A Crash Course for Neural NLP using Pretrained LLMs

Prachya Boonkwan (Arm)

Sirindhorn International Institute of Technology
Thammasat University

prachya@siit.tu.ac.th, kaamanita@gmail.com

Slides URL ⇒ https://tinyurl.com/3rde5dnm

Who? Me?

- Nickname: **Arm** (P'/N' Arm, etc.)
- Born: Aug 1981
- Work
  - Researcher at NECTEC 2005-2024
  - Lecturer at SIIT, Thammasat University 2025-now
- Education
  - B.Eng & M.Eng, CPE Kasetsart University
  - Obtained Ministry of Science Scholarship in early 2008
  - Did a PhD in Informatics (AI & Computational Linguistics) at University of Edinburgh, UK from 2008 to 2013 (4.5 years)

Outline

- Introduction
- Zero-shot learning
- Ready-made models and pipelines
- Few-shot learning
- Retrieval-augmented generation
- OpenThaiGPT

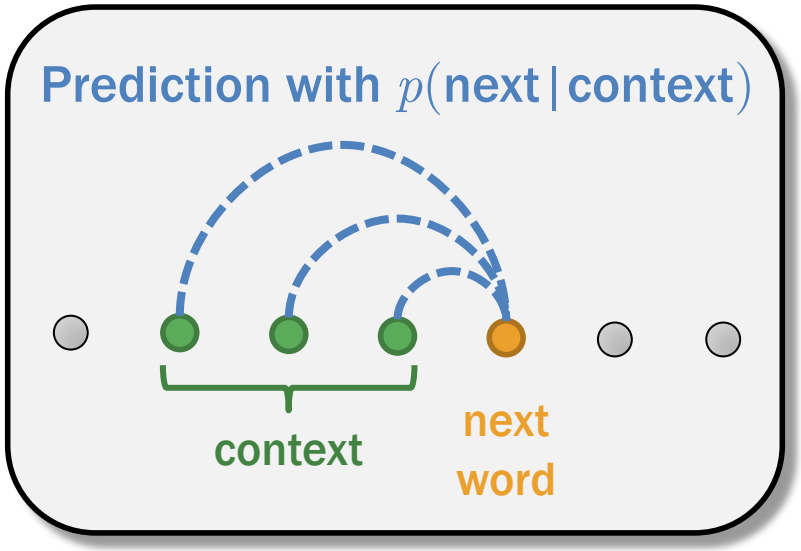
1. Introduction

Large Language Model (LLM)

- Statistical prediction for how strings are produced in a language
- Interpreted as a generative model
  1. Generate the first word w_1
  2. Keep generating the **next word** w_k based on the previous words (a.k.a. **context**) w_1 ... w_{k-1} until the whole sentence of length N is produced

$$P(w_1 \dots w_N) = p(w_1) \prod_{k=2}^{N} p(w_k | w_1 \dots w_{k-1})$$

next word: w_k — context: $w_1 \dots w_{k-1}$



We need at least 1 billion words of text data to train a stable LLM, which learns **word collocations** and **phrase structures**

What is GPT?

อรอุมา สิงห์ศิริ

น้อง	เป็น	สาว	_____	___	___	___	__	_____	_____	___	____	_____

What is GPT?

อรอุมา สิงห์ศิริ

น้อง	เป็น	สาว	ขอนแก่น	___	___	___	__	_____	_____	___	____	_____

What is GPT?

อรอุมา สิงห์ศิริ

น้อง	เป็น	สาว	ขอนแก่น	ยัง	___	___	__	_____	_____	___	____	_____

What is GPT?

อรอุมา สิงห์ศิริ

น้อง	เป็น	สาว	ขอนแก่น	ยัง	บ่	___	__	_____	_____	___	____	_____

What is GPT?

อรอุมา สิงห์ศิริ

น้อง	เป็น	สาว	ขอนแก่น	ยัง	บ่	เคย	__	_____	_____	___	____	_____

What is GPT?

เพลง: สาวอีสานรอรัก

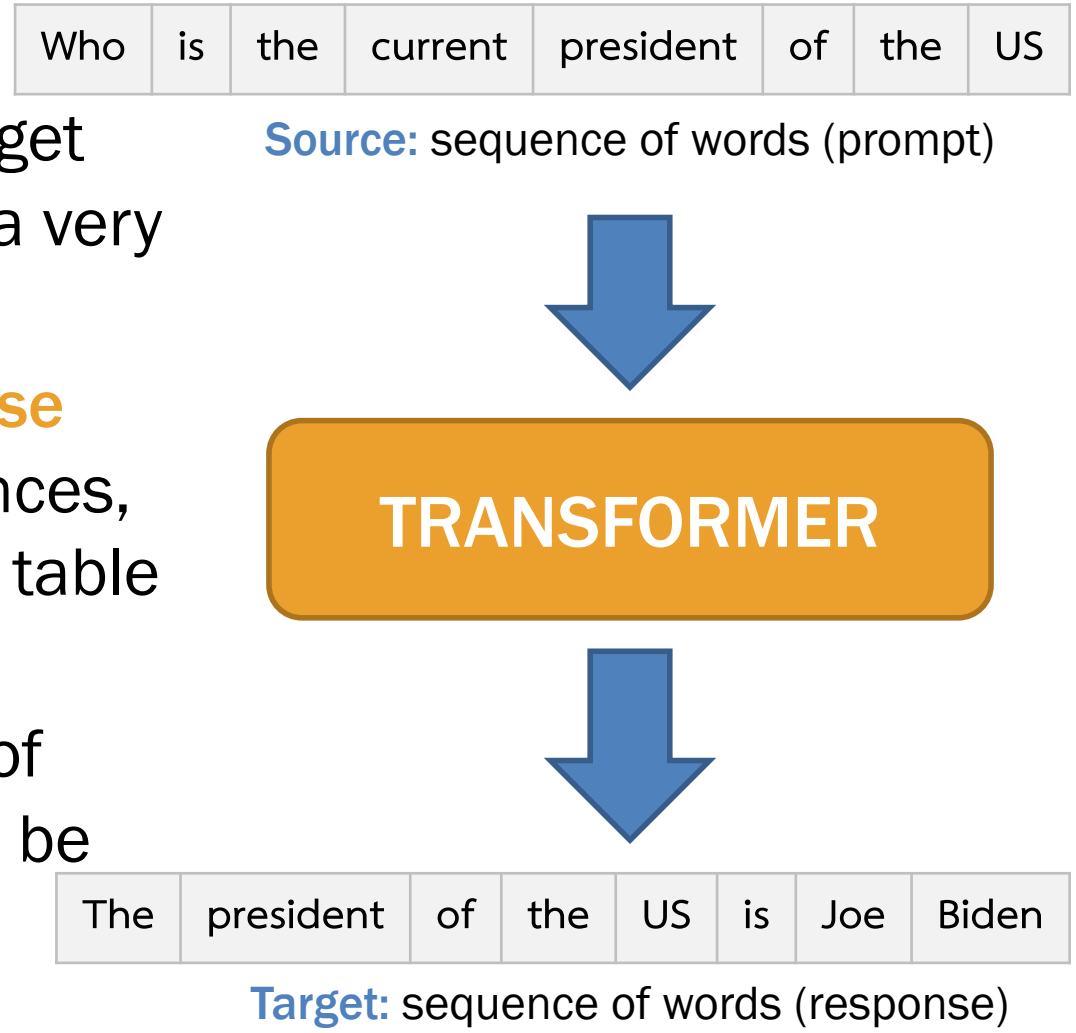
อรอุมา สิงห์ศิริ

น้อง	เป็น	สาว	ขอนแก่น	ยัง	บ่	เคย	มี	แฟน	บ้าน	อยู่	แดน	อีสาน

https://www.youtube.com/watch?v=stLQVIau1ns

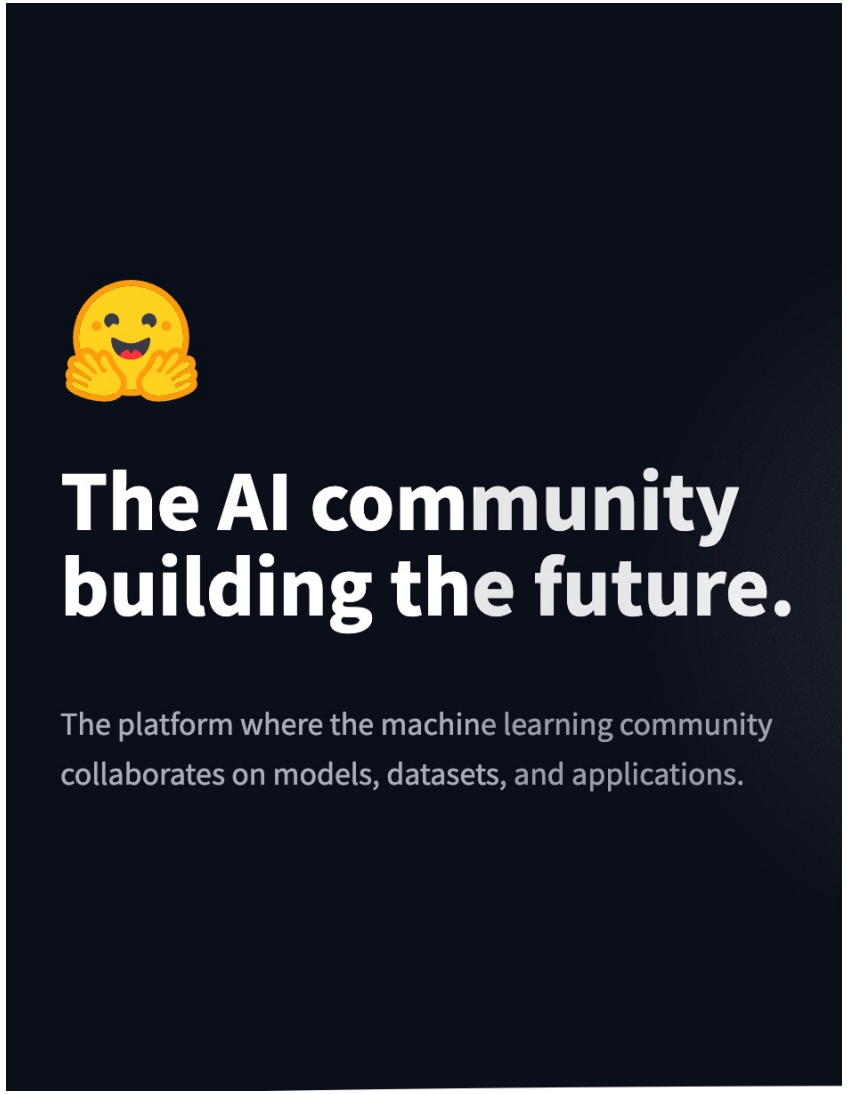
Transformer Model (Vaswani et al., 2016)

- Sequence-to-sequence generation
  - **Translation:** It learns how to produce a target sequence from a source sequence, given a very large dataset of sequence pairs
  - **Pros:** It learns **word collocations** and **phrase structures** on the input and output sequences, and associates them cross-lingually in the table of **translation alignments**
  - **Cons:** It consists of an expansive amount of neuron cells, and the training process can be quite time-consuming



Hugging Face Library

- A compendium of Transformer-based pretrained models, pretrained text tokenizers, and prepared datasets
  - Providing an end-to-end pipeline from raw texts to final outputs according to various NLP tasks
  - Offering a platform where developers can share their models, text tokenizers, and datasets to the HuggingFace community with millions of users worldwide



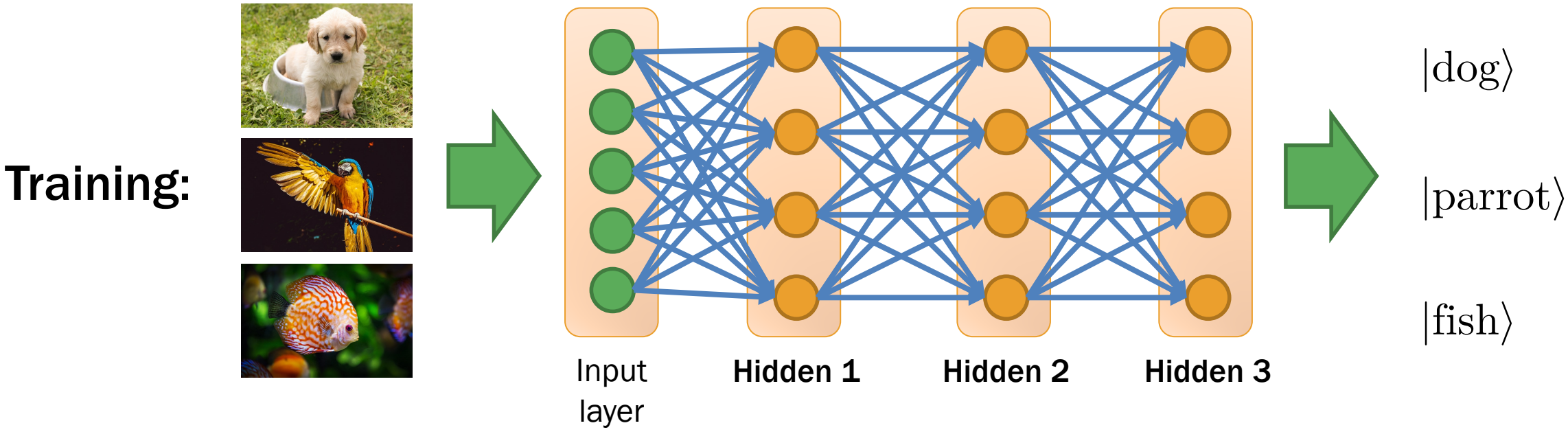
Google Colab

https://tinyurl.com/4j7jsf9t

2. Zero-Shot Learning

Zero-Shot Learning

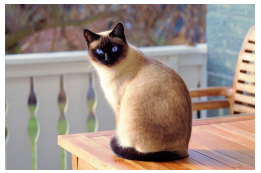
- Classifying objects to previously unseen classes, without receiving any specific training for those classes
- Zero shots = zero examples

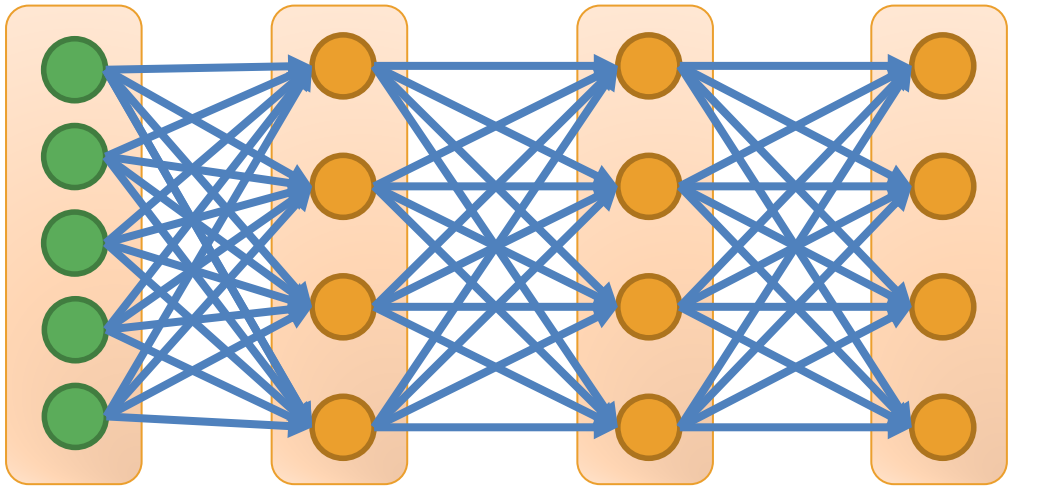


Zero-Shot Learning

- Classifying objects to previously unseen classes, without receiving any specific training for those classes
- Zero shots = zero examples

Testing:





$|\text{cat}\rangle$

(unseen class)

Zero-Shot Learning with LLM

```
[ ]
# We use a pretrained multilingual DeBERTa with the capability of natural language inference.
model = pipeline(
    task='zero-shot-classification',
    model='MoritzLaurer/mDeBERTa-v3-base-mnli-xnli',
    device=get_device()
)

# This is the input text. Note that it can be untokenized!
text = "หมากปริญเผยไม่เคยปิดบังว่ามีแฟนแล้ว"

# This is the label set.
labels = ["politics", "economy", "entertainment", "environment"]

# Classify the text into one of these labels.
output = model(text, labels, multi_label=False)

# Print out the result.
print(f'Input : {text}')
print(f'Output: {output}')
print(f'Most probable class is: {output["labels"][0]} with probability {output["scores"][0]}.')
```

```
Input : หมากปริญเผยไม่เคยปิดบังว่ามีแฟนแล้ว
Output: {'sequence': 'หมากปริญเผยไม่เคยปิดบังว่ามีแฟนแล้ว', 'labels': ['entertainment', 'politics', 'econom
Most probable class is: entertainment with probability 0.5786380767822266.
```

Zero-Shot Classification

```
[ ]
```

```python
# We use a pretrained DeBERTa with the capability of mask filling.
model = pipeline(
    task='fill-mask',
    model='MilaNLProc/deberta-v3-large-mlm-reddit-gab',
    device=get_device()
)

# Input text
text = 'Mary has lost her beloved dog, and her eyes are now tearful.'

# The mask is `model.tokenizer.mask_token`.
question = 'The emotion of this text is: ' + model.tokenizer.mask_token + ' .'

# Prompt
prompt = f'"{text}" {question}'
print(f'Prompt: {prompt}')

# The input text and the prompt are put together.
output = model(prompt)

# Print out the result.
for idx, output_entry in enumerate(output):
    print(f'Output {idx} = {output_entry}')
```

```
Prompt: "Mary has lost her beloved dog, and her eyes are now tearful." The emotion of this text is: [MASK] .
Output 0 = {'score': 0.07433608919382095, 'token': 12819, 'token_str': 'sadness', 'sequence': '"Mary has lost
Output 1 = {'score': 0.07278111577033997, 'token': 26817, 'token_str': 'disgust', 'sequence': '"Mary has lost
Output 2 = {'score': 0.02848154865205288, 'token': 5530, 'token_str': 'tears', 'sequence': '"Mary has lost he
Output 3 = {'score': 0.01766805909574032, 'token': 17642, 'token_str': 'sorrow', 'sequence': '"Mary has lost
Output 4 = {'score': 0.014488041400909424, 'token': 10511, 'token_str': 'disappointment', 'sequence': '"Mary
```

3. Ready-Made Models and Pipelines

Ready-Made Models

NLP

Tasks	Descriptions	Examples
`conversational`	Conversational agent	PygmalionAI
`fill-mask`	Mask filling for zero-shot learning	BERT
`question-answering`	Question answering (extractive)	RoBERTa
`table-question-answering`	Question answering from a given table	Tapas
`text2text-generation`	Text generation from a given text	Parrot Paraphraser
`text-classification`	Document classification and sentiment analysis	DistillBERT
`sentiment-analysis`	(same as `text-classification`)	
`text-generation`	Text generation	GPT-2
`token-classification`	Sequence prediction (e.g. word segmentation, POS tagging, and NER)	BERT
`translation`	Machine translation	T5
`translation_xx_to_yy`	Machine translation from XX to YY (language codes must be small letters)	T5
`summarization`	Automatic text summarization	BART

Speech Processing

Tasks	Descriptions	Examples
`audio-classification`	Audio classification	XLSR
`automatic-speech-recognition`	Automatic speech recognition	Wav2Vec
`zero-shot-audio-classification`	Zero-shot audio classification	CLIP-HTSAT

Image Processing

Tasks	Descriptions	Examples
`depth-estimation`	Depth estimation	GLPN
`feature-extraction`	Image feature extraction for few-shot learning	DINO Vision Transformer
`image-classification`	Image classification	ResNet-50
`image-segmentation`	Image segmentation	ClipSeg
`mask-generation`	Generation of object masks in image processing	SAM Vision Transformer
`object-detection`	Object detection	YOLOS
`video-classification`	Video classification	XClip
`zero-shot-classification`	Zero-shot classification	BART
`zero-shot-image-classification`	Zero-shot image classification	CLIP Vision Transformer
`zero-shot-object-detection`	Zero-shot object detection	OWL Vision Transformer

Multimodal Processing

Tasks	Descriptions	Examples
`document-question-answering`	Question answering from scanned documents	LayoutLM
`image-to-text`	Caption generation	Vision Transformer + GPT2
`visual-question-answering`	Visual question answering	ViLT

Example: Sentiment Analysis

```
[ ]
# We let the pipeline choose the best model for sentiment analysis.
model = pipeline(
    task='sentiment-analysis',
    device=get_device()
)

# This is the input text.
text = 'I am promoted as the managing director! Let\'s celebrate tonight.'

# Let's run the model.
output = model(text)

# Here's the result.
print(f'Input : {text}')
print(f'Output: {output}')
```

```
No model was supplied, defaulted to distilbert-base-uncased-finetuned-sst-2-engli
Using a pipeline without specifying a model name and revision in production is no
Downloading (…)lve/main/config.json: 100% 629/629 [00:00<00:0
Downloading model.safetensors: 100% 268M/268M [00:04<00:00
Downloading (…)okenizer_config.json: 100% 48.0/48.0 [00:00<00
Downloading (…)solve/main/vocab.txt: 100% 232k/232k [00:00<0
Input : I am promoted as the managing director! Let's celebrate tonight.
Output: [{'label': 'POSITIVE', 'score': 0.9998561143875122}]
```

4. Few-Shot Learning

Few-Shot Learning

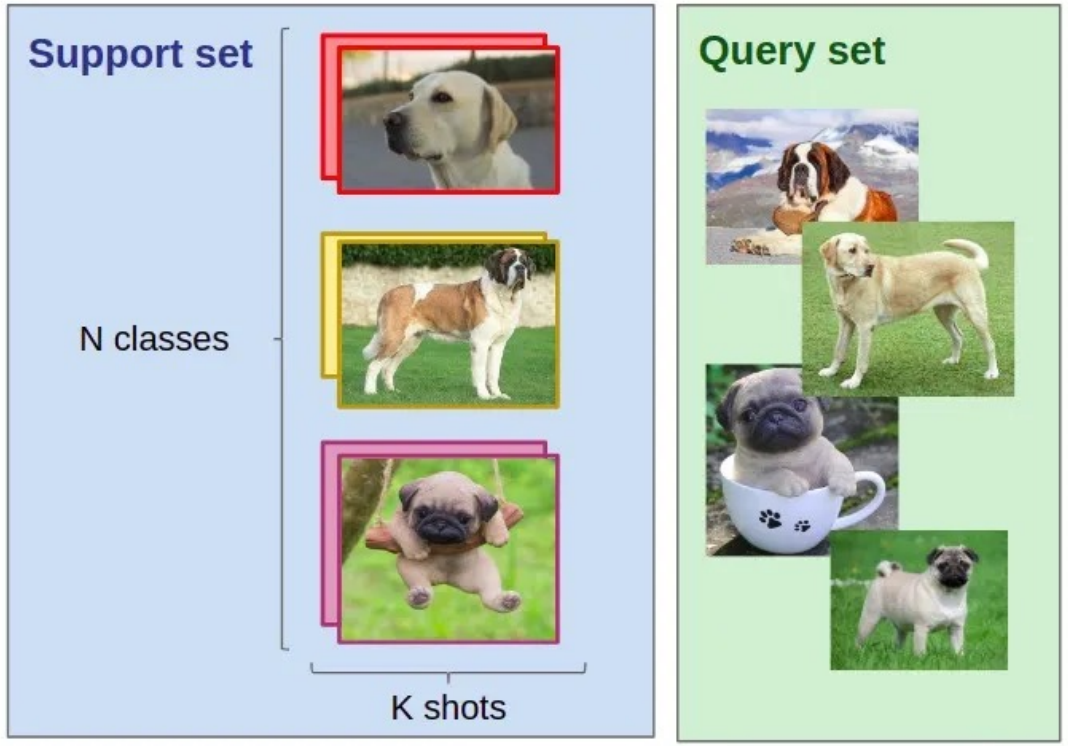
- Learn how to classify objects from previously unseen classes w.r.t. a small example set
- Few shots = few examples

Meta Learning: learning how to build a model

We train a **similarity model** sim($\mathbf{x}$,$\mathbf{x}'$), where $\mathbf{x}$ and $\mathbf{x}'$ are two examples.

- If $\mathbf{x}$ and $\mathbf{x}'$ belong to the same groups, sim($\mathbf{x}$,$\mathbf{x}'$) should be 1.
- If $\mathbf{x}$ and $\mathbf{x}'$ belong to different groups, sim($\mathbf{x}$,$\mathbf{x}'$) should be 0.

The training with backpropagation is done based on the above constraints until convergence. Contrastive learning is employed for training the similarity model, where we use more negative samples with $\mathbf{sim}(\mathbf{x},\mathbf{x}')=0$ than positive samples with $\mathbf{sim}(\mathbf{x},\mathbf{x}')=1$.



Few-Shot Learning with LLM

```python
# dataset = load_dataset('StatsGary/socialmedia-abuse')
dataset = load_dataset('tweet_eval', 'stance_atheism')
```

```python
dataset['train'].to_pandas()
```

	text	label
0	@user Bless Almighty God, Almighty Holy Spirit...	1
1	Take away hatred from some people, and you hav...	1
2	I took my troubles to the Lord: I cried out to...	1
3	You can't think by yourself about life and bel...	2
4	RT @user Humanist love to everyone at #100AEUA...	0
...	...	...
456	Pain and setbacks are a part of life. Stay in ...	1
457	Light is not called into Light, its called int...	1
458	Now that the SCOC has ruled Canadians have fre...	2
459	Indigenous would lose more than Madhesi if da ...	0
460	If children weren't religiously indoctrinated ...	1

461 rows × 2 columns

Few-Shot Learning with LLM

```python
# model = SetFitModel.from_pretrained('all-mpnet-base-v2', num_labels=2)
model = SetFitModel.from_pretrained('all-mpnet-base-v2', num_labels=3)
model.to(get_device())
```

```python
trainer = SetFitTrainer(
    model=model,                        # Sentence transformer
    train_dataset=train_ds,             # Training set
    eval_dataset=eval_ds,               # Test set
    loss_class=CosineSimilarityLoss,    # Similarity measure
    batch_size=16,
    num_iterations=20,                  # Negative samples for contrastive learning
    column_mapping={"text": "text", "label": "label"},
    num_epochs=2,
    learning_rate=2e-5
)
```

```python
trainer.train()     # This step takes 11 minutes on CPU-only training.
```

```
Applying column mapping to training dataset
Generating Training Pairs: 100% 20/20 [00:00<00:00, 340.02it/s]
***** Running training *****
  Num examples = 960
  Num epochs = 2
  Total optimization steps = 120
  Total train batch size = 16
Epoch: 100% 2/2 [28:09<00:00, 844.43s/it]
Iteration: 100% 60/60 [14:08<00:00, 13.67s/it]
Iteration: 100% 60/60 [14:01<00:00, 13.76s/it]
```

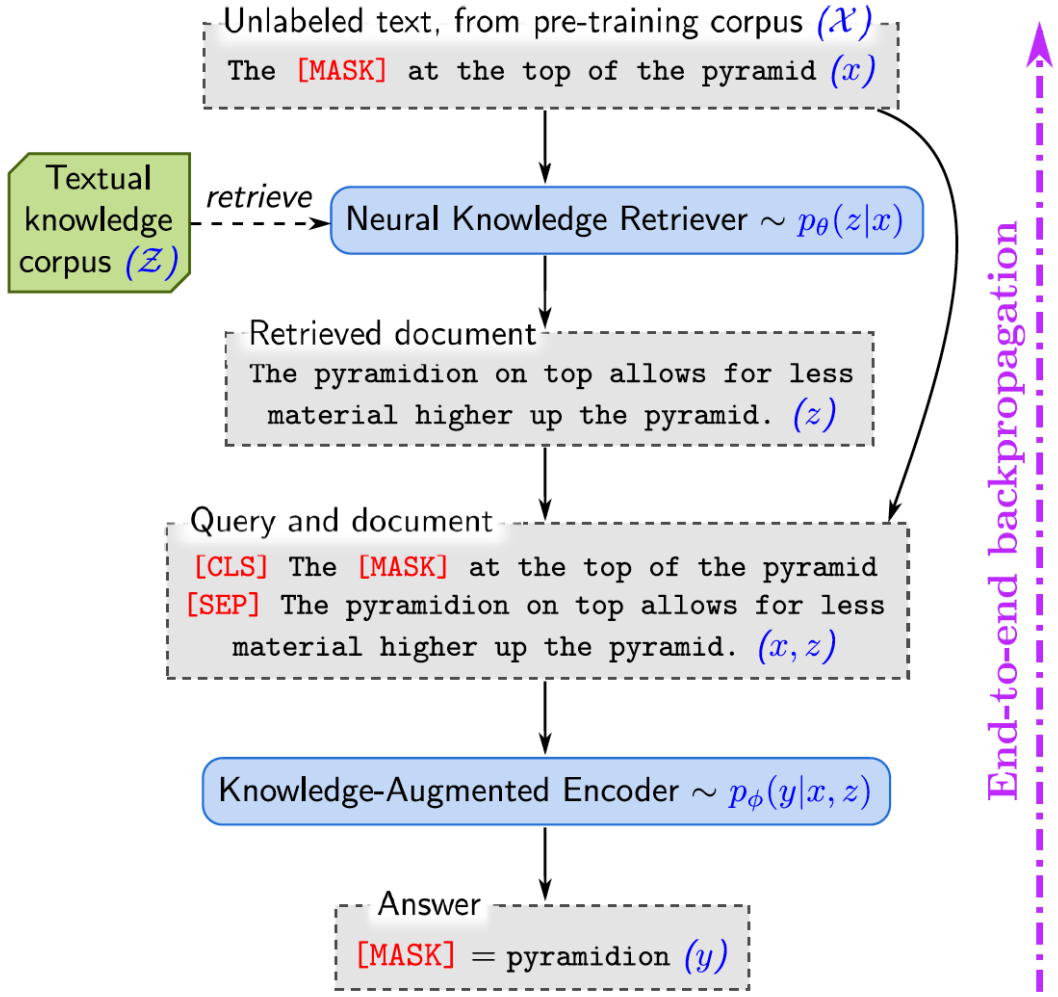
```python
texts = [
    'Look at that fat chick! Ewwww.',
    'She looks lovely in blonde hair.',
    'These dumbers deserve a low life.'
]

model.predict(texts)
```

```
tensor([2, 1, 2])
```

5. Retrieval-Augmented Generation

Retrieval-Augmented Generation

- Enhancing question answering by enriching the query with relevant info
  - Mitigating the hallucination issue in traditional LLMs
  - **In-context learning:** simply attaching the query with relevant documents retrieved with information retrieval
  - Vector databases (e.g. FAISS and VectorDB) are used to store internal documents on on-premise servers
  - Advantageous for content access control



RAG in Practical Uses

- Dify
  - Low-code RAG system (recommended)
  - https://dify.ai/
- LangChain
  - Generic customizable framework for RAG
  - https://www.langchain.com/
- LocalGPT
  - Quite easy to get started and user-friendly
  - https://github.com/PromtEngineer/localGPT

6. OpenThaiGPT

OpenThaiGPT

Try me at https://openthaigpt.openservice.in.th

NECTEC
a member of NSTDA

Thai government agency

AiAT
ARTIFICIAL INTELLIGENCE ASSOCIATION OF THAILAND

EAT
สมาคมผู้ประกอบการปัญญาประดิษฐ์ประเทศไทย
Artificial Intelligence Entrepreneur Association of Thailand

Training Datasets

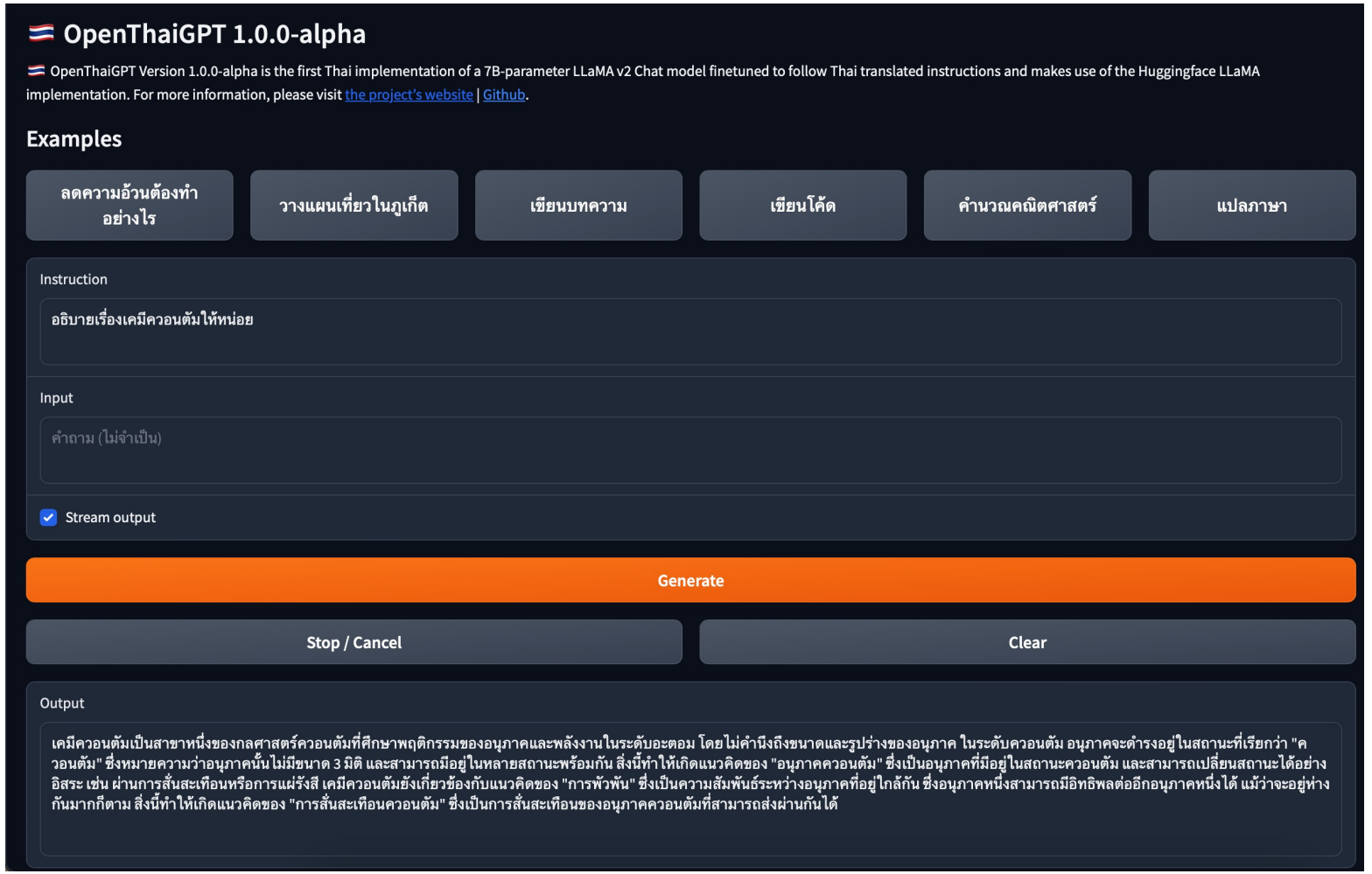
Pretraining datasets

Name	Source Type	Access Type	token size (GPT Thai)	size (GB)
mC4	Web Crawl	Public	16B	56 GB
OSCAR 2019	Web Crawl	Public	3.4B	16GB
OSCAR 2021	Web Crawl	Public	3.8B	16GB
OSCAR 2022	Web Crawl	Public	10.8B	60GB
OSCAR 2023	Web Crawl	Public, Access needed	17B	100GB
CC100	Web Crawl	Public	12B	80GB
LST20	News	Public, Access needed	3M (Compound Word)	
Twitter	Social	P Conan, Private	560M	
Linetoday	News	P Conan, Private	105M	
TraffyFondue	Complaint	P Conan, Private	5M	
Wikipedia	Wiki	Public	10M	
prachathai67k	News	Public	160M	
ThaiPBS	News	Public		
Scb-th-en(extract th)	MISC	Public	50.3M	
wisesight_sentiment	Social	Public	0.978M	
Wongnai_reviews	Social	Public	11.1M	
ThaiRath		Public		
Best		Public, Access needed		

- OpenThaiGPT was trained on the dataset consisting of over 2 trillion tokens (sub-words): CC100, OSCAR, and mC4
- After cleansing and anonymization, we obtained 37.3 billion tokens
- Pantip.com supplied additional 20.0 billion tokens of social media datasets
- In total, we obtained 60 billion tokens for training

OpenThaiGPT 1.0.0

https://openthaigpt.openservice.in.th



- Thai LLaMa v.2
  - 7 billion and 13 billion parameters
  - Natural word-level generation
  - BPE tokenizer with 20,000 Thai frequent tokens

🤗 Open LLM Leaderboard

📐 The 🤗 Open LLM Leaderboard aims to track, rank and evaluate open LLMs and chatbots.

🤗 Submit a model for automated evaluation on the 🤗 GPU cluster on the "Submit" page! The leaderboard's backend runs the great Eleuther AI Language Model Evaluation Harness - read more details in the "About" page!

Archived in 14 SEP 2023

🏅 LLM Benchmark | 📝 About | 🚀 Submit here!

Select columns to show

- [x] Average ⬆️
- [x] ARC
- [x] HellaSwag
- [x] MMLU
- [x] TruthfulQA
- [ ] Type
- [ ] Precision
- [ ] Hub License
- [ ] #Params (B)
- [ ] Hub ❤️
- [ ] Model sha

- [x] Show gated/private/deleted models

thai

Model types

- [x] 🟢 pretrained
- [x] 🔶 fine-tuned
- [x] ⭕ instruction-tuned
- [x] 🟦 RL-tuned

Model sizes

- [x] Unknown
- [x] < 1.5B
- [x] ~3B
- [x] ~7B
- [x] ~13B
- [x] ~35B
- [x] 60B+

T	Model	Average ⬆️	ARC	HellaSwag	MMLU	TruthfulQA
⭕	openthaigpt/openthaigpt-1.0.0-alpha-7b-chat-ckpt-hf	53.25	50.85	74.89	40.02	47.23
🔶	wannaphong/openthaigpt-0.1.0-beta-full-model_for_open_llm_leaderboard	51.3	51.28	77.46	33.18	43.28
🔶	pythainlp/wangchanglm-7.5B-sft-en-sharded	38.7	34.47	59.81	26.37	34.15
🔶	pythainlp/wangchanglm-7.5B-sft-enth	38.05	33.79	58.99	24.52	34.9

LLM Project @ NECTEC

- It simulates how humans string words to become a meaningful sentence
- Trained models:
  - **LLaMa (GPT)** and **Nemo-Megatron (GPT-X)** for natural sentence generation
  - **ELECTRA** for document classification
  - **BART** for machine translation and document summarization
  - **DeBERTa** for classification and zero-shot learning
  - **Multimodality:** text, speech, images, videos
- Making Thai one of the pivot languages for LLM via knowledge distillation and model grafting

7. Conclusion

Conclusion

- Large language models
- Zero-shot learning
- Ready-made models
- Few-shot learning
- RAG
- OpenThaiGPT

Thank You

prachya@siit.tu.ac.th

kaamanita@gmail.com