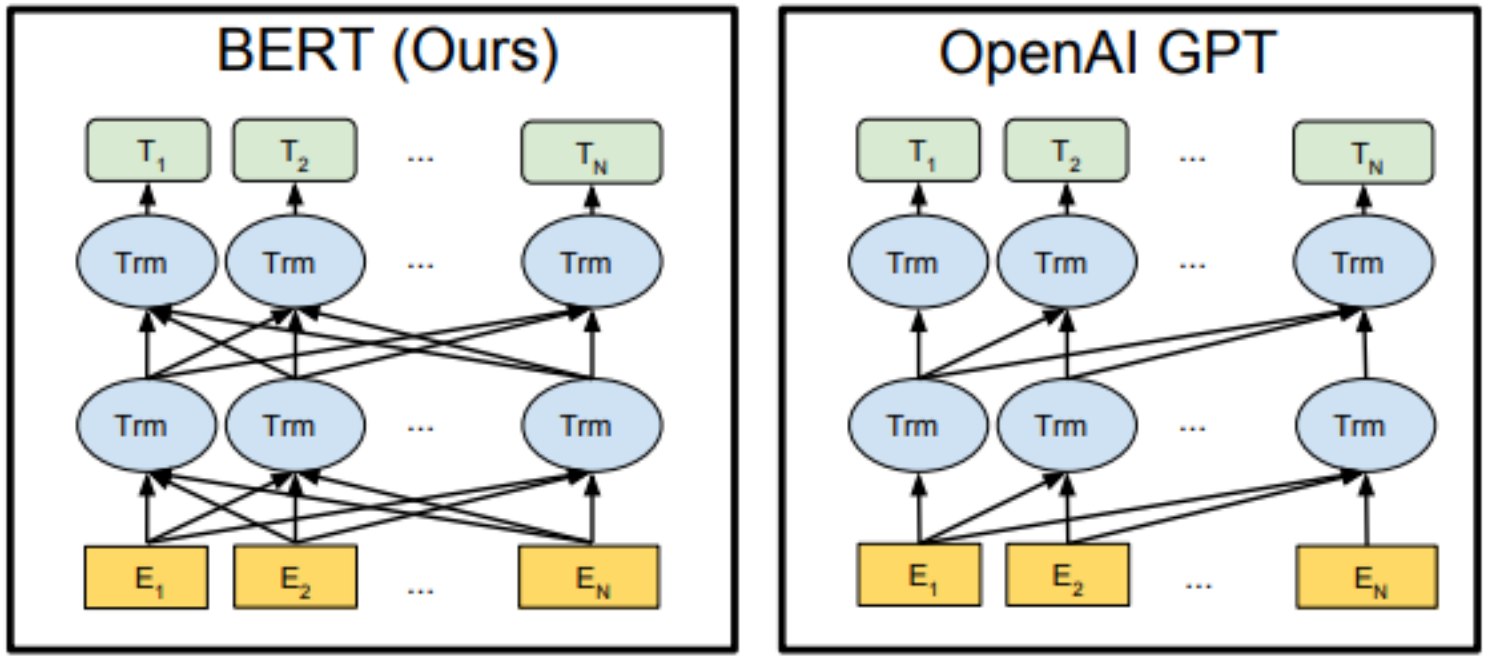


* https://jalammar.github.io/illustrated-gpt2/

Fine-tuning


Text Prediction
Task Classifier
Layer Norm
Feed Forward
Layer Norm
Masked Multi Self Attention
12x
Text & Position Embed
Classification
Start
Text
Extract
Transformer
Linear
Entailment
Start
Premise
Delim
Hypothesis
Extract
Transformer
Linear
Similarity
Start
Text 1
Delim
Text 2
Extract
Start
Text 2
Delim
Text 1
Extract
Transformer
Transformer
Linear
Multiple Choice
Start
Context
Delim
Answer 1
Extract
Start
Context
Delim
Answer 2
Extract
Start
Context
Delim
Answer N
Extract
Transformer
Linear
Transformer
Linear
Transformer
Linear

BERT and GPT


BERT (Ours)
T1
T2
...
TN
Trm
Trm
...
Trm
Trm
Trm
...
Trm
E1
E2
...
EN
OpenAI GPT
T1
T2
...
TN
Trm
Trm
...
Trm
Trm
Trm
...
Trm
E1
E2
...
EN

GPT-2

Language Models are Unsupervised Multitask Learners

OpenAI 2018

From Derek Hoiem’s slides

Language Models are Unsupervised Multitask Learners

GPT-2 (Radford et al. 2019)

Aims to create a general purpose language learner

“Current systems are better characterized as narrow experts rather than competent generalists. We would like to move towards more general systems which can perform many tasks – eventually without the need to manually create and label a training dataset for each one.

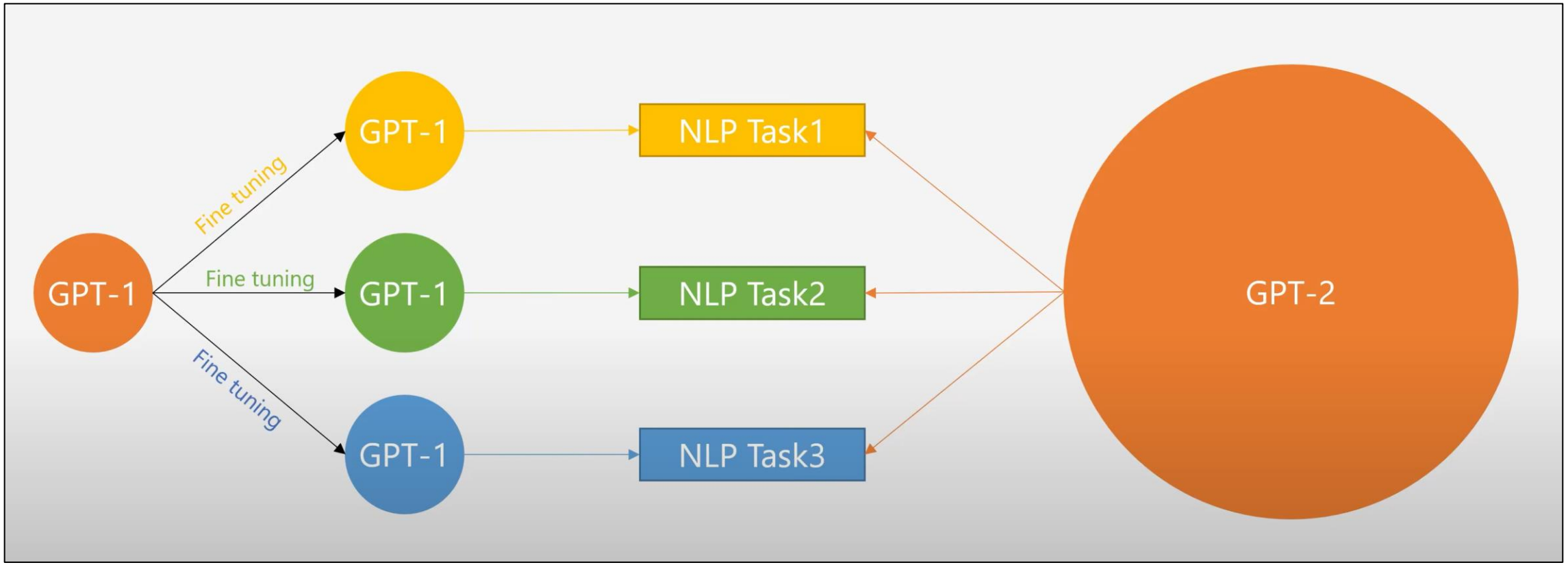
The dominant approach to creating ML systems is to collect a dataset of training examples demonstrating correct behavior for a desired task, train a system to imitate these behaviors, and then test its performance on independent and identically distributed (IID) held-out examples. This has served well to make progress on narrow experts. But the often erratic behavior of captioning models (Lake et al., 2017), reading comprehension systems (Jia & Liang, 2017), and image classifiers (Alcorn et al., 2018) on the diversity and variety of possible inputs highlights some of the shortcomings of this approach.

Our suspicion is that the prevalence of single task training on single domain datasets is a major contributor to the lack of generalization observed in current systems. Progress towards robust systems with current architectures is likely to require training and measuring performance on a wide range of domains and tasks.”

PDF

Language Models are Unsupervised Multitask Learners

GPT1 vs GPT2



* GPT-2 (밑바닥부터 알아보는 GPT 2강) https://www.youtube.com/watch?v=3n6157XNYyw

GPT-2

- A general systems should learn to model $P(output|input, task)$
- The task can be specified in natural language, so language tasks can be framed as sequenc e-to-sequence text processing
- Example)
  - Translation: `(translate to french, english text, french text)`
  - Reading comprehension: `(answer the question, document, question, answer)`

Motivation

”I’m not the cleverest man in the world, but like they say in French: **Je ne suis pas un imbecile [I’m not a fool].**
In a now-deleted post from Aug. 16, Soheil Eid, Tory candidate in the riding of Joliette, wrote in French: ”**Mentez mentez, il en restera toujours quelque chose**,” which translates as, ”**Lie lie and something will always remain.**”
“I hate the word ‘**perfume**,”’ Burr says. ‘It’s somewhat better in French: ‘**parfum**.’
If listened carefully at 29:55, a conversation can be heard between two guys in French: “**-Comment on fait pour aller de l’autre coté? -Quel autre coté?**”, which means “**- How do you get to the other side? - What side?**”.
If this sounds like a bit of a stretch, consider this question in French: **As-tu aller au cinéma?**, or **Did you go to the movies?**, which literally translates as Have-you to go to movies/theater?
“**Brevet Sans Garantie Du Gouvernement**”, translated to English: “**Patented without government warranty**”.

Table 1. Examples of naturally occurring demonstrations of English to French and French to English translation found throughout the WebText training set.

GPT-2 Architecture and Model Sizes

- Architecture is basically the same as GPT-1

Parameters	Layers	d_{model}
117M	12	768
345M	24	1024
762M	36	1280
1542M	48	1600

GPT-1 Size

BERT Size

GPT-2: Zero shot results

	LAMBADA (PPL)	LAMBADA (ACC)	CBT-CN (ACC)	CBT-NE (ACC)	WikiText2 (PPL)	PTB (PPL)	enwik8 (BPB)	text8 (BPC)	WikiText103 (PPL)	1BW (PPL)
SOTA	99.8	59.23	85.7	82.3	39.14	46.54	0.99	1.08	18.3	**21.8**
117M	**35.13**	45.99	**87.65**	**83.4**	**29.41**	65.85	1.16	1.17	37.50	75.20
345M	**15.60**	55.48	**92.35**	**87.1**	**22.76**	47.33	1.01	**1.06**	26.37	55.72
762M	**10.87**	**60.12**	**93.45**	**88.0**	**19.93**	**40.31**	**0.97**	**1.02**	22.05	44.575
1542M	**8.63**	**63.24**	**93.30**	**89.05**	**18.34**	**35.76**	**0.93**	**0.98**	**17.48**	42.16

Table 3. Zero-shot results on many datasets. No training or fine-tuning was performed for any of these results. PTB and WikiText-2 results are from (Gong et al., 2018). CBT results are from (Bajgar et al., 2016). LAMBADA accuracy result is from (Hoang et al., 2018) and LAMBADA perplexity result is from (Grave et al., 2016). Other results are from (Dai et al., 2019).

Perplexity (PPL) is 2^entropy; lower is better

- Achieves state-of-art in many tasks without tuning for them

- Performs much worse than state-of-art in summarization and translation (though can effect ively translate word for word)

Continued log-linear improvement with model size

Conclusion: “The diversity of tasks the model is able to perform in a zero-shot setting suggests that **high-capacity models trained to maximize the likelihood of a sufficiently varied text corpus begin to learn how to perform a surprising amount of tasks without the need for explicit supervision.**”

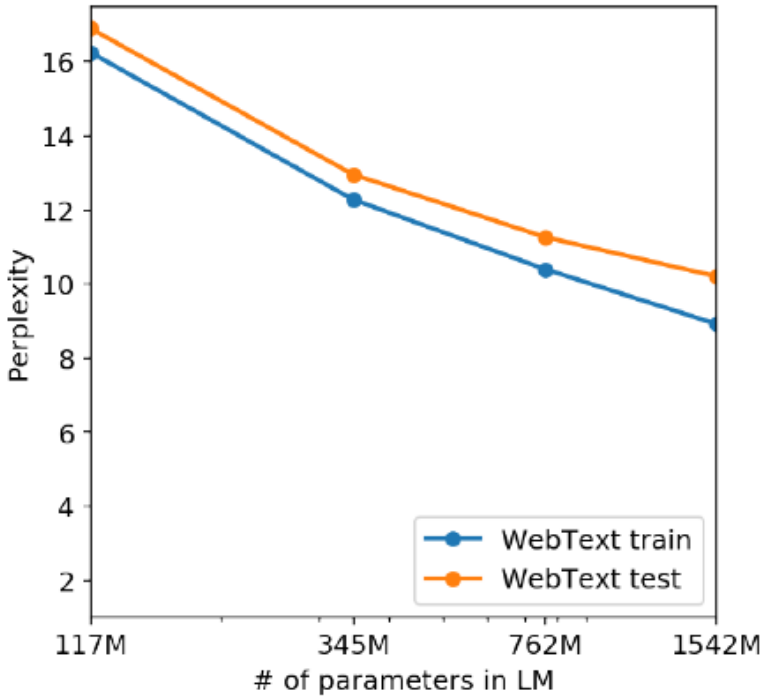
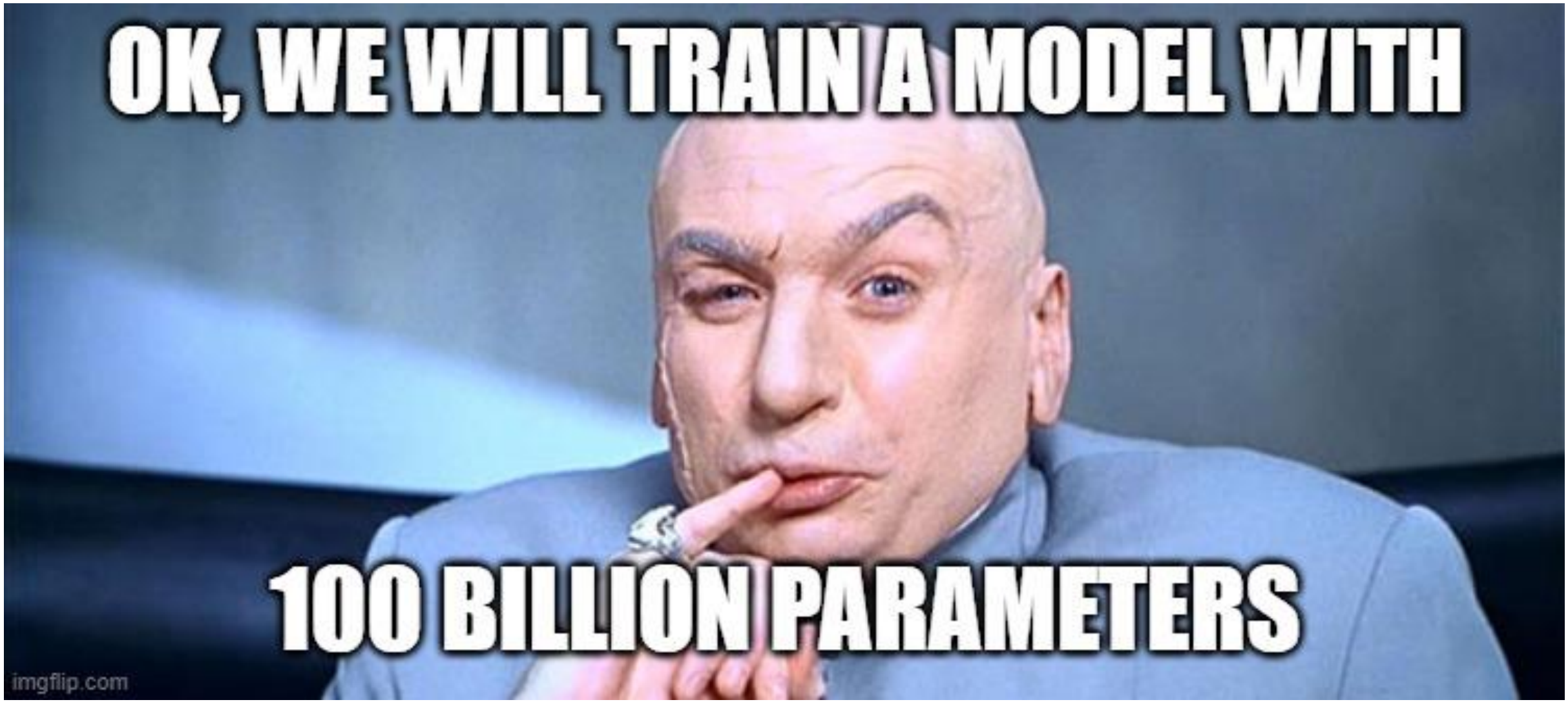


Figure 4. The performance of LMs trained on WebText as a function of model size.


OK, WE WILL TRAIN A MODEL WITH
100 BILLION PARAMETERS
imgflip.com

GPT-3 (Brown et al. 2020)

Language Models are Few-Shot Learners

Tom B. Brown* Benjamin Mann* Nick Ryder* Melanie Subbiah*

Jared Kaplan† Prafulla Dhariwal Arvind Neelakantan Pranav Shyam Girish Sastry

Amanda Askell Sandhini Agarwal Ariel Herbert-Voss Gretchen Krueger Tom Henighan

Rewon Child Aditya Ramesh Daniel M. Ziegler Jeffrey Wu Clemens Winter

Christopher Hesse Mark Chen Eric Sigler Mateusz Litwin Scott Gray

Benjamin Chess Jack Clark Christopher Berner

Sam McCandlish Alec Radford Ilya Sutskever Dario Amodei

OpenAI

Models and Architectures

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

Table 2.1: Sizes, architectures, and learning hyper-parameters (batch size in tokens and learning rate) of the models which we trained. All models were trained for a total of 300 billion tokens.

Training data

Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4

Table 2.2: Datasets used to train GPT-3. "Weight in training mix" refers to the fraction of examples during training

Training cost

- OpenAI recently published GPT-3, the largest language model ever trained. GPT-3 has 175 billion parameters and would require 355 years and **$4,600,000** to train - even with the lowest priced GPU cloud on the market.[1]
-

https://lambdalabs.com/blog/demystifying-gpt-3

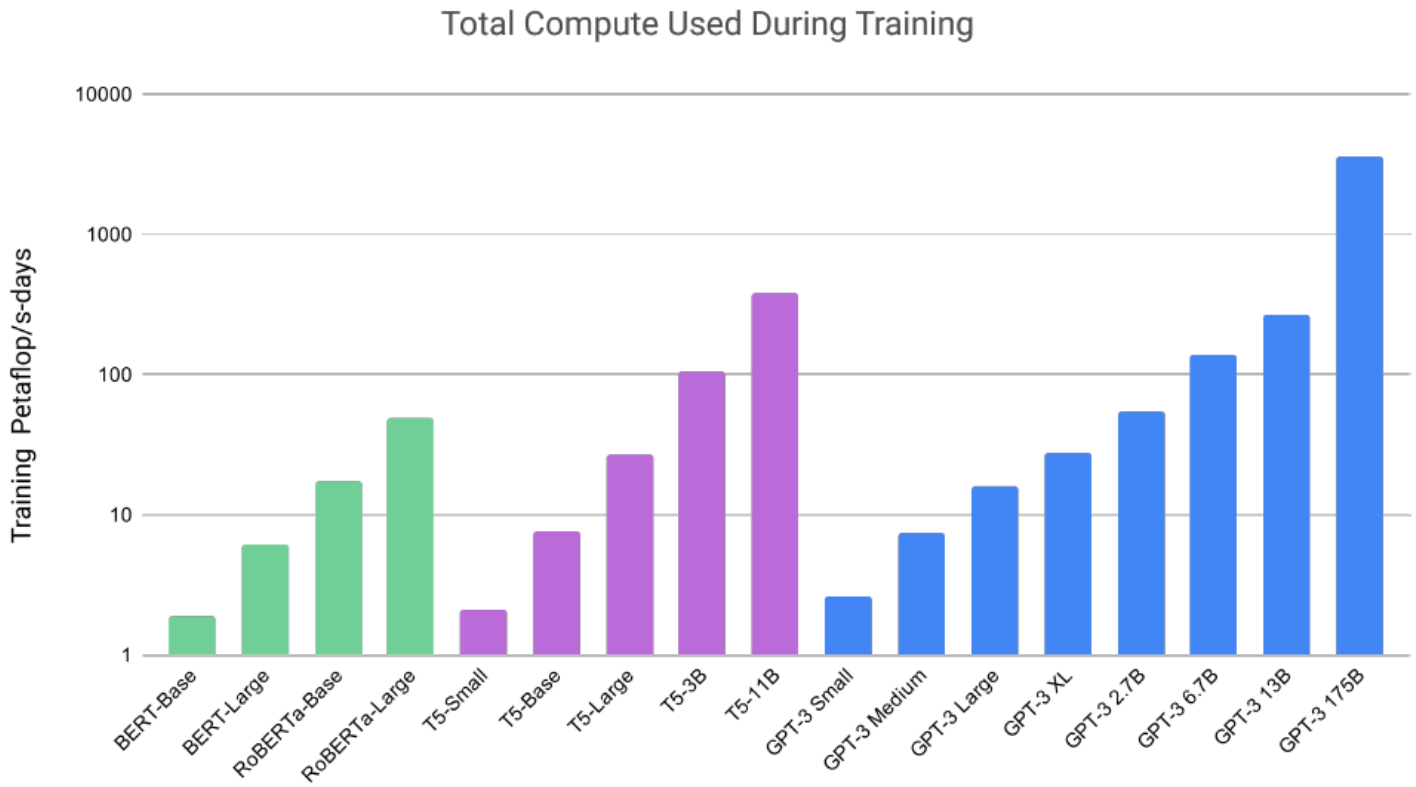


Figure 2.2: Total compute used during training. Based on the analysis in Scaling Laws For Neural Language Models [KMH+20] we train much larger models on many fewer tokens than is typical. As a consequence, although GPT-3 3B is almost 10x larger than RoBERTa-Large (355M params), both models took roughly 50 petaflop/s-days of compute during pre-training. Methodology for these calculations can be found in Appendix D.

Unfortunately, a bug in the filtering caused us to ignore some overlaps, and due to the cost of training it was not feasible to retrain the model. In Section 4 we characterize the impact of the remaining overlaps, and in future work we will

Few-shot
“In Context Learning”

The three settings we explore for in-context learning

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  cheese =>                           ← prompt
```

One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← example
3  cheese =>                           ← prompt
```

Few-shot

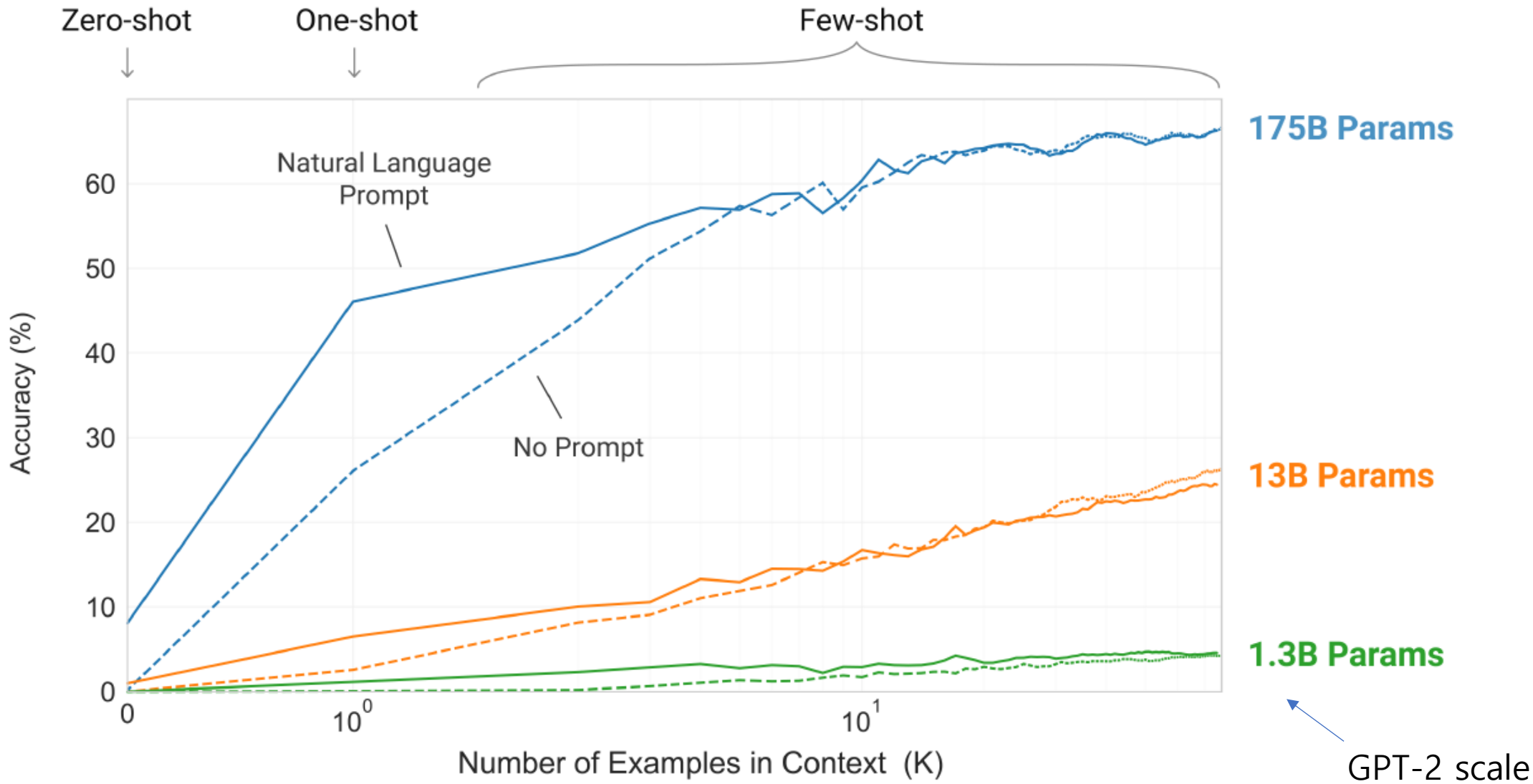
In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← examples
3  peppermint => menthe poivrée        ←
4  plush girafe => girafe peluche      ←
5  cheese =>                           ← prompt
```

Traditional fine-tuning (not used for GPT-3)

Fine-tuning

The model is trained via repeated gradient updates using a large corpus of example tasks.



GPT-2 scale

Accuracy on a simple task to remove random symbols from a word

Performance of GPT-3

	SuperGLUE Average	BoolQ Accuracy	CB Accuracy	CB F1	COPA Accuracy	RTE Accuracy
Fine-tuned SOTA	**89.0**	**91.0**	**96.9**	**93.9**	**94.8**	**92.5**
Fine-tuned BERT-Large	69.0	77.4	83.6	75.7	70.6	71.7
GPT-3 Few-Shot	71.8	76.4	75.6	52.0	92.0	69.0

	WiC Accuracy	WSC Accuracy	MultiRC Accuracy	MultiRC F1a	ReCoRD Accuracy	ReCoRD F1
Fine-tuned SOTA	**76.1**	**93.8**	**62.3**	**88.2**	**92.5**	**93.3**
Fine-tuned BERT-Large	69.6	64.6	24.1	70.0	71.3	72.0
GPT-3 Few-Shot	49.4	80.1	30.5	75.4	90.2	91.1

- Average performance of few-shot is about the same as fine-tuned BERT-Large, but varies by task
- Per-task specialized SOTA models are still best

What to learn from the GPT Series

- GPT: generative-pretraining (GPT) is effective for large language models
- GPT-2: GPT models can perform reasonable zero-shot task performance with larger models trained on more data
- GPT-3: Even larger GPT models trained on even more data are good at many tasks, especially text generation, and can be “trained” at inference time with in-context examples

Homework

- Finetuning BERT for any task
- Report
  - Source code
  - Test result
- Due: 2023/12/20 PM 11:59

Appendix: Sentiment Analysis on IMDB Movie Reviews using BERT Model

Deep Learning

한양대학교
HANYANG UNIVERSITY

DSLAB
데이터사이언스 연구실

Setup

- Install the modules

✓ 3분
```
[2]  !pip install transformers
     !pip install portalocker
     !pip install torchtext==0.15.2
```

- **Load the libraries**

✓ 2초
```
[4]  import random
     import numpy as np
     from tqdm import tqdm

     import torch
     import torch.nn.functional as F
     from torch.utils.data import Dataset, DataLoader

     from transformers import AdamW, get_linear_schedule_with_warmup
```

Setup

- **Set the seed**
  - In order to ensure consistent and reproducible results during model training and inference,
  - it is essential to set the seed value for random number generation to the same value for identical inputs.

```
[5]  # set seed
     SEED = 1234
     random.seed(SEED)
     np.random.seed(SEED)
     torch.manual_seed(SEED)
     torch.cuda.manual_seed(SEED)
```

- **Set the device**
  - Setting the device to use ‘GPU(cuda)’ or ‘CPU’ during the model training and inference.

```
# set device
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(device)
```

```
cuda
```

Dataset

- Load the IMDB Dataset from Torchtext

  https://pytorch.org/text/stable/datasets.html#imdb

```
[8]  from torchtext.datasets import IMDB

     # Load IMDB Dataset
     train_iter, test_iter = IMDB(split=('train', 'test'))
```

```
[9]  list(train_iter)
```

Label

Text (review)

```
[(1,
  'I rented I AM CURIOUS-YELLOW from my video store because of all the controversy that
surrounded it when it was first released in 1967. I also heard that at first it was seized
by U.S. customs if it ever tried to enter this country, therefore being a fan of films
considered "controversial" I really had to see this for myself.<br /><br />The plot is
centered around a young Swedish drama student named Lena who wants to learn everything she
can about life. In particular she wants to focus her attentions to making some sort of
documentary on what the average Swede thought about certain political issues such as the
Vietnam War and race issues in the United States. In between asking politicians and
ordinary denizens of Stockholm about their opinions on politics, she has sex with her
drama teacher, classmates, and married men.<br /><br />What kills me about I AM CURIOUS-
YELLOW is that 40 years ago, this was considered pornographic. Really, the sex and nudity
scenes are few and far between, even then it\'s not shot like some cheaply made porno.
While my countrymen mind find it shocking, in reality sex and nudity are a major staple in
Swedish cinema. Even Ingmar Bergman, arguably their answer to good old boy John Ford, had
sex scenes in his films.<br /><br />I do commend the filmmakers for the fact that any sex
shown in the film is shown for artistic purposes rather than just to shock people and make
money to be shown in pornographic theaters in America. I AM CURIOUS-YELLOW is a good film
for anyone wanting to study the meat and potatoes (no pun intended) of Swedish cinema. But
really, this film doesn\'t have much of a plot.'),
```

Docs > torchtext.datasets

IMDb

torchtext.datasets.IMDB(*root: str = '.data', split: Union[Tuple[str], str] = ('train', 'test')*) [SOURCE]

IMDB Dataset

WARNING

using datapipes is still currently subject to a few caveats. if you wish to use this dataset with shuffling, multi-processing, or distributed learning, please see this note for further instructions.

For additional details refer to http://ai.stanford.edu/~amaas/data/sentiment/

Number of lines per split:

- train: 25000
- test: 25000

Parameters:

- **root** – Directory where the datasets are saved. Default: os.path.expanduser('~/.torchtext/cache')
- **split** – split or splits to be returned. Can be a string or tuple of strings. Default: (*train*, *test*)

Returns:

DataPipe that yields tuple of label (1 to 2) and text containing the movie review

Return type:

(int, str)

Dataset

- Create custom dataset for using IMDB data

```
[6]  class IMDBDataset(Dataset):
         def __init__(self, data, tokenizer):
             self.data = data
             self.tokenizer = tokenizer
             self.max_len = 512

         def __len__(self):
             return len(self.data)

         def __getitem__(self, index):
             (label, text) = self.data[index]
             # convert labels (1,2) to (0,1)
             conv_label = 1 if label == 2 else 0
             encoding = self.tokenizer(text,
                                       truncation=True,
                                       max_length=self.max_len,
                                       padding='max_length',
                                       return_tensors='pt')

             return {
                     'input_ids' : encoding['input_ids'].flatten(),
                     'attention_mask' : encoding['attention_mask'].flatten(),
                     'label' : torch.tensor(conv_label, dtype=torch.long)
                     }
```

- Negative = 1 → 0
- Positive = 2 → 1

Model

- Load the BERT tokenizer and model parameters from Huggingface

  https://huggingface.co/docs/transformers/model_doc/bert#transformers.BertForSequenceClassification

```
[7]  from transformers import BertTokenizer, BertForSequenceClassification

     # Load the BERT model and tokenizer from huggingface
     tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
     model = BertForSequenceClassification.from_pretrained('bert-base-uncased', num_labels=2)
     model.to(device)
```

```
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at bert-base-uncased
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
BertForSequenceClassification(
  (bert): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(30522, 768, padding_idx=0)
      (position_embeddings): Embedding(512, 768)
      (token_type_embeddings): Embedding(2, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): BertEncoder(
      (layer): ModuleList(
        (0-11): 12 x BertLayer(
          (attention): BertAttention(
            (self): BertSelfAttention(
              (query): Linear(in_features=768, out_features=768, bias=True)
              (key): Linear(in_features=768, out_features=768, bias=True)
              (value): Linear(in_features=768, out_features=768, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
```

Train

- Load mini-batch using Dataloader

```
[9]  train_dataset = IMDBDataset(list(train_iter), tokenizer)
     test_dataset = IMDBDataset(list(test_iter), tokenizer)
     train_dataloader = DataLoader(train_dataset,
                                   batch_size=8,
                                   shuffle=True)
     test_dataloader = DataLoader(test_dataset,
                                   batch_size=8,
                                   shuffle=False)
```

- **Set the optimizer and scheduler**

```
[10] optimizer = AdamW(model.parameters(), lr=2e-5)
     num_training_steps = len(train_dataloader) * 2
     scheduler = get_linear_schedule_with_warmup(optimizer, num_warmup_steps=0, num_training_steps=num_training_steps)
```

Train

- **Model Training**
  - Set the model in training mode

✓ 1시간

```
[11] # Train
     num_epochs = 3
     for epoch in range(num_epochs):
         # set the model to training mode
         model.train()
         total_loss = 0
         for batch in tqdm(train_dataloader):
             input_ids = batch['input_ids'].to(device)
             attention_mask = batch['attention_mask'].to(device)
             labels = batch['label'].to(device)

             optimizer.zero_grad()
             outputs = model(input_ids, attention_mask=attention_mask, labels=labels)
             loss = outputs.loss
             total_loss += loss.item()

             loss.backward()
             optimizer.step()
             scheduler.step()

         average_loss = total_loss / len(train_dataloader)
         print(f"Epoch {epoch + 1}, Average Loss: {average_loss}")
```

```
100%|██████████| 3125/3125 [39:52<00:00,  1.31it/s]
Epoch 1, Average Loss: 0.23159894025303424
100%|██████████| 3125/3125 [39:51<00:00,  1.31it/s]
Epoch 2, Average Loss: 0.08994403915494681
100%|██████████| 3125/3125 [39:51<00:00,  1.31it/s]
Epoch 3, Average Loss: 0.04703274596109986
```

Evaluate

- **Model Evaluation**
  - Set the model in evaluation mode

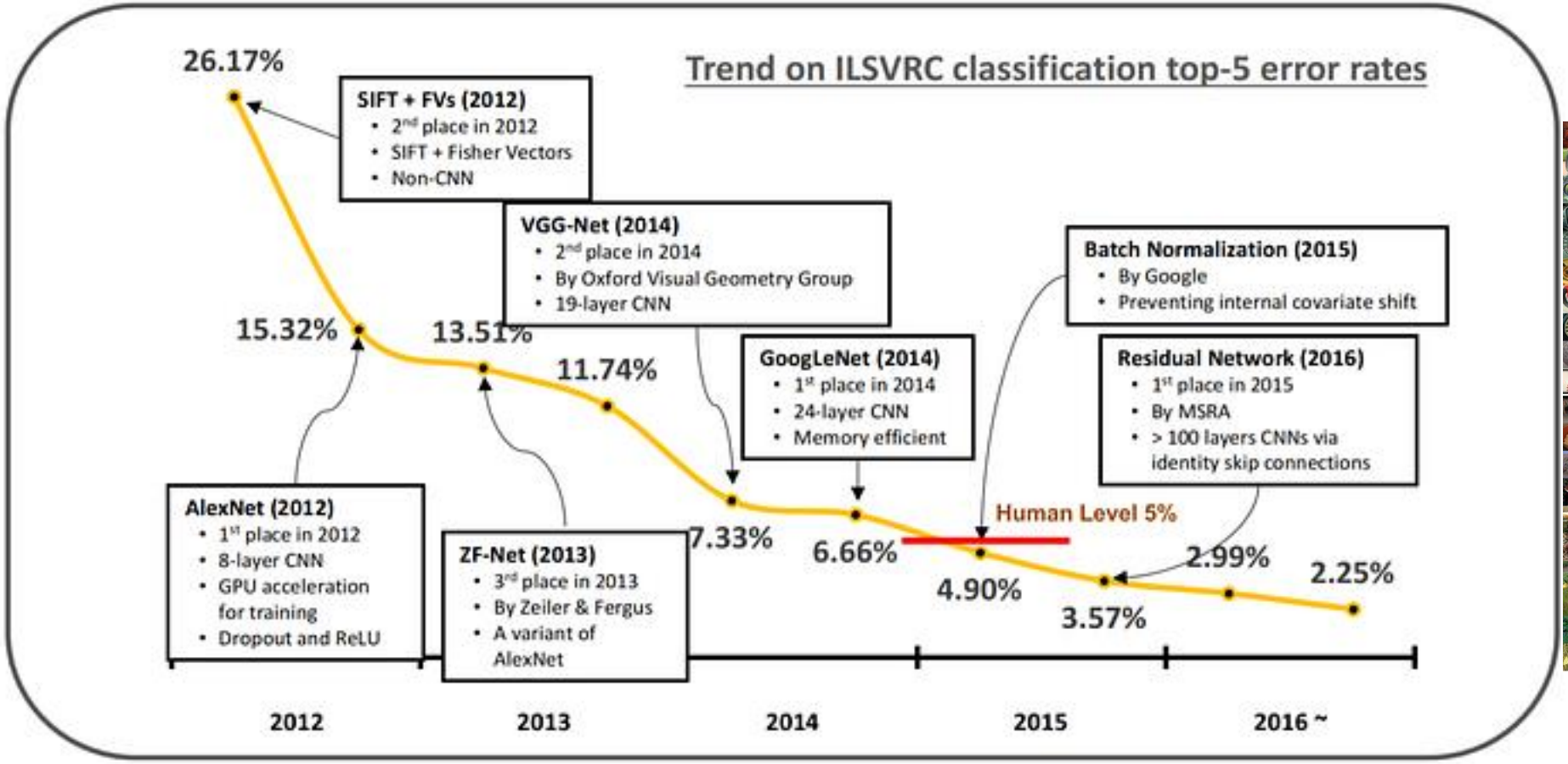
✓ 15분

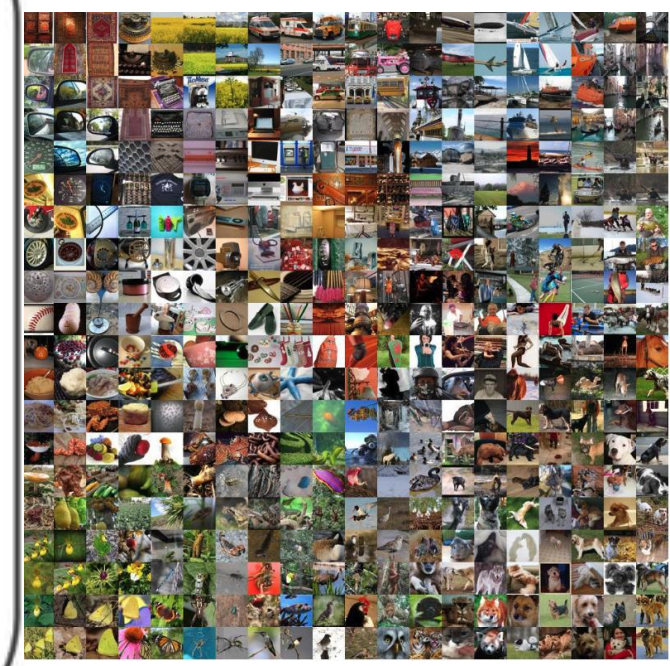
```
[12] # Test
     # set the model to evaluation mode
     model.eval()
     correct = 0
     total = 0

     with torch.no_grad():
         for batch in tqdm(test_dataloader):
             input_ids = batch['input_ids'].to(device)
             attention_mask = batch['attention_mask'].to(device)
             labels = batch['label'].to(device)

             outputs = model(input_ids, attention_mask=attention_mask)
             logits = outputs.logits
             predictions = F.softmax(logits, dim=1)
             predicted_labels = torch.argmax(predictions, dim=1)

             total += labels.size(0)
             correct += (predicted_labels == labels).sum().item()

     accuracy = correct / total
     print(f"Validation Accuracy: {accuracy * 100:.2f}%")
```

```
100%|██████████| 3125/3125 [15:07<00:00,  3.44it/s]
Validation Accuracy: 94.24%
```

Appendix: Artificial General Intelligence. Where We Are

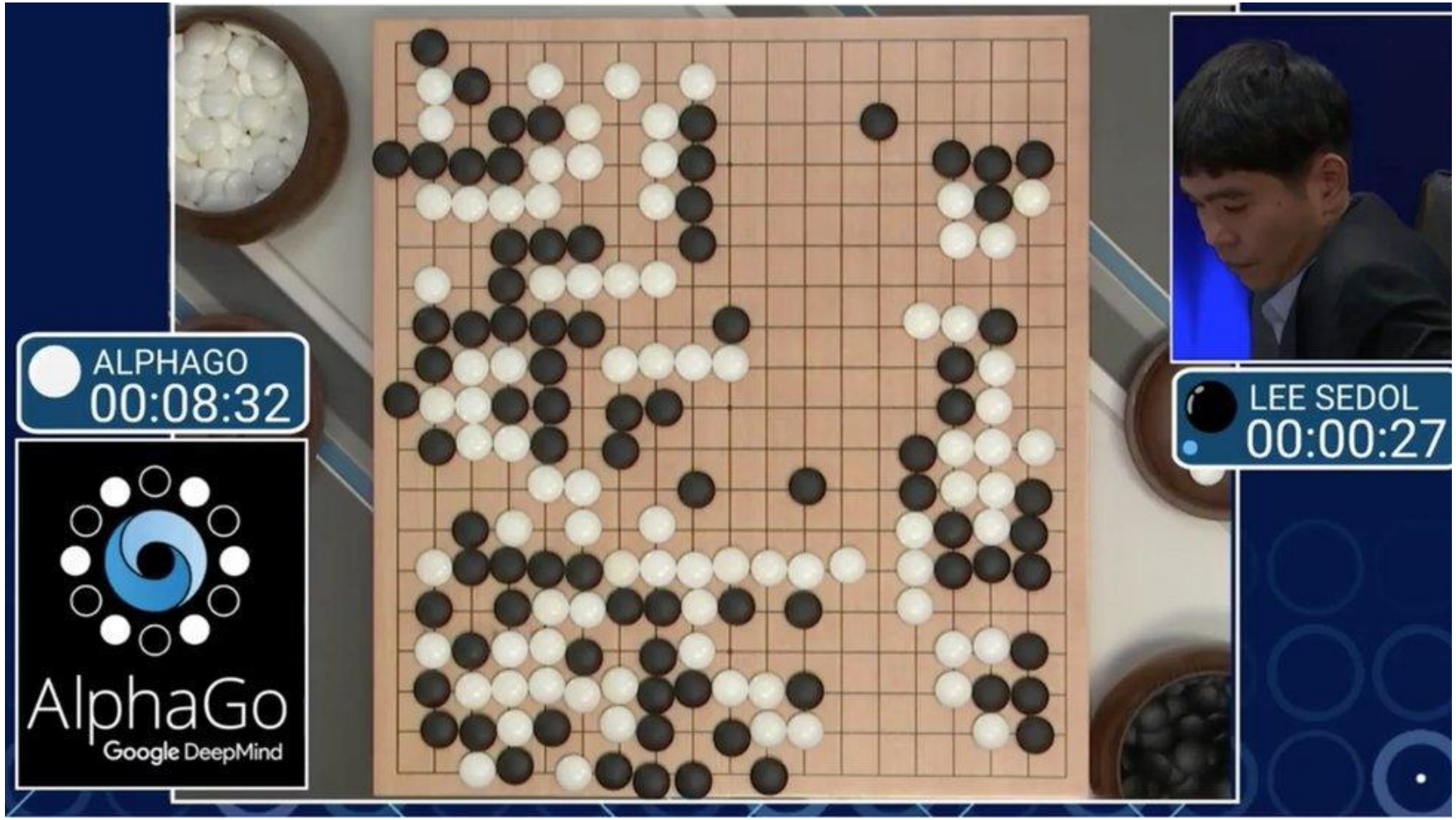
사람보다 똑똑한 AI? (2015)





https://medium.com/@weiminlu/the-scale-advantage-of-llms-reflections-on-the-bitter-lesson-d22de0ad051

전문가 보다 똑똑한 AI? (2016)



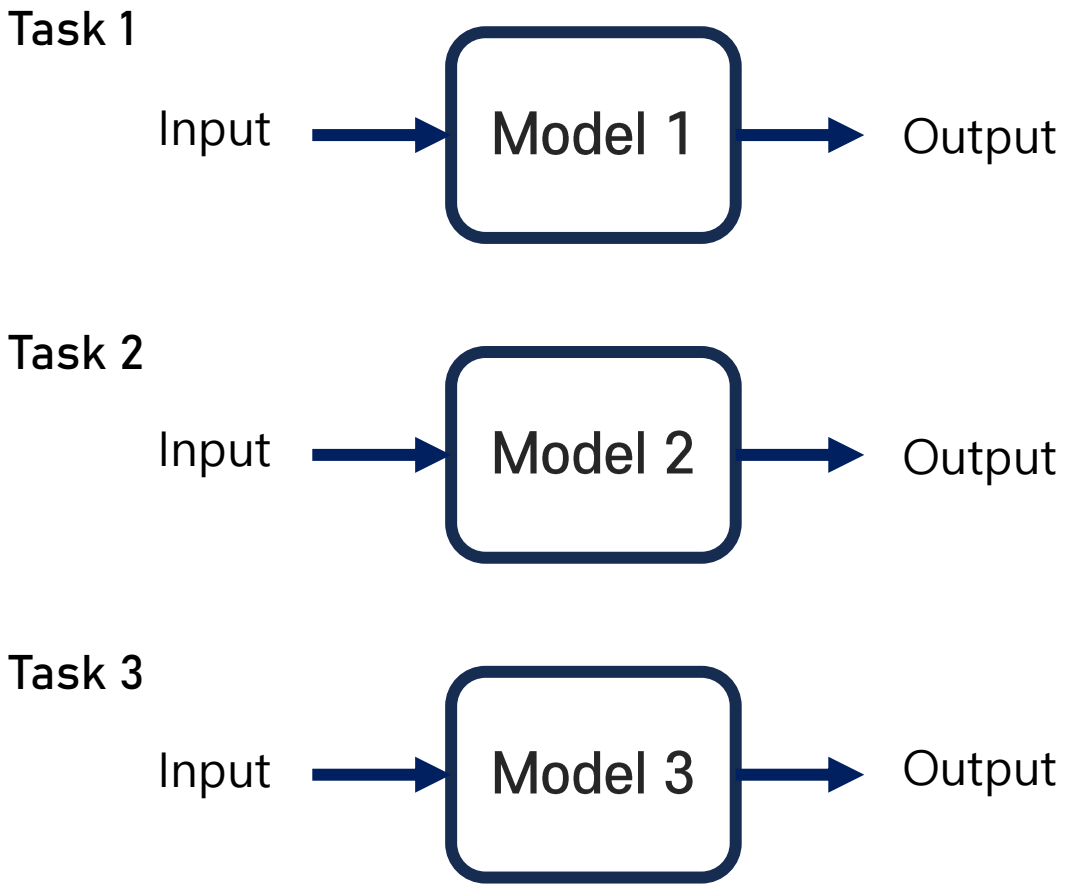
Why LLMs are so important?

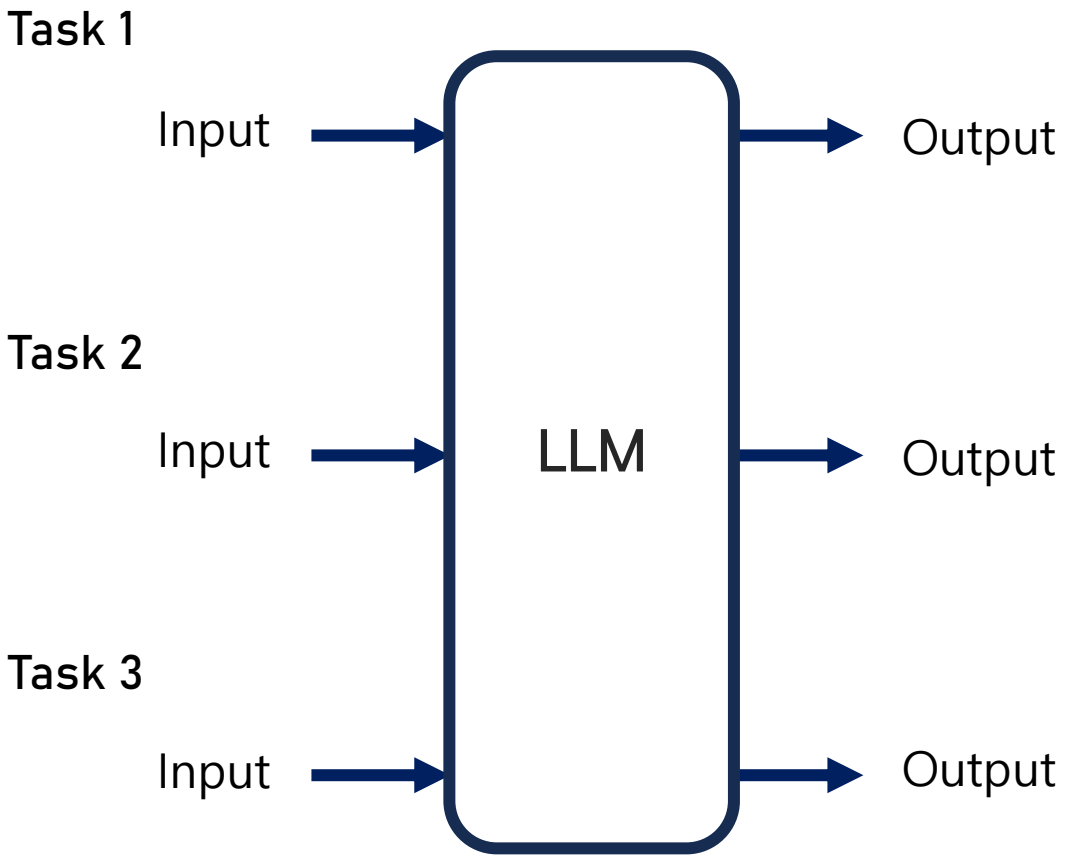
Why LLMs are so important?

- **Because it's smarter than humans**
  - The performance (accuracy) barrier was already surpassed in the mid-2010s
- **Because it can be flexibly applied to various problems, just like humans**
  - Not like traditional (TT) models
    - ResNet (2015): Train: Image classification -> Test: Image classification
    - AlphaGo (2016): Train: Go -> Test: Go
- **AI Projects before LLMs**
  - High costs for data construction (outsourcing is also difficult)
  - High cost for training
  - If the setup changes, a completely new project retraining is required (e.g., adding a new class in image classification)
- **Projects Using LLMs:**
  - Can develop prototypes with low development costs
  - Can flexibly adapt to changing environments

LLM이 기존의 NLP 모델들에 비해 가장 뛰어난 점?

이전에 경험 해보지 못한 일반화 성능으로 인간의 지능에 더 가까워 짐





2020년대초 인류는 **일반화**라는 벽을 넘어 지능의 다음단계로 넘어옴

Do current LLMs have human-level intelligence?

Do current LLMs have human-level intelligence?

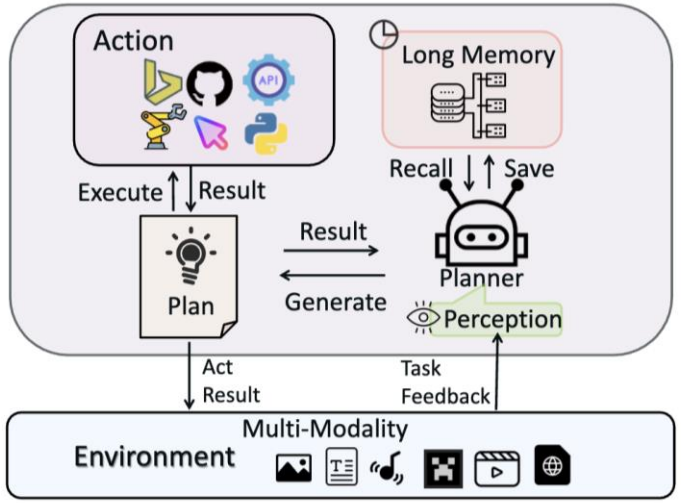
- Check the exam guidelines.
- If a photo is required, go to a photo studio and have one taken.
- Submit the application form.
- Confirm the exam location and plan your arrival to the venue on time.
- Go to the exam site.
- Identify your designated examination room.
- Follow the proctor's instructions, such as presenting your ID and verifying the test booklet.
- Read each question carefully.
- Solve the problems (**predict the correct answer** for each).
- Use the provided pen to mark your answers on the answer sheet.

The next step of AI

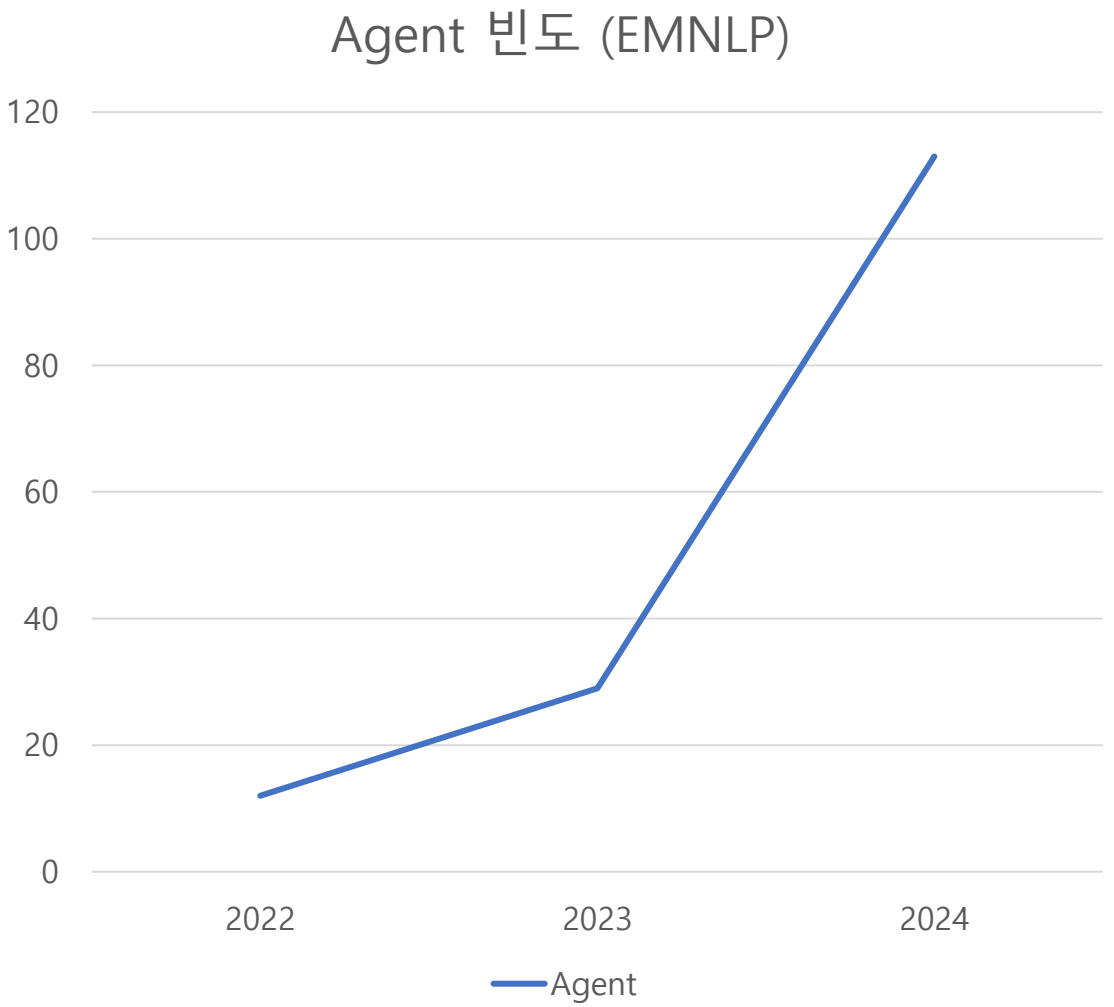
AI agent: Perception + Planning + Execution

AI Agent

- Perception + Planning + Execution



(d)



https://arxiv.org/pdf/2402.15116

Natural Language Processing

- LLM architectures
- Basics
  - Q-learning
  - Policy gradient
- Training LLMs
  - Pretraining
  - Instruction fine-tuning
  - Efficient fine-tuning
  - RLHF
- Advanced topics
  - Retrieval augmented generation
  - AI agents

RNN

BERT, GPT1

Pretraining & Finetuning

GPT2, GPT3, Llama
BART, T5
Text generation via
Next token prediction
Text infilling

FLAN-T5
Llama2
Instruction finetuning

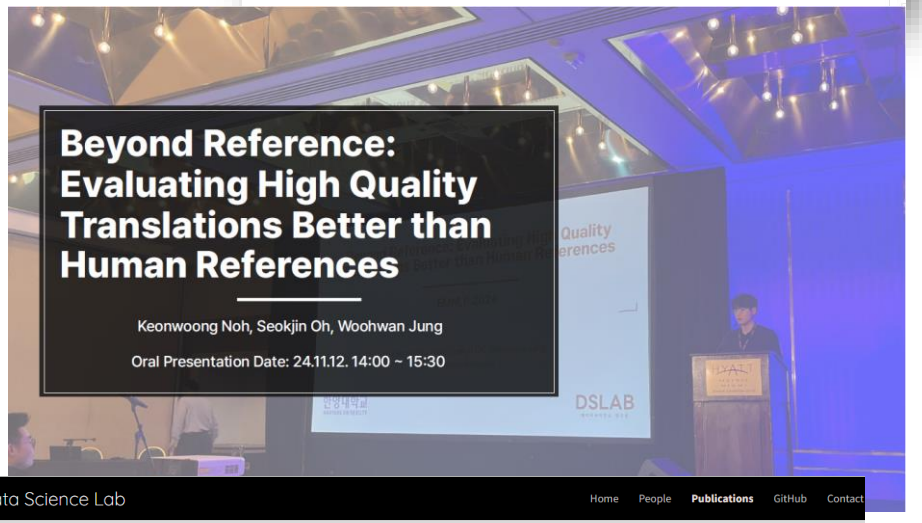
ChatGPT (~2023)
Llama3
RLHF

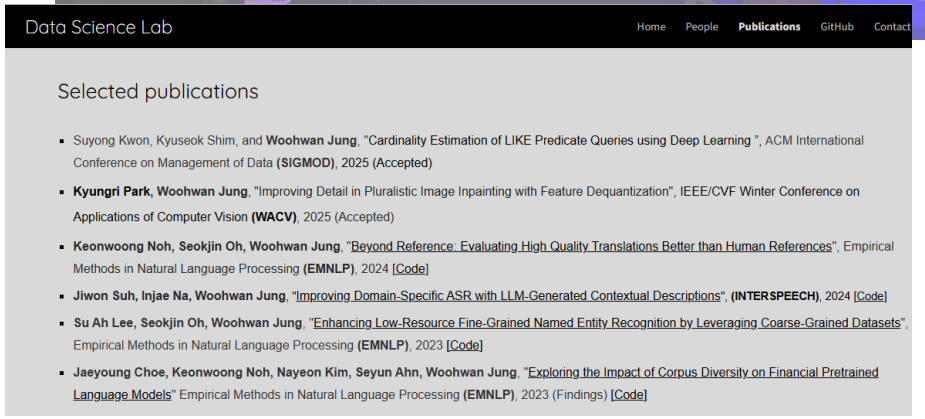
ChatGPT (2024~)

Retrieval augmented generation

Join our DSLAB - Undergraduate internship

- We are looking for highly motivated students
- Application deadline: December 20th
- To apply, send an email to Prof. Jung (whjung@hanyang.ac.kr)
  - Your email should include a brief introduction or your CV
- Internship will start from January 6th
- What interns do
  - Participating in lab seminars to expand your knowledge
  - Engaging in research projects
    - Writing papers
    - Presenting papers
    - …
    - Do what you want to do with Deep Learning!
- Research topics
  - Natural language processing
  - Computer vision
  - Multimodal AI





Thank you
Good luck on your finals
Have a happy winter vacation!