

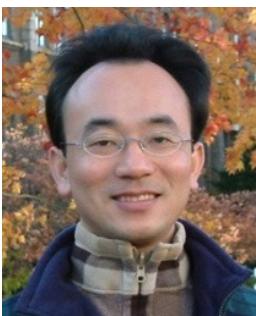
Deep Learning, Generative AI, Transfer Learning, Zero-Shot, and Few-Shot Learning for Text Analytics

1121AITA10

MBA, IM, NTPU (M5265) (Fall 2023)
Tue 2, 3, 4 (9:10-12:00) (B3F17)



<https://meet.google.com/miy-fbif-max>



Min-Yuh Day, Ph.D,
Associate Professor

Institute of Information Management, National Taipei University

<https://web.ntpu.edu.tw/~myday>



Syllabus

Week Date Subject/Topics

- 1 2023/09/13 Introduction to Artificial Intelligence for Text Analytics
- 2 2023/09/20 Foundations of Text Analytics:
Natural Language Processing (NLP)
- 3 2023/09/27 Python for Natural Language Processing
- 4 2023/10/04 Natural Language Processing with Transformers
- 5 2023/10/11 Case Study on Artificial Intelligence for Text Analytics I
- 6 2023/10/18 Text Classification and Sentiment Analysis

Syllabus

Week Date Subject/Topics

7 2023/10/25 Multilingual Named Entity Recognition (NER)

8 2023/11/01 Midterm Project Report

9 2023/11/08 Text Similarity and Clustering

10 2023/11/15 Text Summarization and Topic Models

11 2023/11/22 Text Generation with Large Language Models (LLMs)

12 2023/11/29 Case Study on Artificial Intelligence for Text Analytics II

Syllabus

Week Date Subject/Topics

13 2023/12/06 Question Answering and Dialogue Systems

14 2023/12/13 Self-learning

15 2023/12/20 Deep Learning, Generative AI, Transfer Learning,
Zero-Shot, and Few-Shot Learning for Text Analytics

16 2023/12/27 Final Project Report I

17 2024/01/03 Final Project Report II

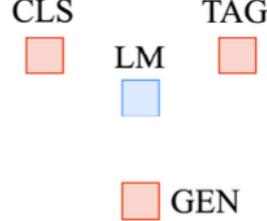
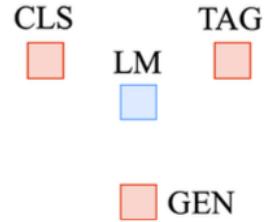
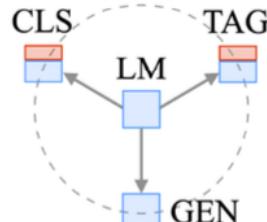
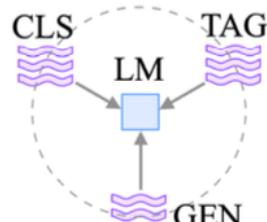
18 2024/01/10 Self-learning

**Deep Learning,
Generative AI,
Transfer Learning,
Zero-Shot, and
Few-Shot Learning
for Text Analytics**

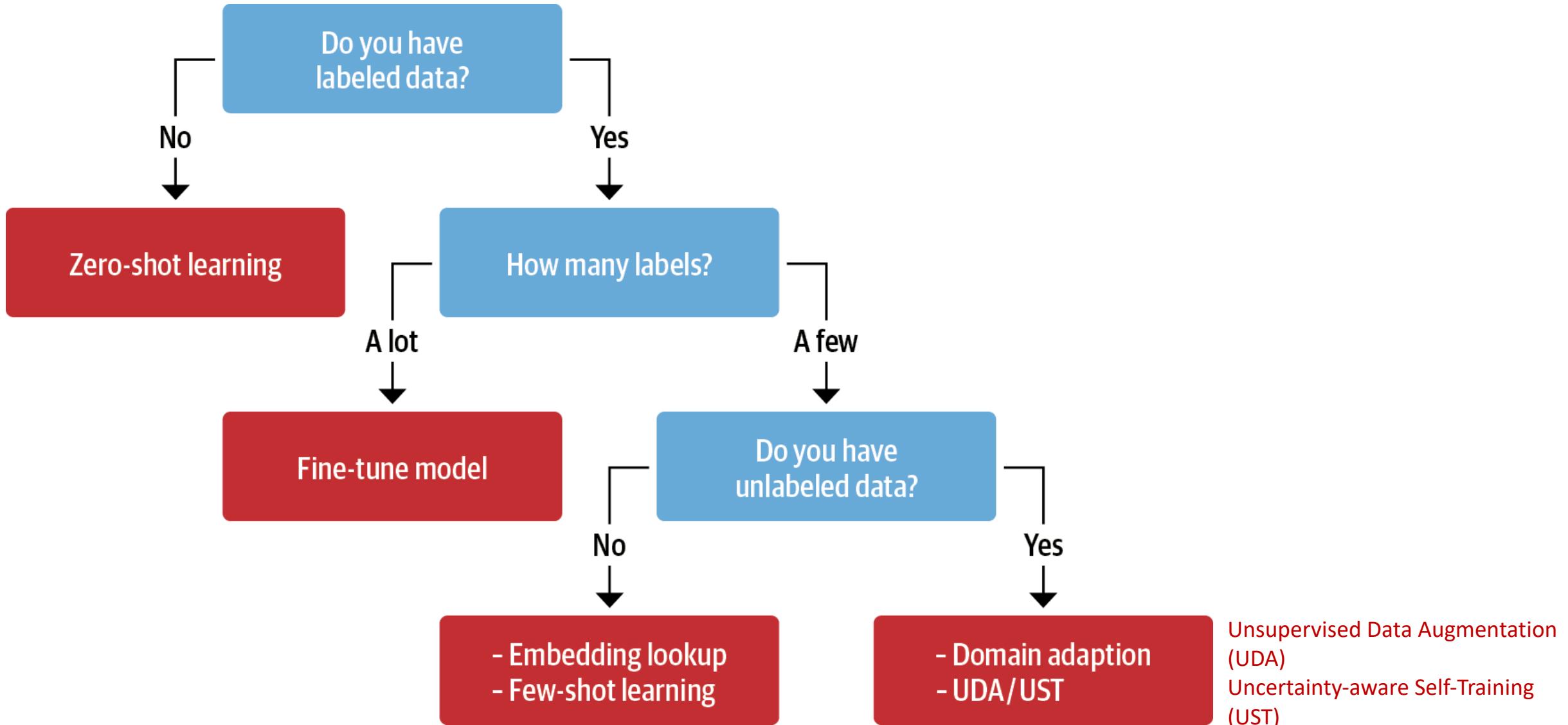
Outline

- Deep Learning
 - Generative AI
 - Pre-train, Prompt, and Predict (Prompting)
 - Transfer Learning
 - Pre-training, Fine-Tuning (FT)
- Few-Shot Learning (FSL)
 - Meta Learning: Learn to Learn
- One-Shot Learning (1SL)
- Zero-Shot Learning (OSL)(ZSL)

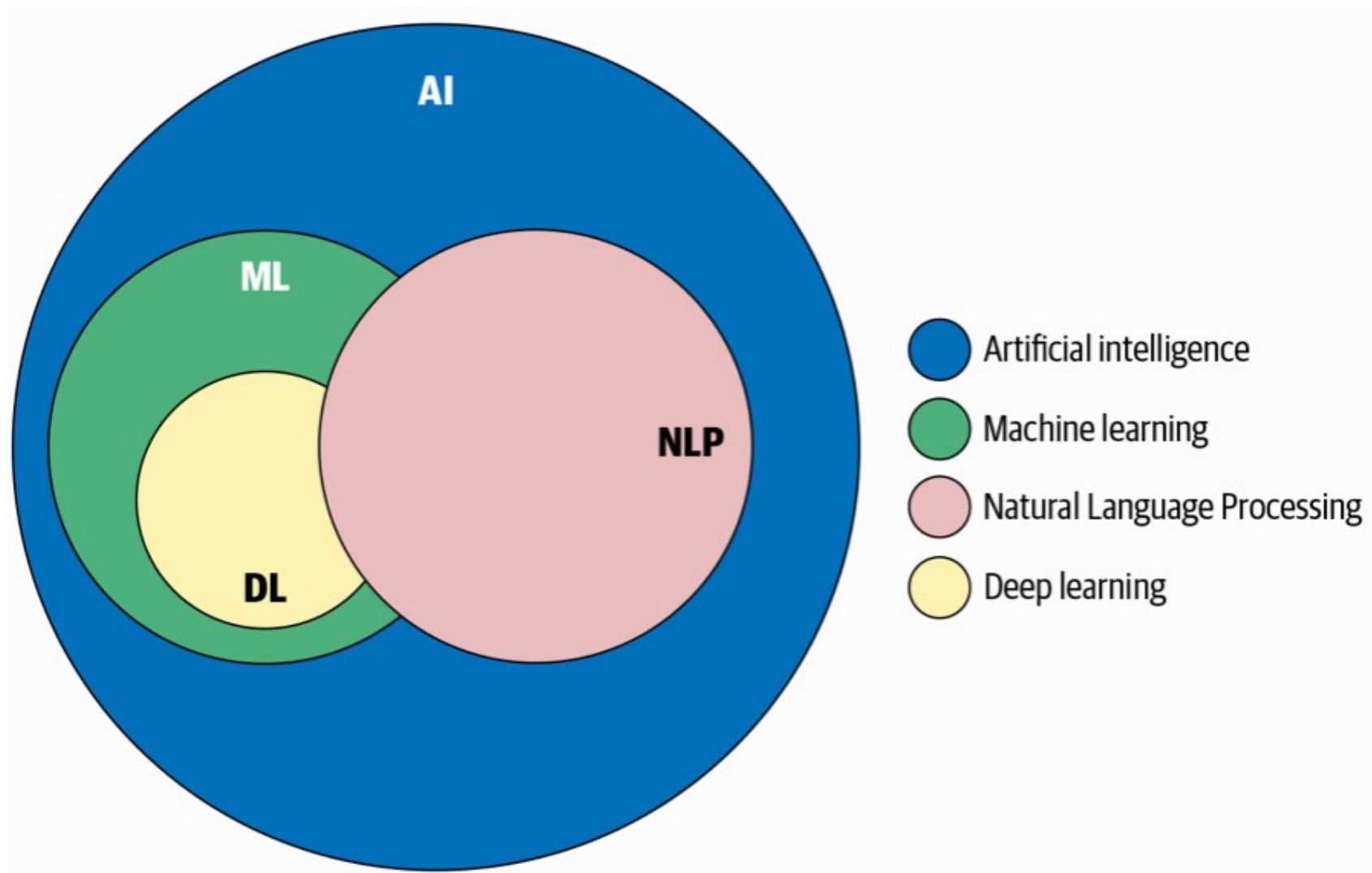
Four Paradigms in NLP (LM)

Paradigm	Engineering	Task Relation
a. Fully Supervised Learning (Non-Neural Network)	Feature (e.g. word identity, part-of-speech, sentence length)	
b. Fully Supervised Learning (Neural Network)	Architecture (e.g. convolutional, recurrent, self-attentional)	
c. Pre-train, Fine-tune	Objective (e.g. masked language modeling, next sentence prediction)	
Transfer Learning: Pre-training, Fine-Tuning (FT)		
d. Pre-train, Prompt, Predict	Prompt (e.g. cloze, prefix)	

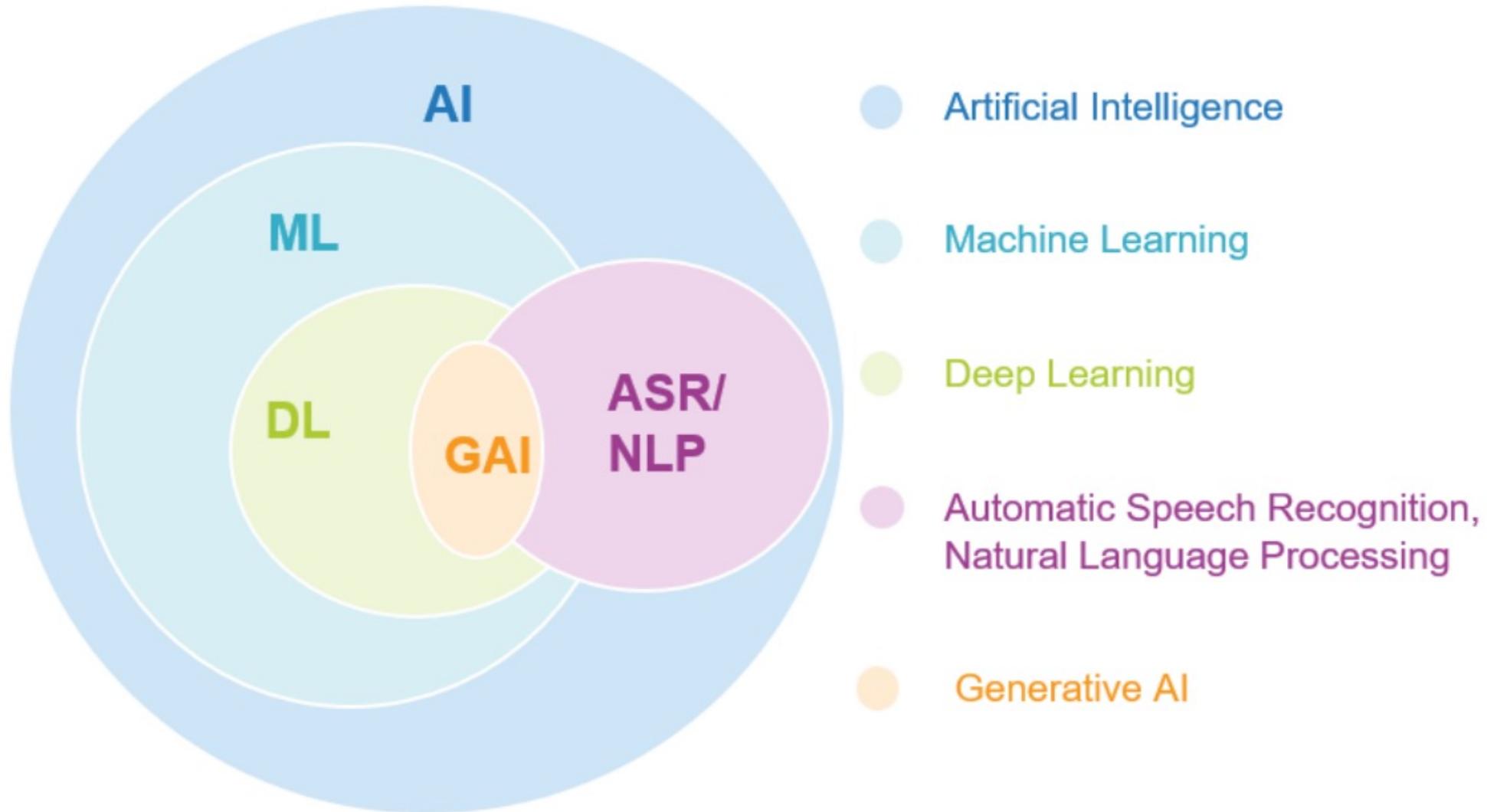
Transfer Learning, Fine-tuning, Few-shot learning



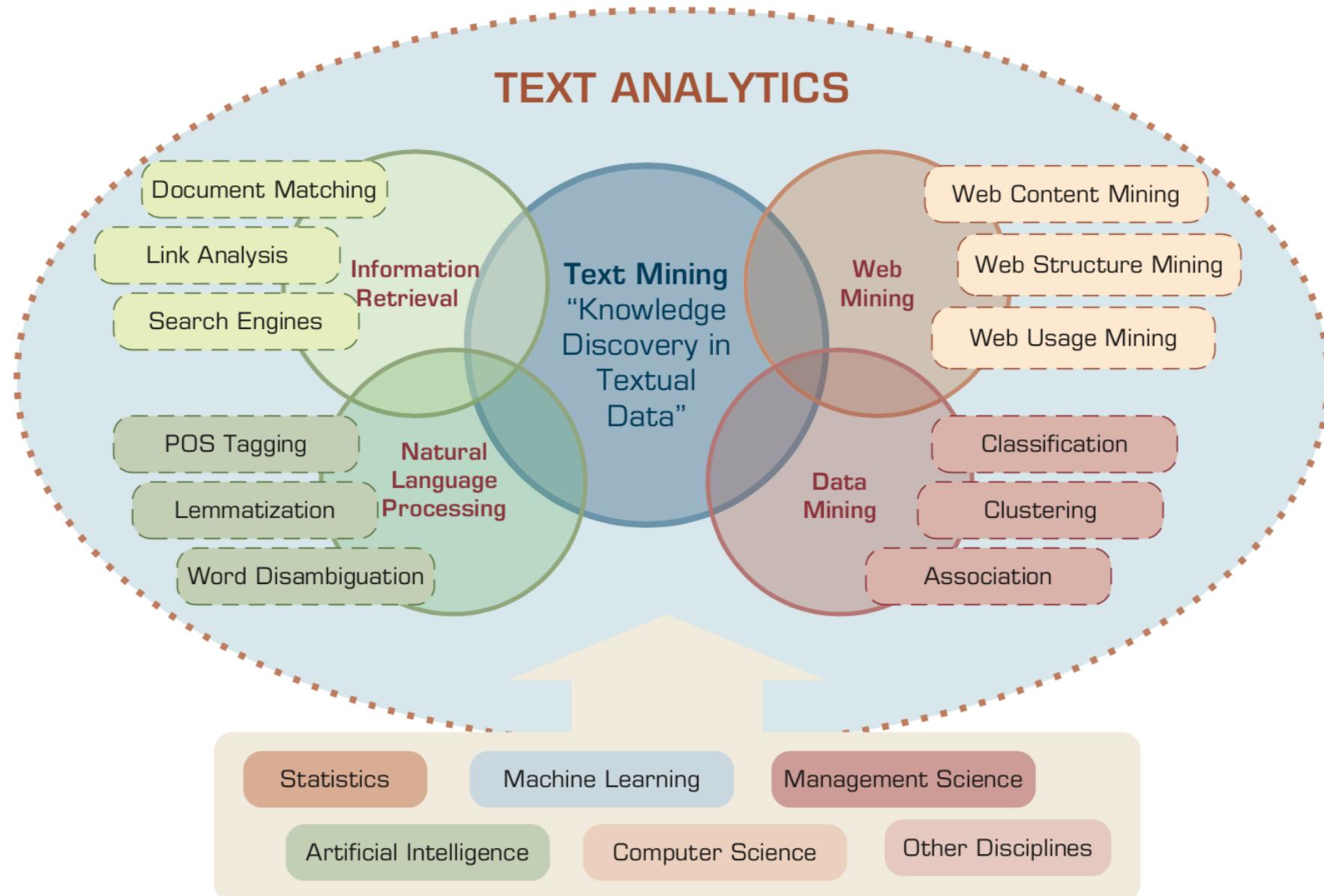
AI, NLP, ML, DL



Generative AI



Text Analytics and Text Mining



AI, ML, DL

Artificial Intelligence (AI)

Machine Learning (ML)

Supervised
Learning

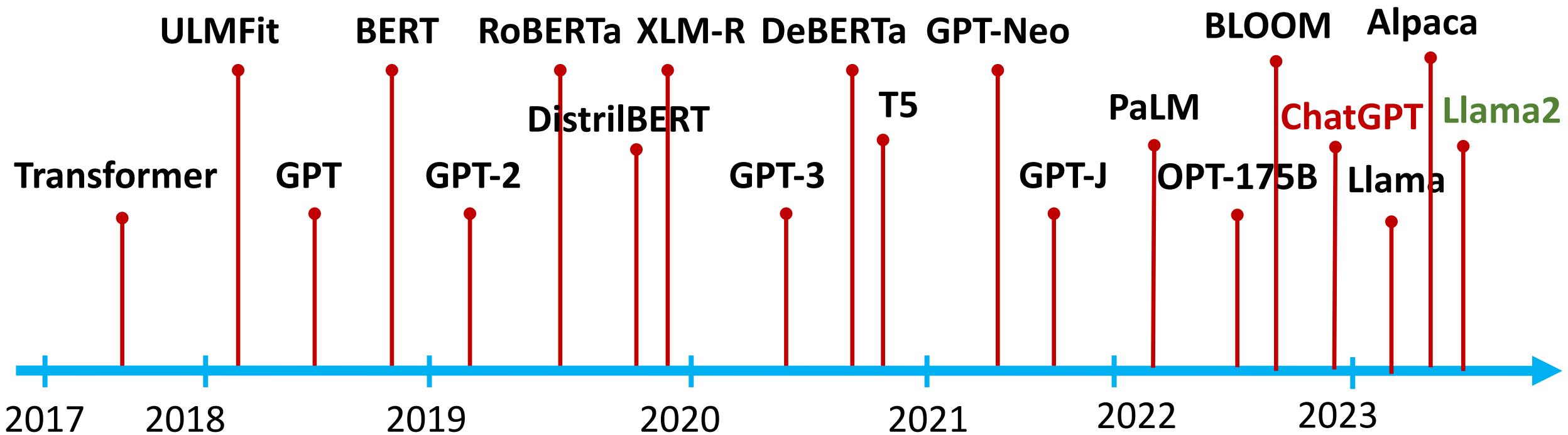
Unsupervised
Learning

Deep Learning (DL)
CNN
RNN LSTM GRU
GAN

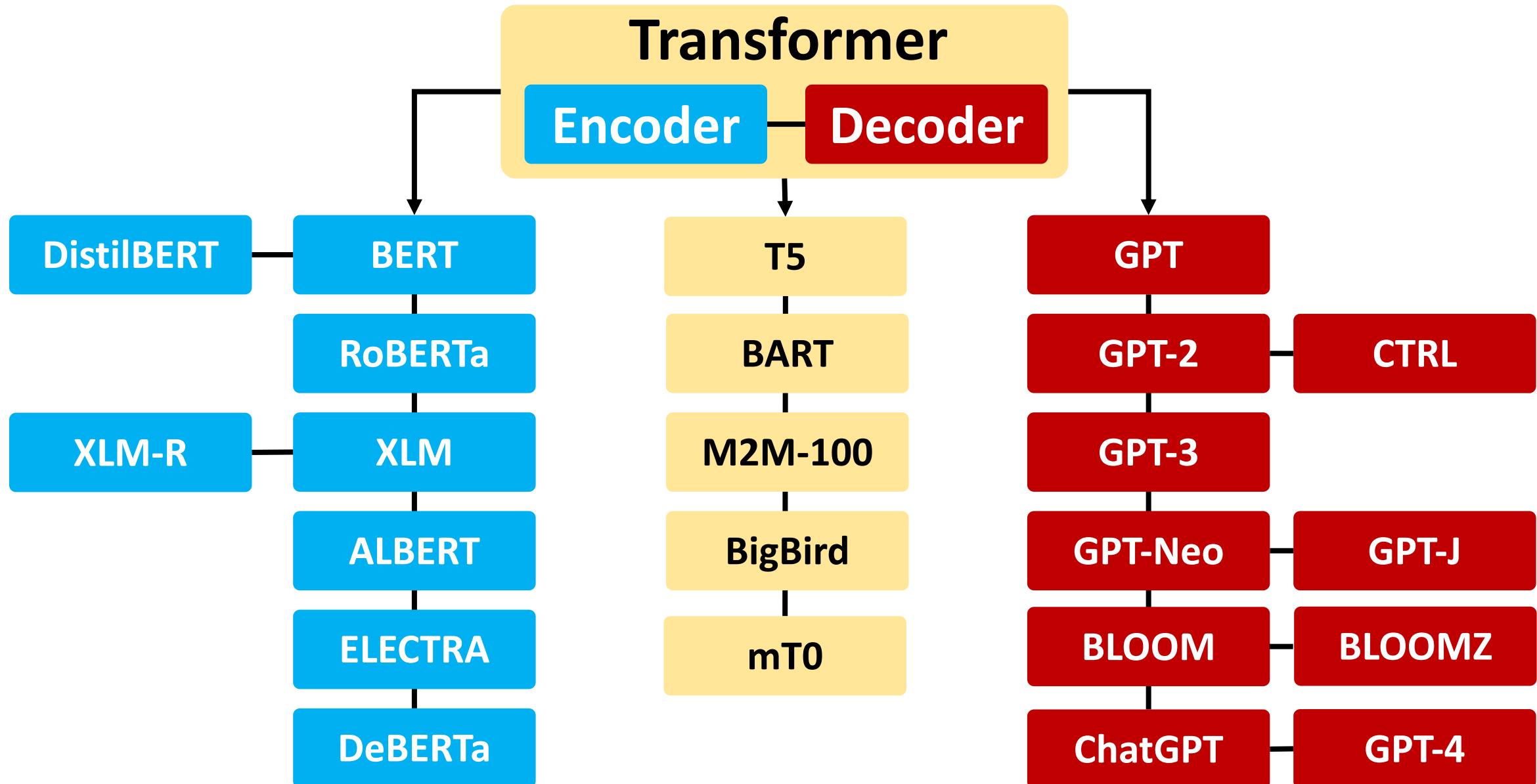
Semi-supervised
Learning

Reinforcement
Learning

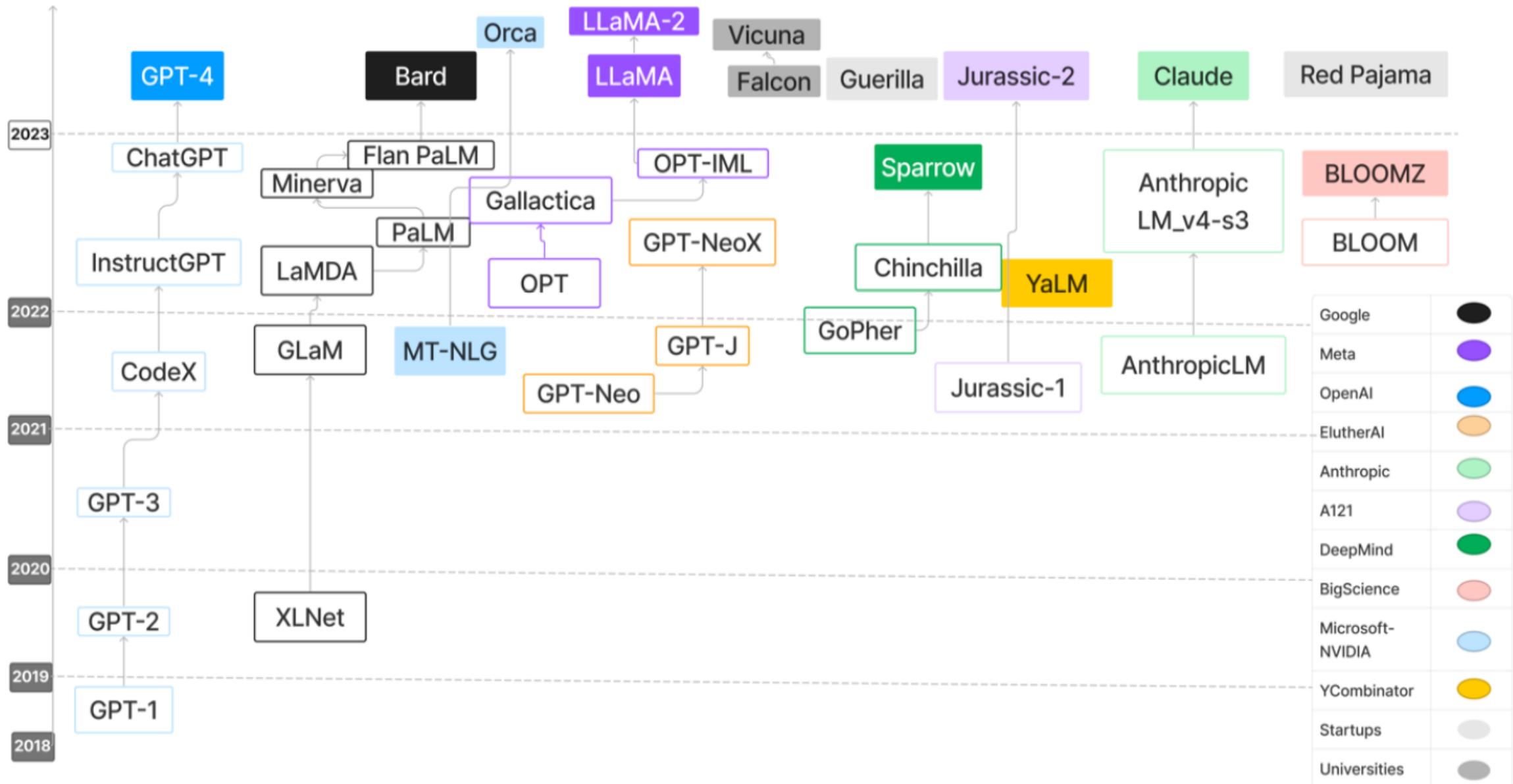
The Transformers Timeline



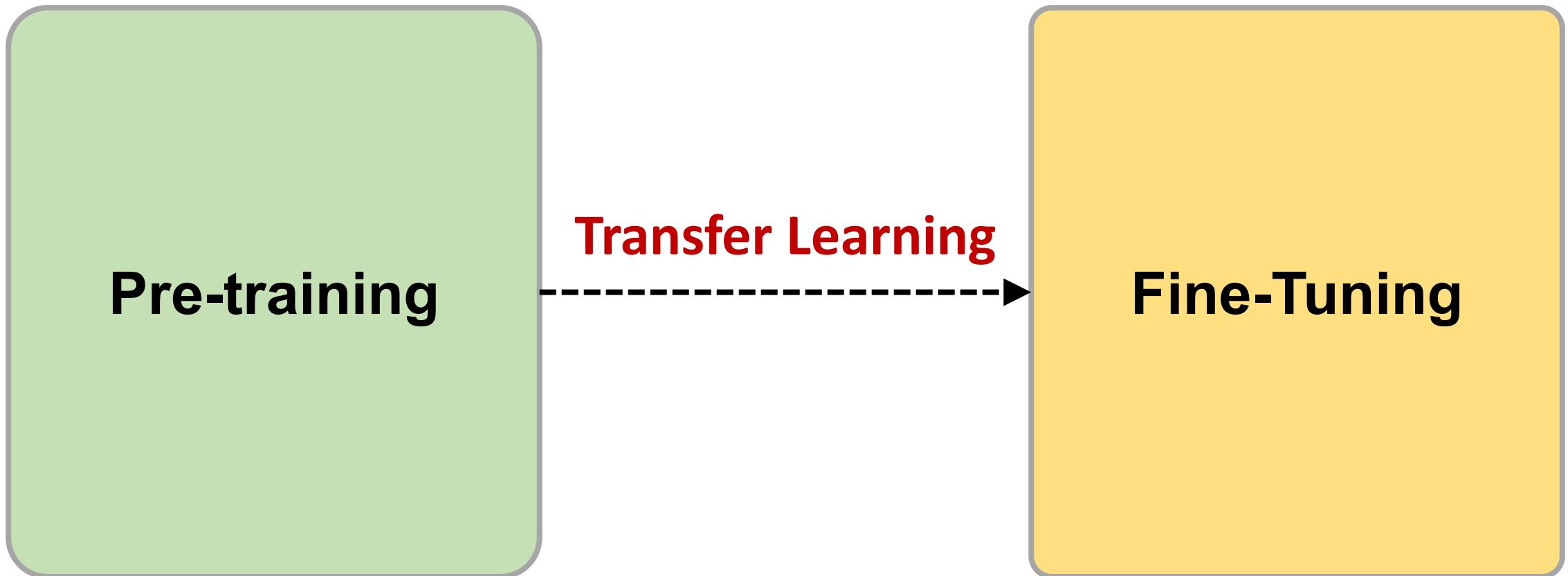
Transformer Models



Large Language Models (LLMs)



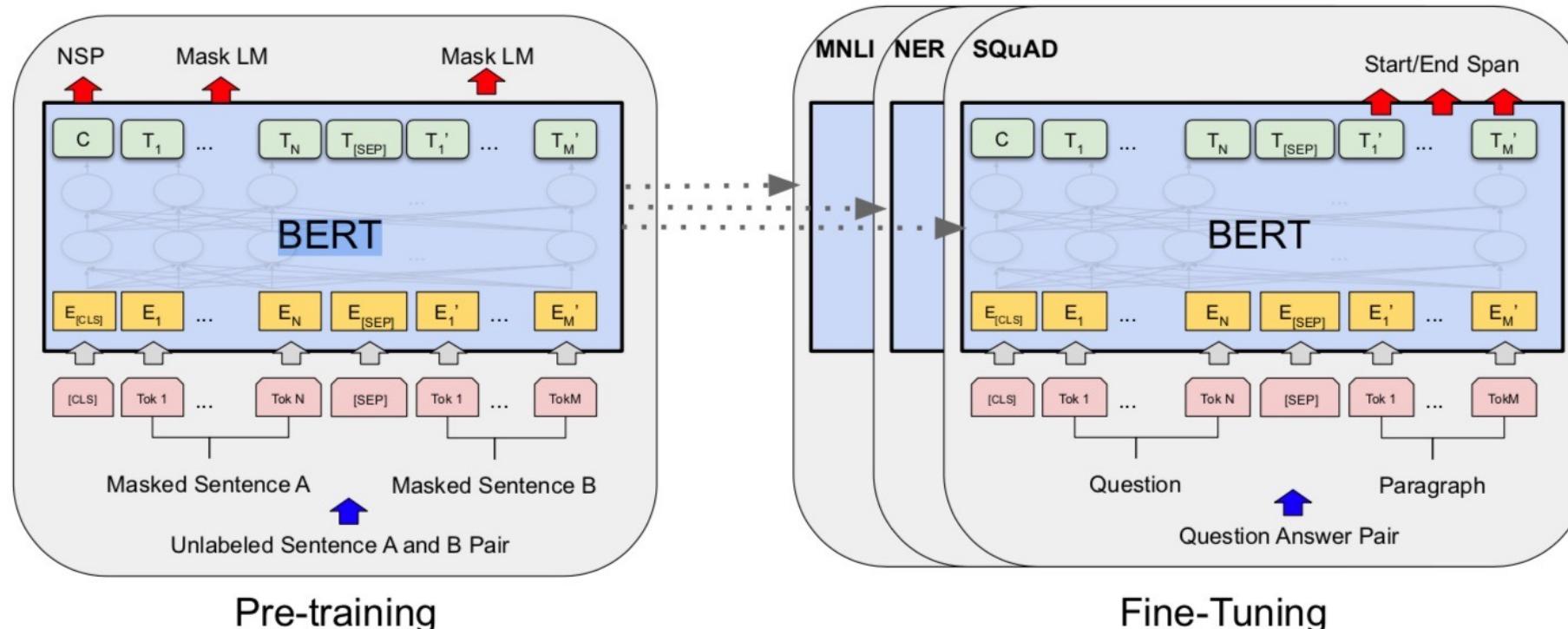
Transfer Learning



BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

BERT (Bidirectional Encoder Representations from Transformers)

Overall pre-training and fine-tuning procedures for BERT

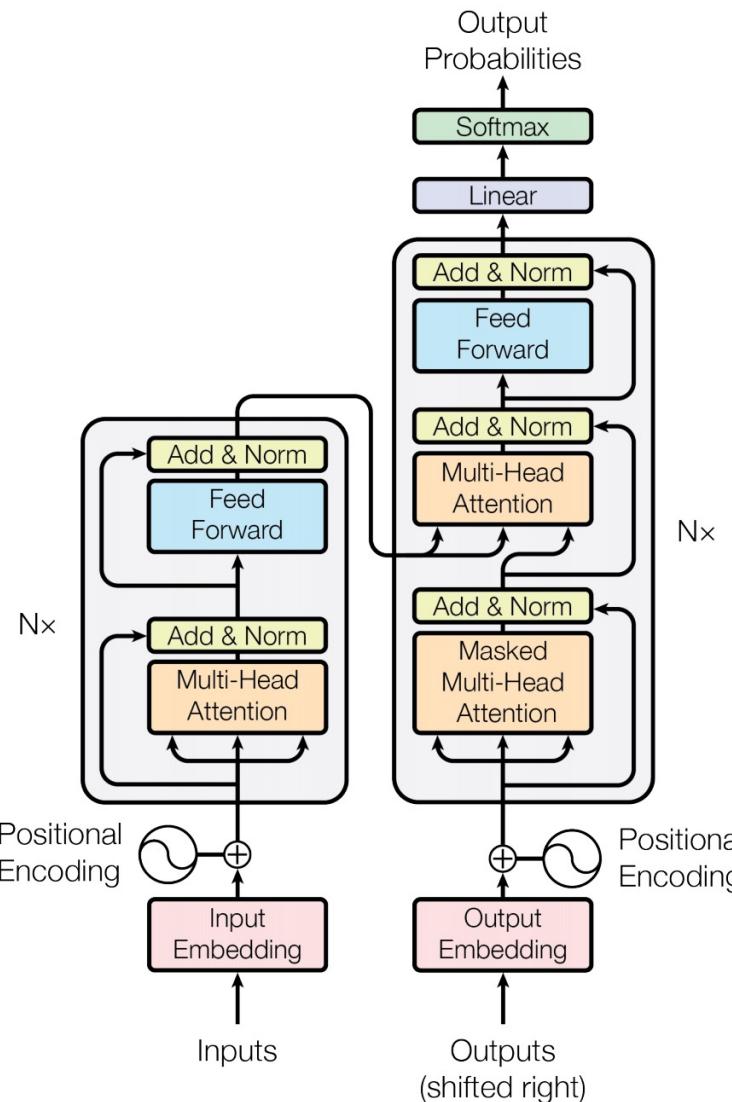


Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).

"Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805.

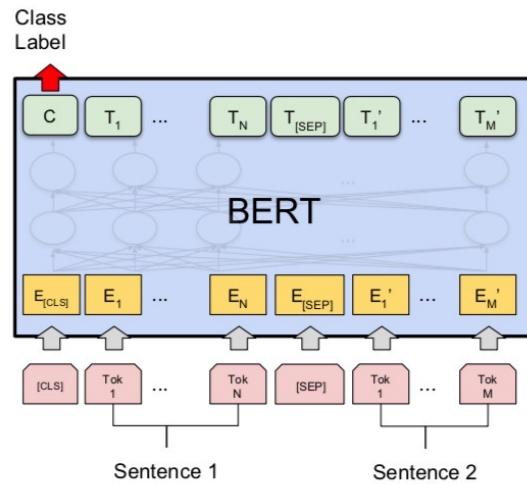
Transformer (Attention is All You Need)

(Vaswani et al., 2017)

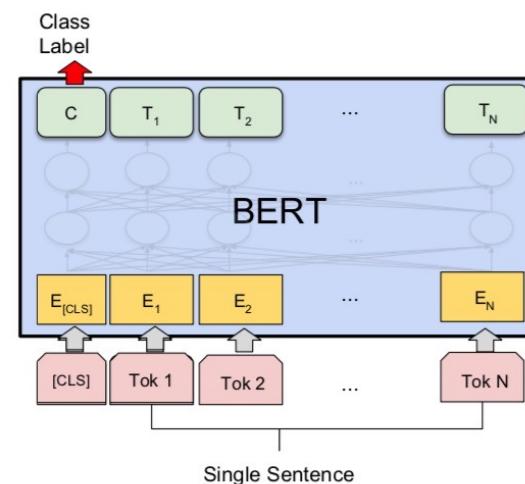


Source: Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin.
"Attention is all you need." In *Advances in neural information processing systems*, pp. 5998-6008. 2017.

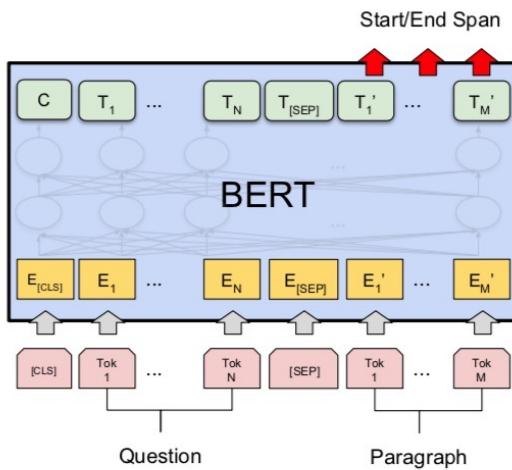
Fine-tuning BERT on Different Tasks



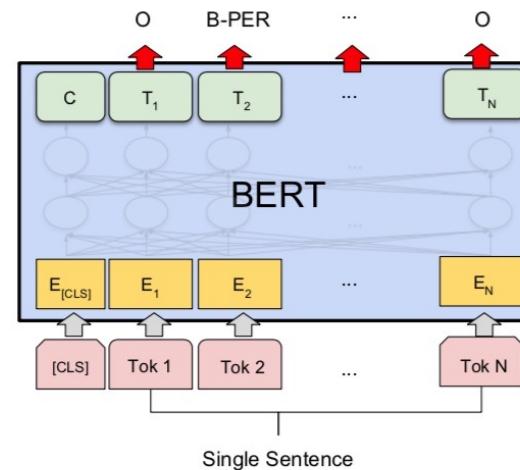
(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG



(b) Single Sentence Classification Tasks:
SST-2, CoLA



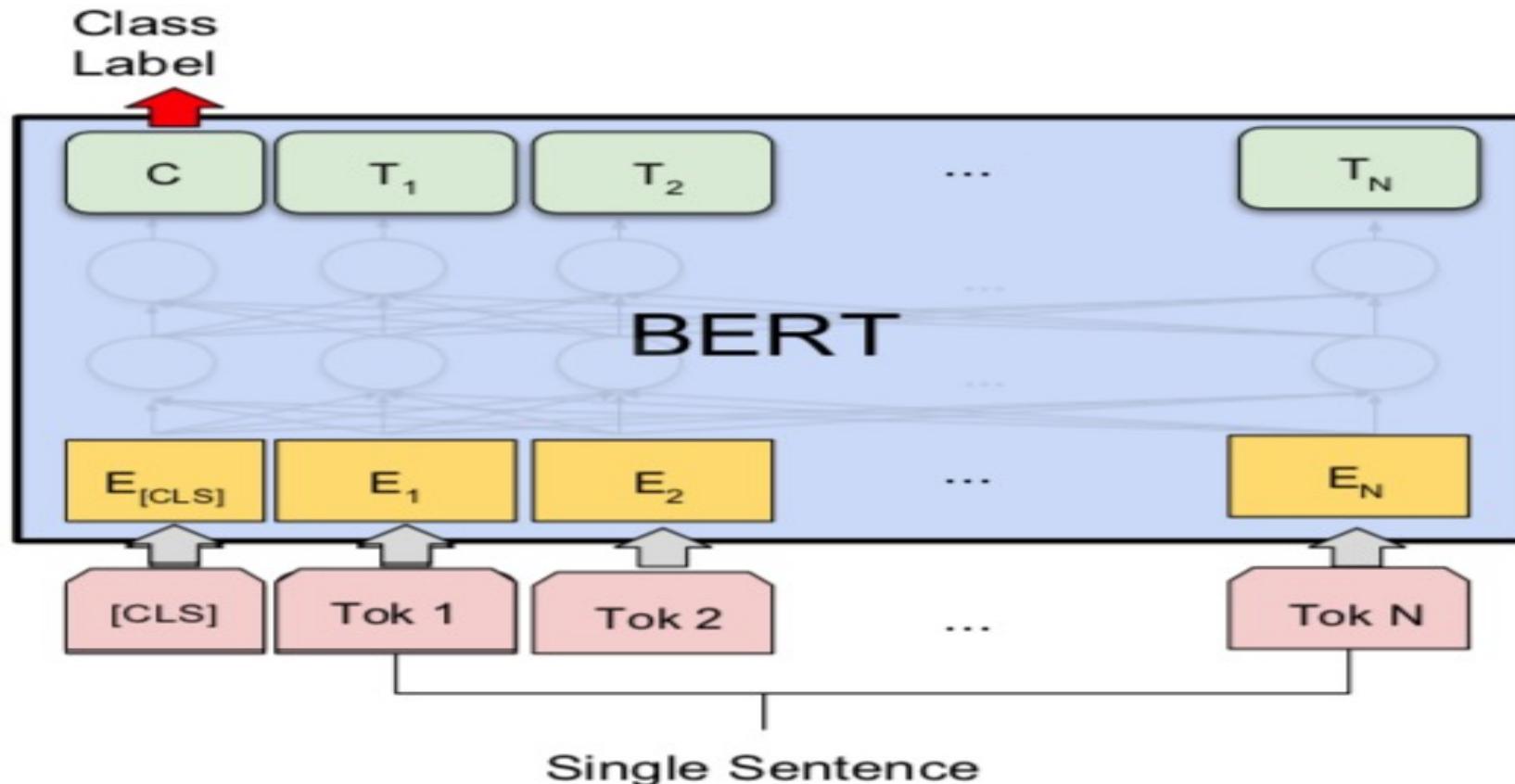
(c) Question Answering Tasks:
SQuAD v1.1



(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).
"Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805.

Sentiment Analysis: Single Sentence Classification

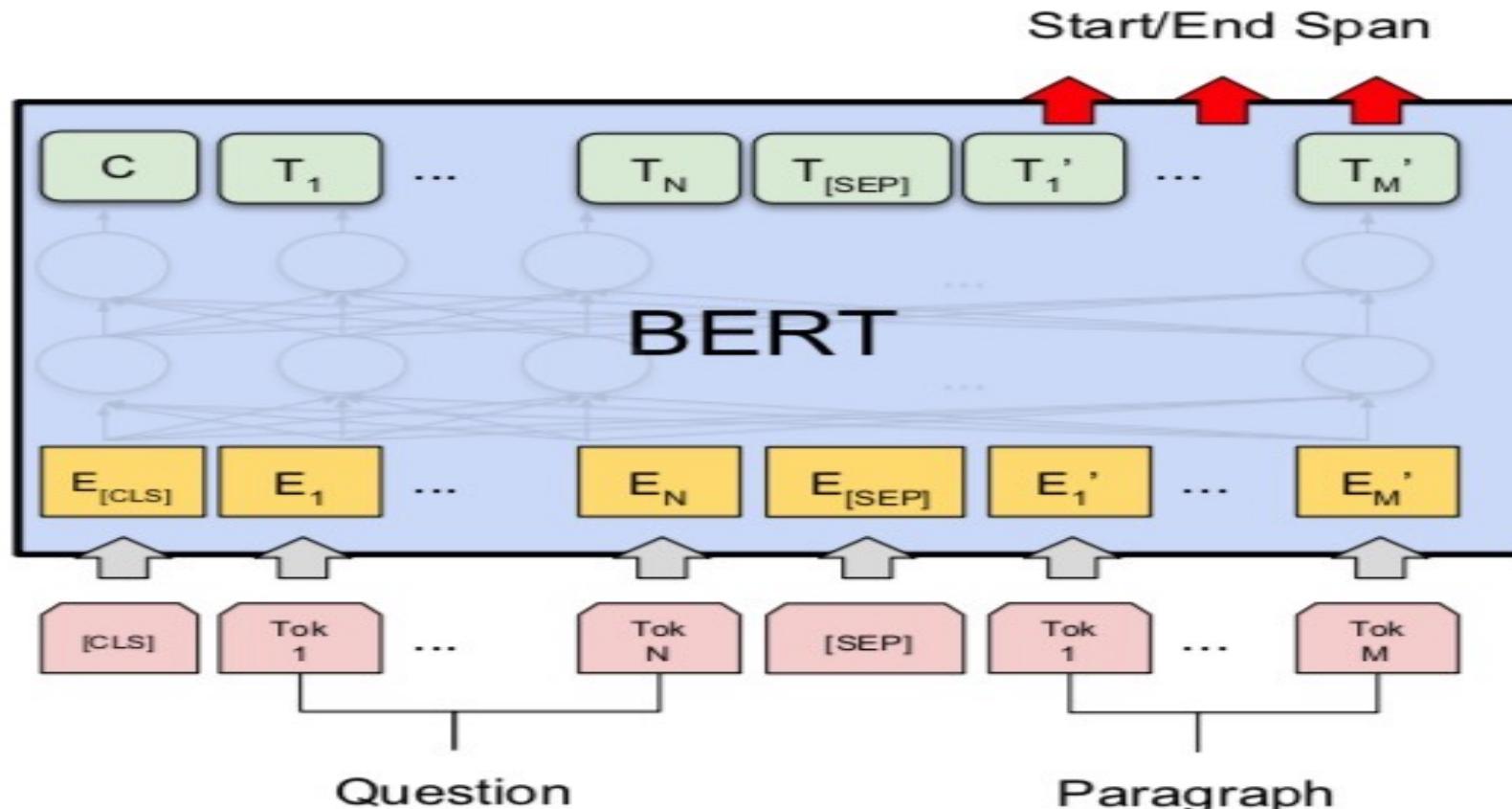


(b) Single Sentence Classification Tasks:
SST-2, CoLA

Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).

"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv preprint arXiv:1810.04805

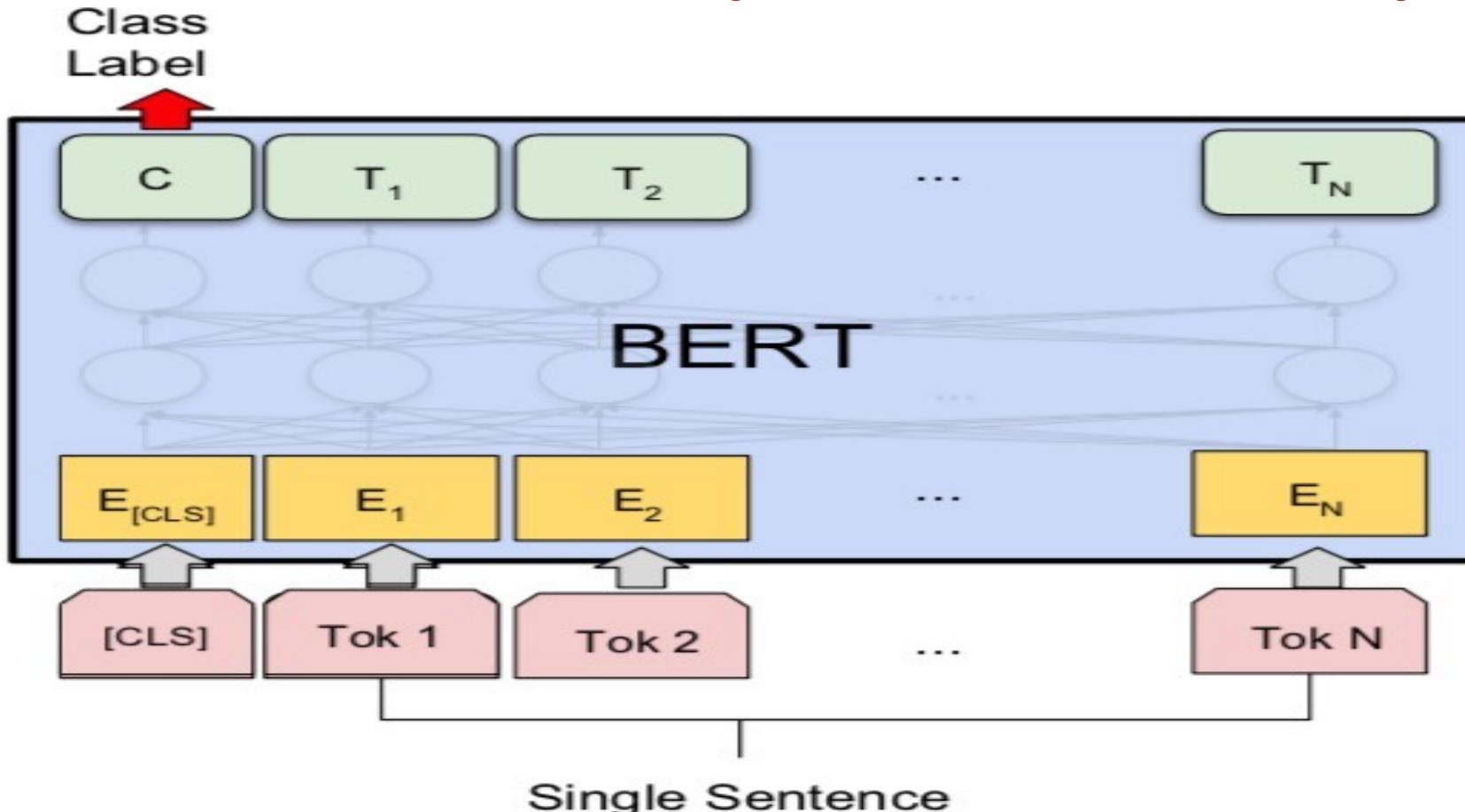
Fine-tuning BERT on Question Answering (QA)



(c) Question Answering Tasks:
SQuAD v1.1

Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).
"Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805.

Fine-tuning BERT on Dialogue Intent Detection (ID; Classification)

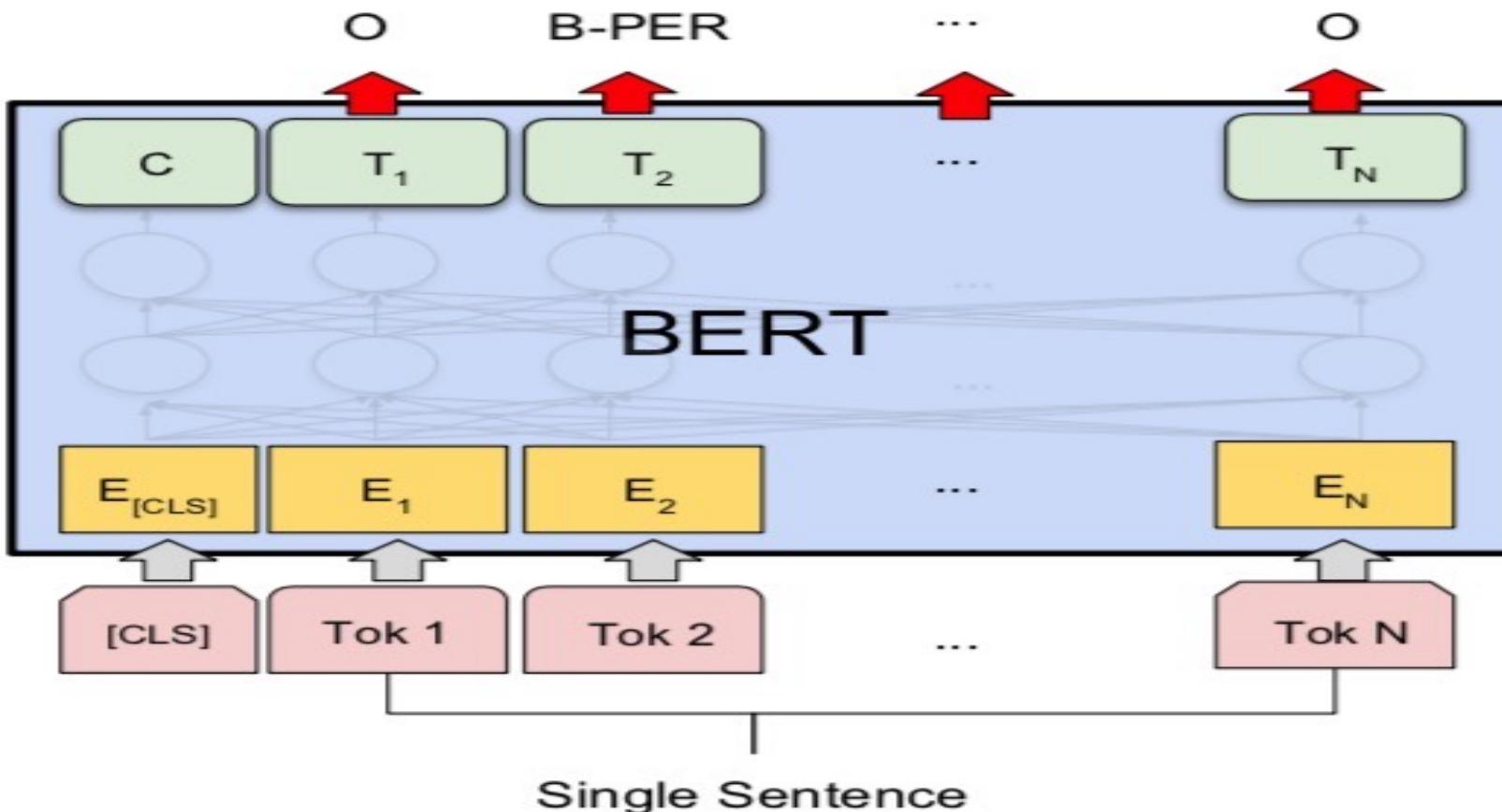


(b) Single Sentence Classification Tasks:
SST-2, CoLA

Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).

"Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805.

Fine-tuning BERT on Dialogue Slot Filling (SF)



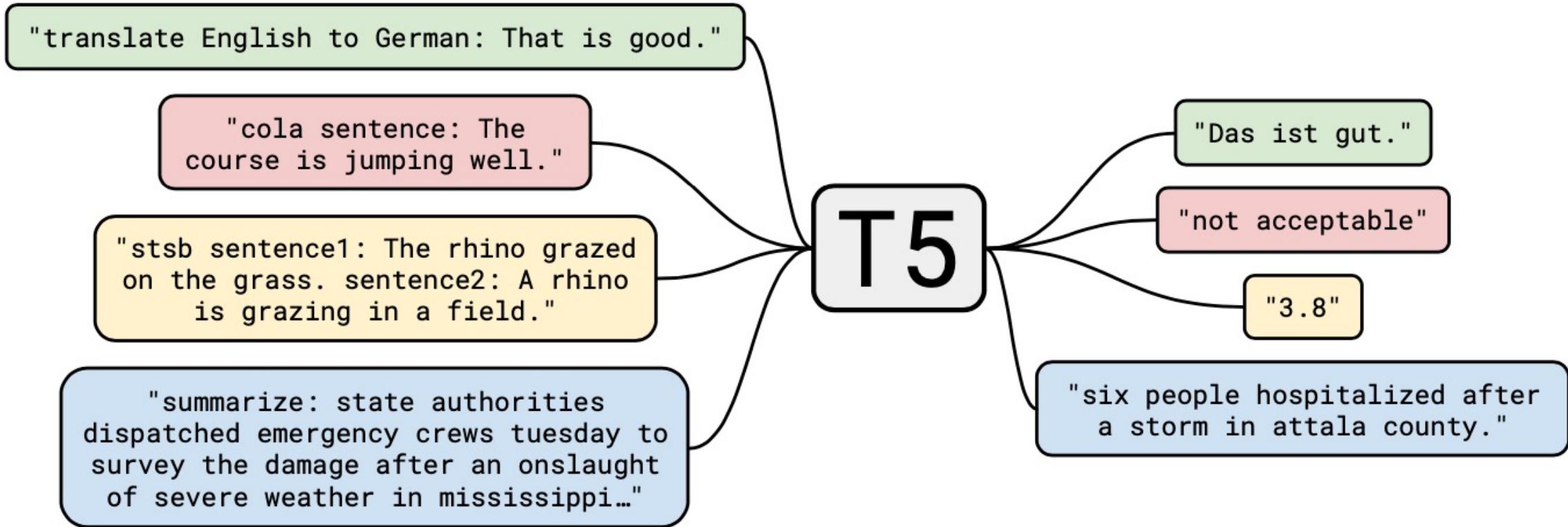
(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).

"Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805.

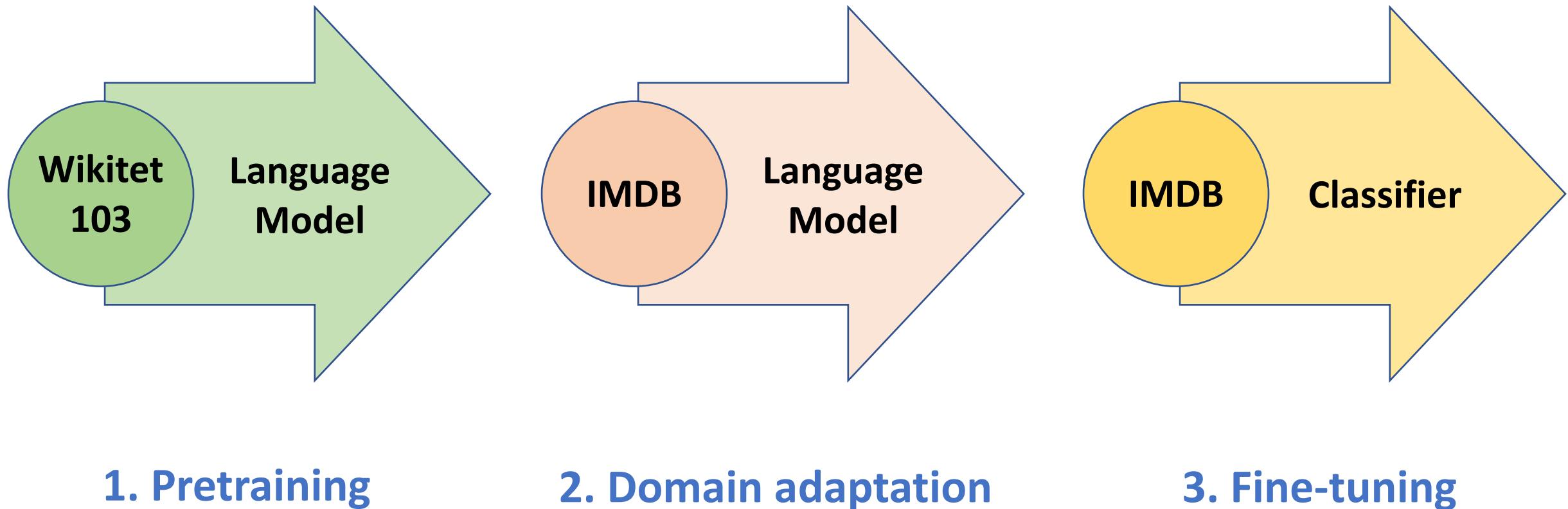
T5

Text-to-Text Transfer Transformer



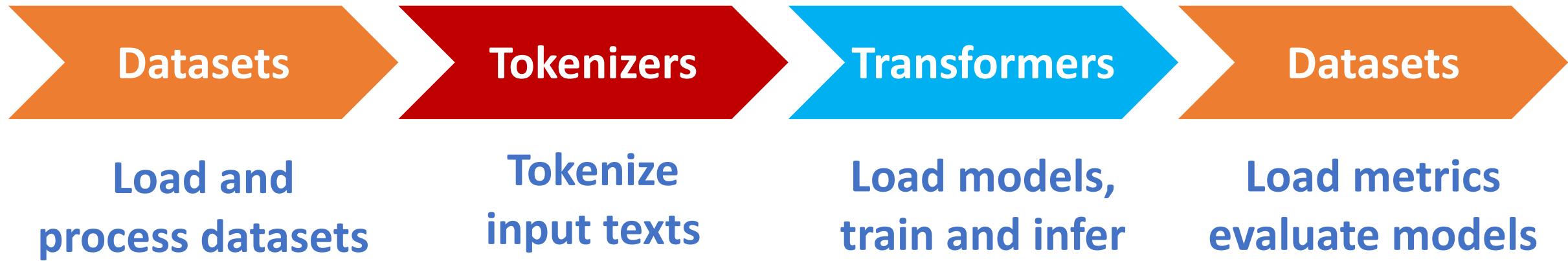
ULMFiT: 3 Steps

Transfer Learning in NLP

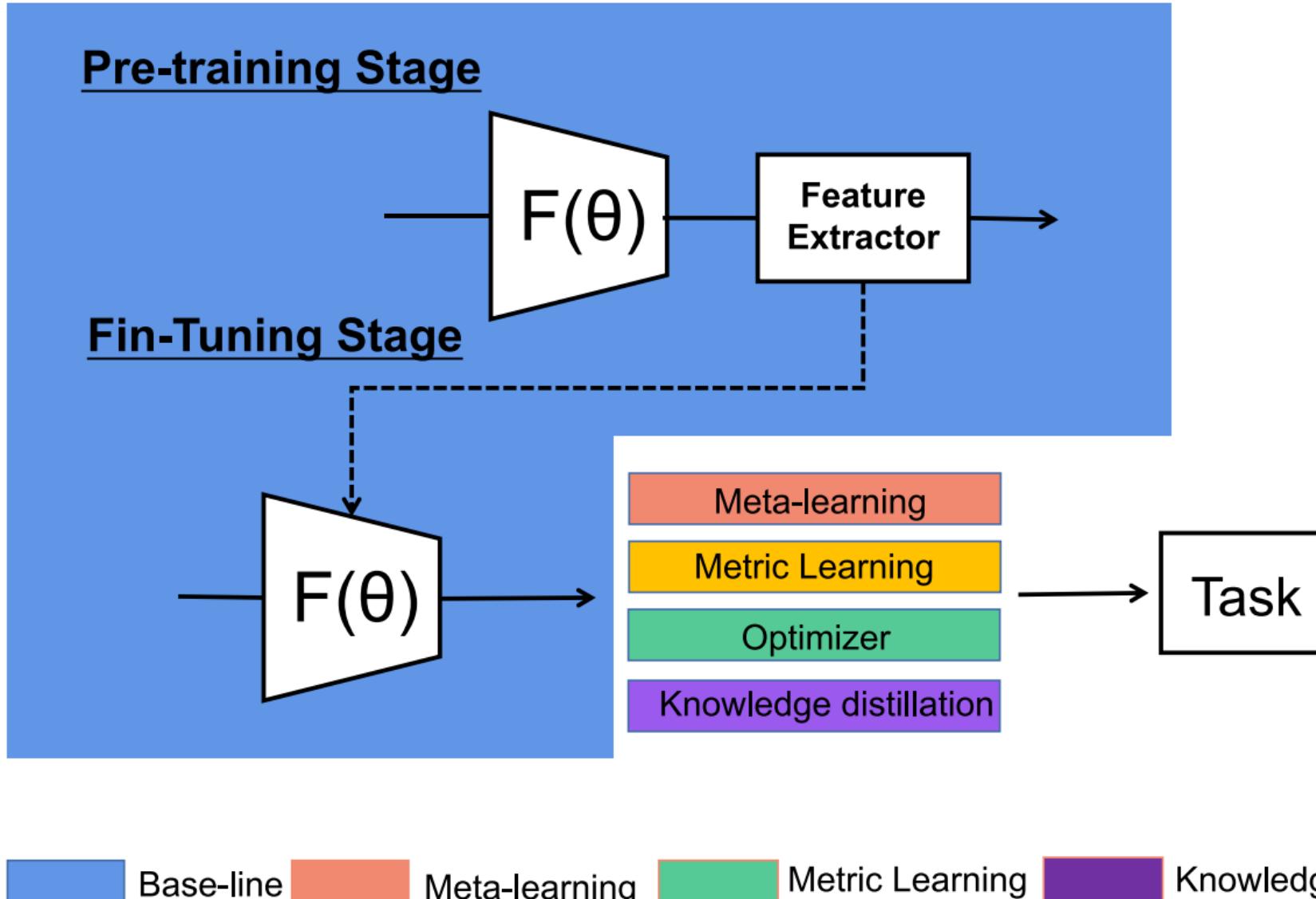


A typical pipeline for training transformer models

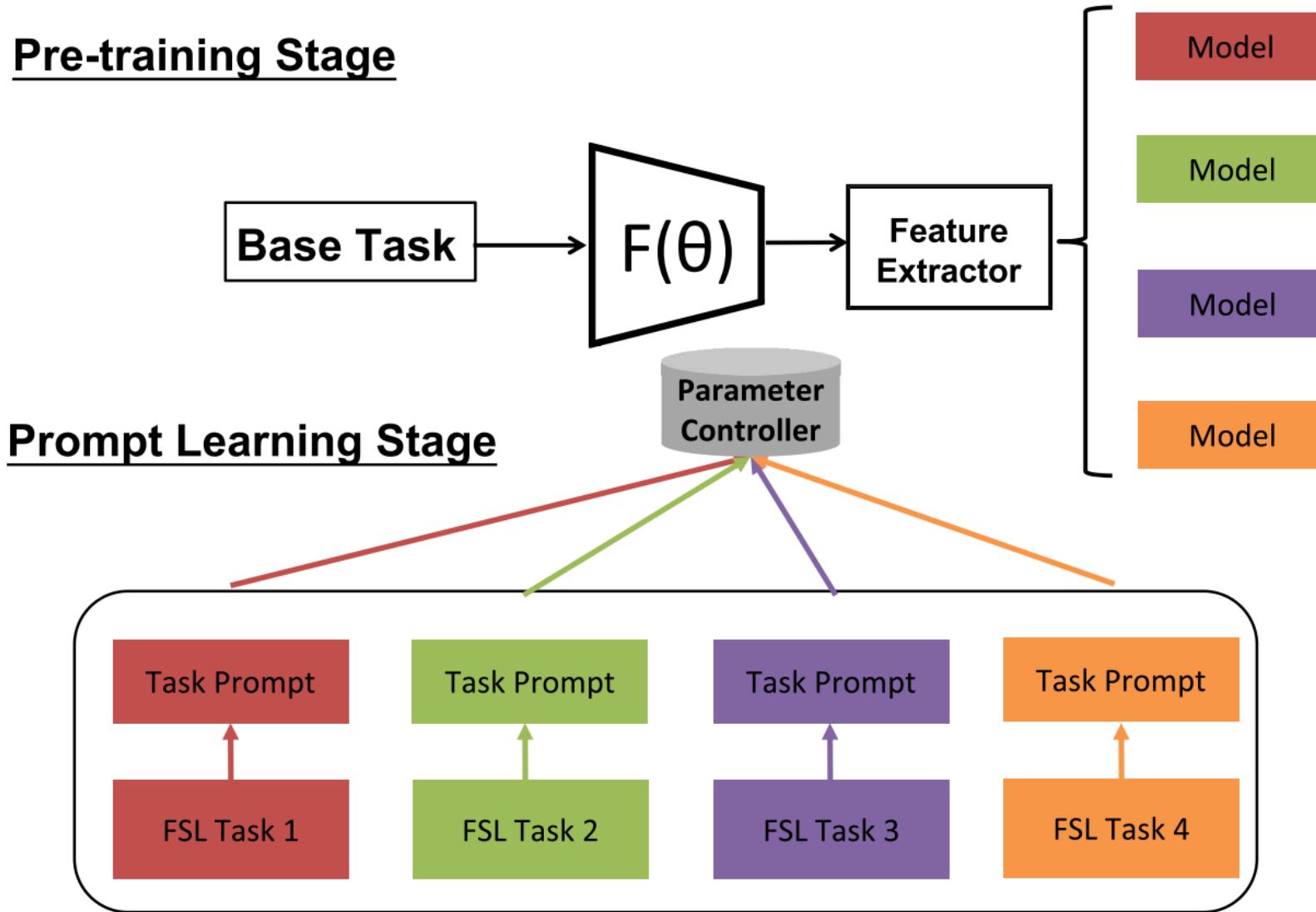
with the Datasets, Tokenizers, and Transformers libraries



Transfer Learning



Prompt Learning



Few-Shot Learning (FSL)

Typical Scenarios

- Acting as a test bed for learning like human
- Learning for rare cases
- Reducing data gathering effort and computational cost

Few-Shot Learning (FSL)

- Few-Shot Learning (FSL) is a sub-area in machine learning.
- Machine Learning Definition
 - A computer program is said to learn from experience **E** with respect to some classes of task **T** and performance measure **P** if its performance can improve with **E** on **T** measured by **P**.
 - Example: Image classification task (**T**), a machine learning program can improve its classification accuracy (**P**) through **E** obtained by training on a large number of labeled images (e.g., the ImageNet data set).

Machine Learning

task T	experience E	performance P
image classification [73]	large-scale labeled images for each class	classification accuracy
the ancient game of Go [120]	a database containing around 30 million recorded moves of human experts and self-play records	winning rate

Few-Shot Learning (FSL)

- Few-shot Learning (FSL) is a type of machine learning problems (specified by E, T, and P), where E contains only a limited number of examples with supervised information for the target T.
 - Existing FSL problems are mainly supervised learning problems.
 - Few-shot classification learns classifiers given only a few labeled examples of each class.
 - image classification
 - sentiment classification from short text
 - object recognition

Few-Shot Learning (FSL)

- Few-shot classification learns a classifier h , which predicts label y_i for each input x_i .
- Usually, one considers the ***N-way-K-shot*** classification, in which D_{train} contains $I = KN$ examples from ***N classes*** each with ***K examples***

Few-Shot Learning (FSL)

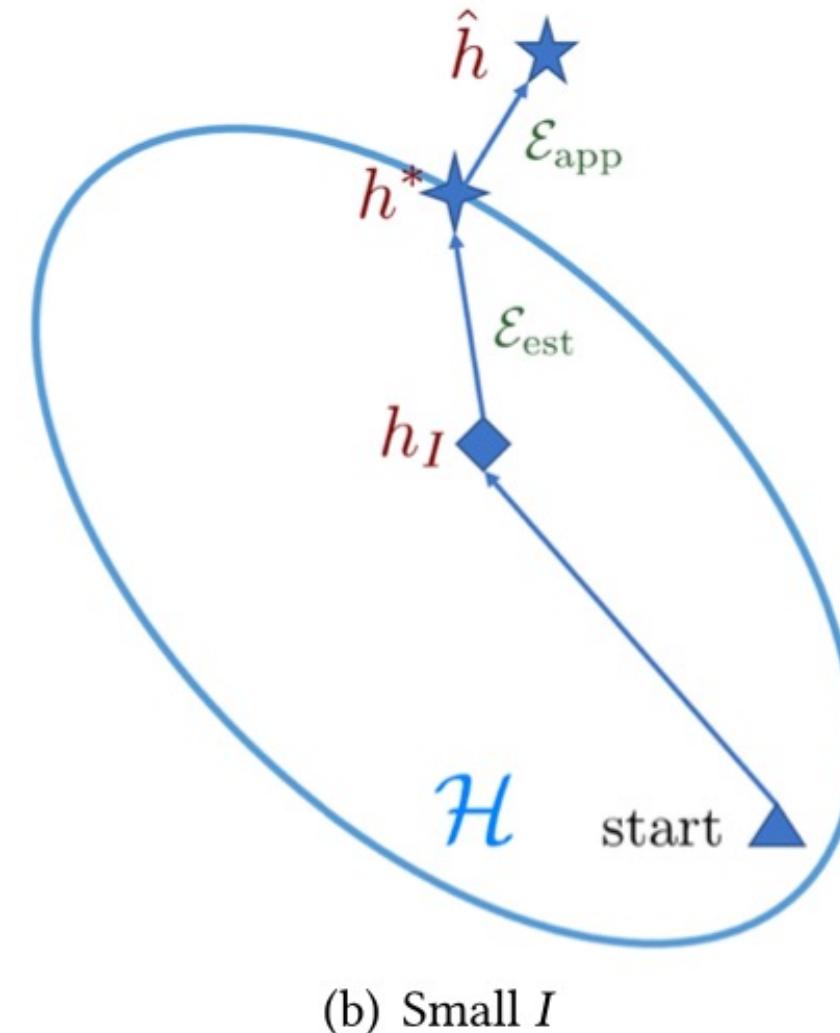
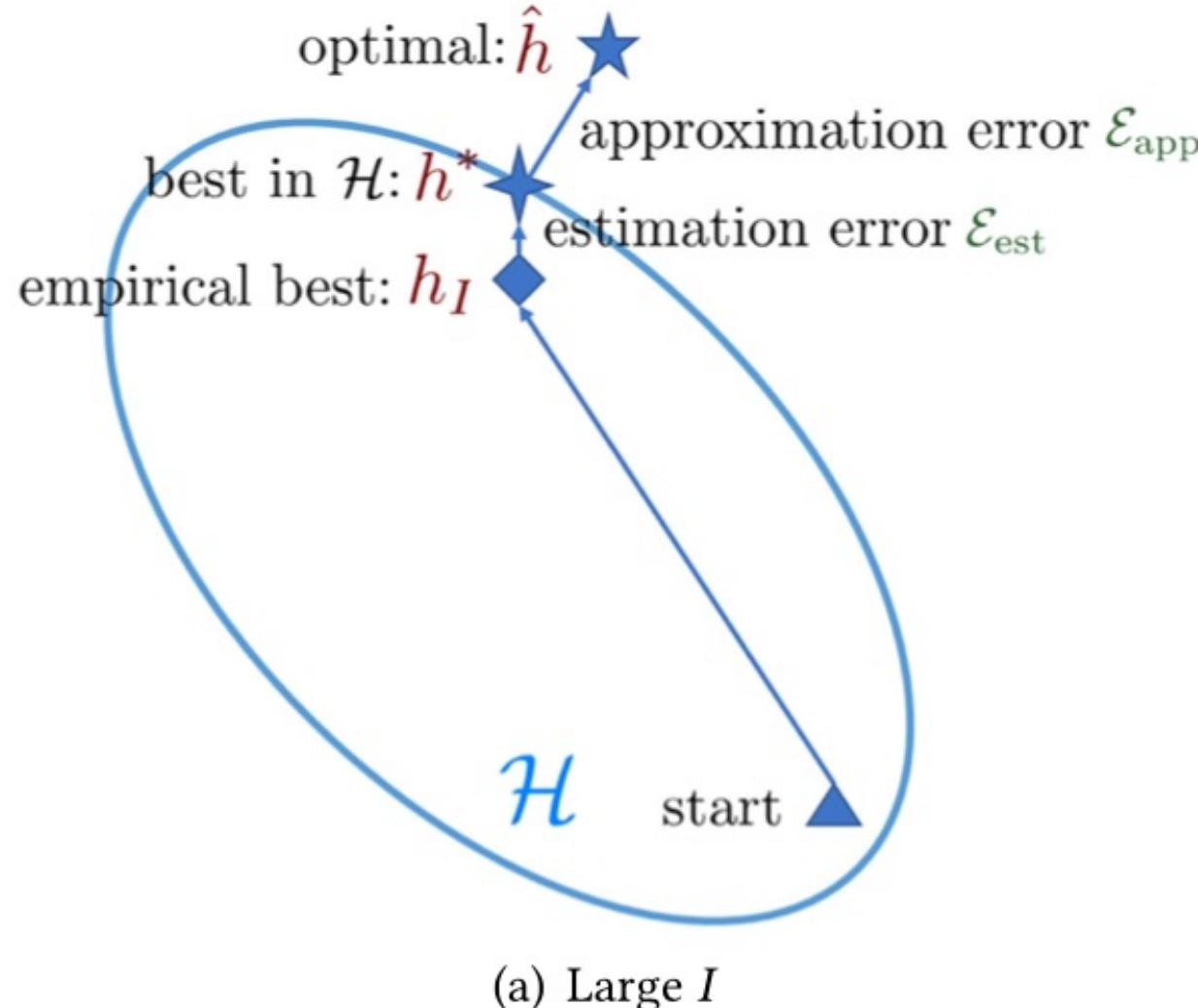
- Few-Shot Learning (FSL)
 - $K = 10 \sim 100$ examples
- One-Shot Learning (1SL)
 - $K = 1$ example
- Zero-Shot Learning (OSL)(ZSL)
 - $K = 0$

Few-Shot Learning (FSL)

task T	experience E		performance P
	supervised information	prior knowledge	
character generation [76]	a few examples of new character	pre-learned knowledge of parts and relations	pass rate of visual Turing test
drug toxicity discovery [4]	new molecule's limited assay	similar molecules' assays	classification accuracy
image classification [70]	a few labeled images for each class of the target T	raw images of other classes, or pre-trained models	classification accuracy

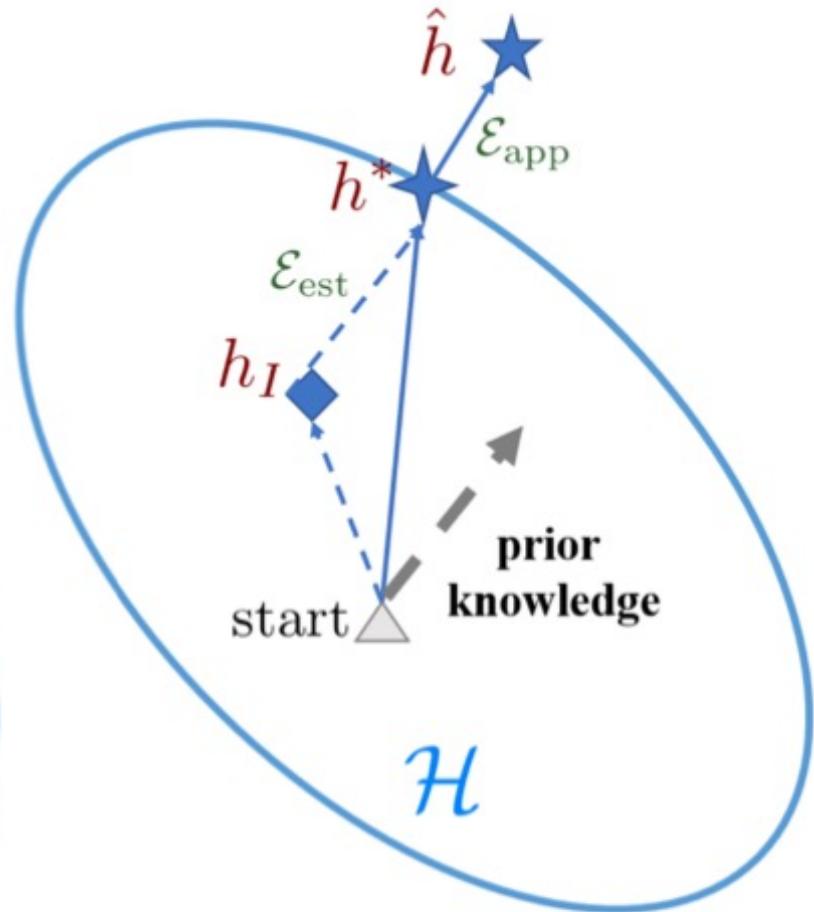
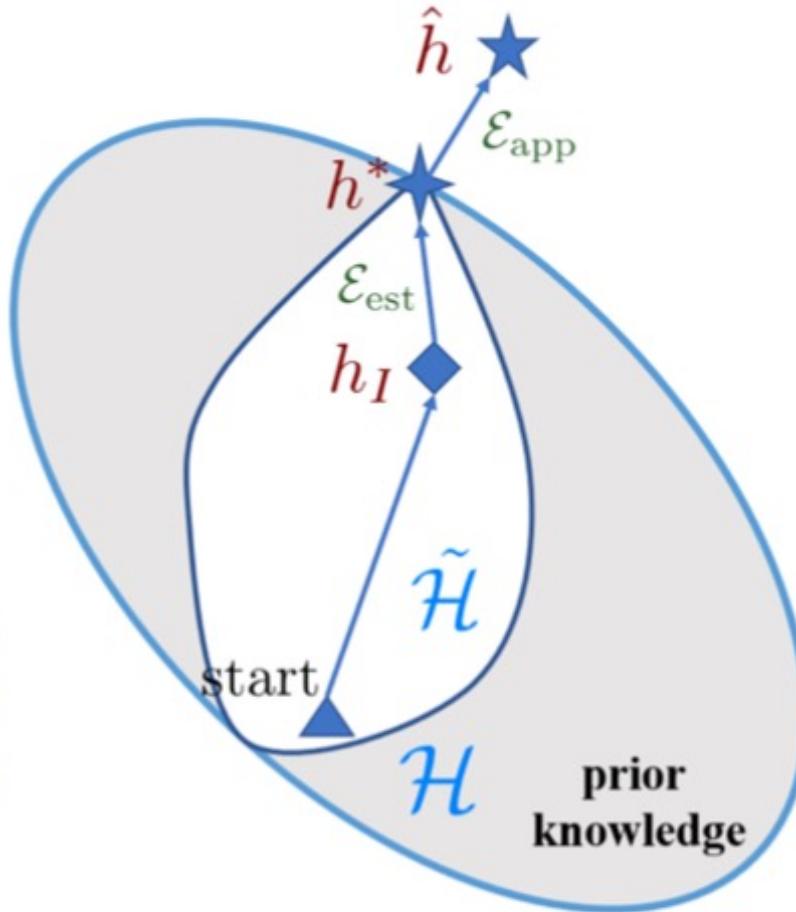
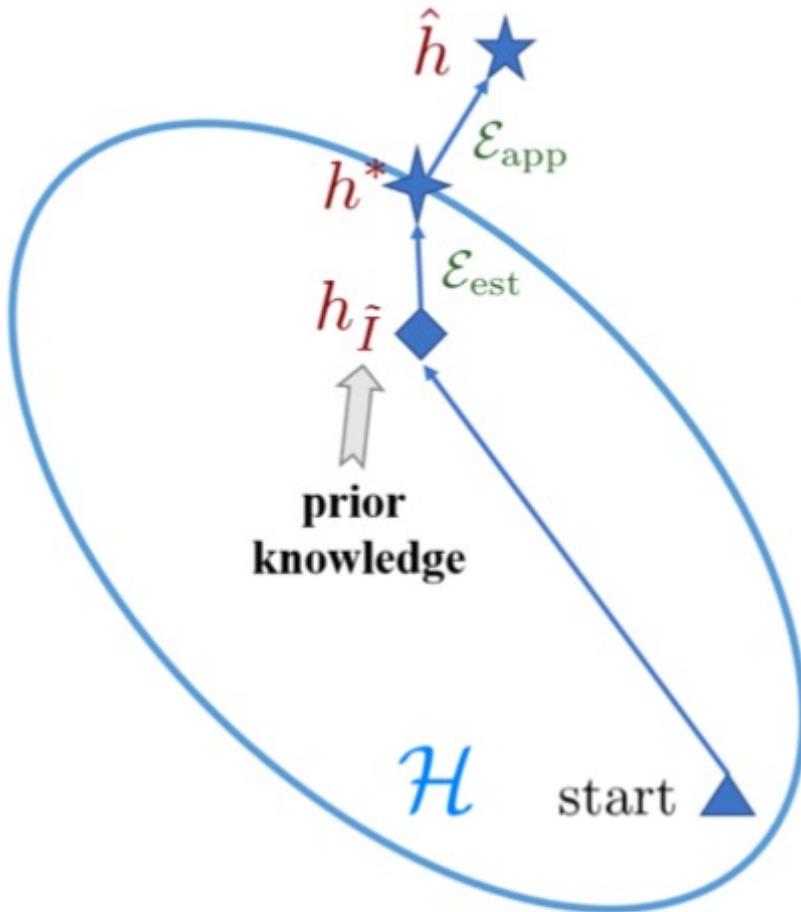
Few-Shot Learning (FSL)

Comparison of learning with sufficient and few training samples



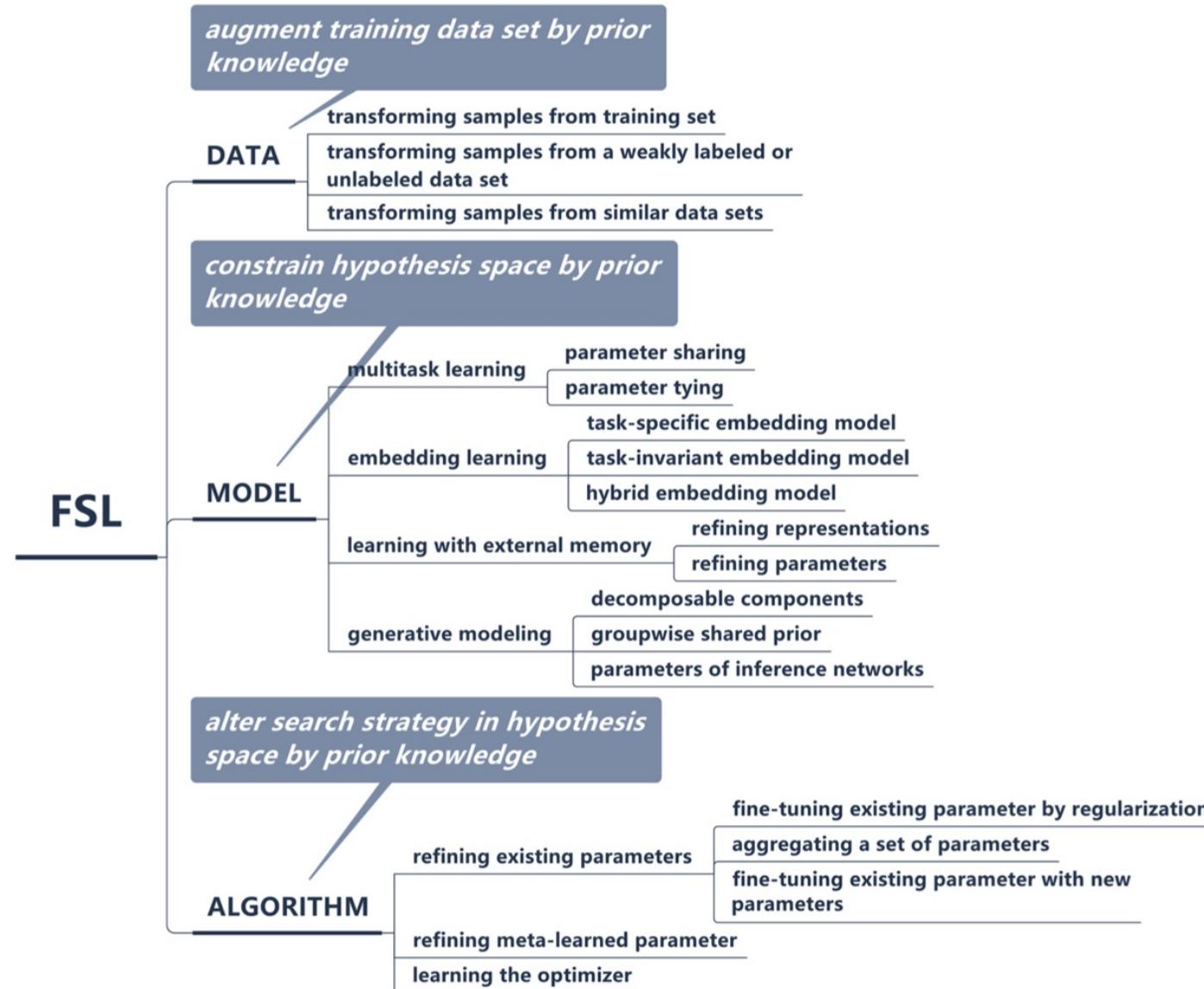
Few-Shot Learning (FSL)

Different perspectives on how FSL methods solve the few-shot problem



Few-Shot Learning (FSL)

A taxonomy of FSL methods



Few-Shot Learning (FSL)

augment training data set by prior knowledge

DATA

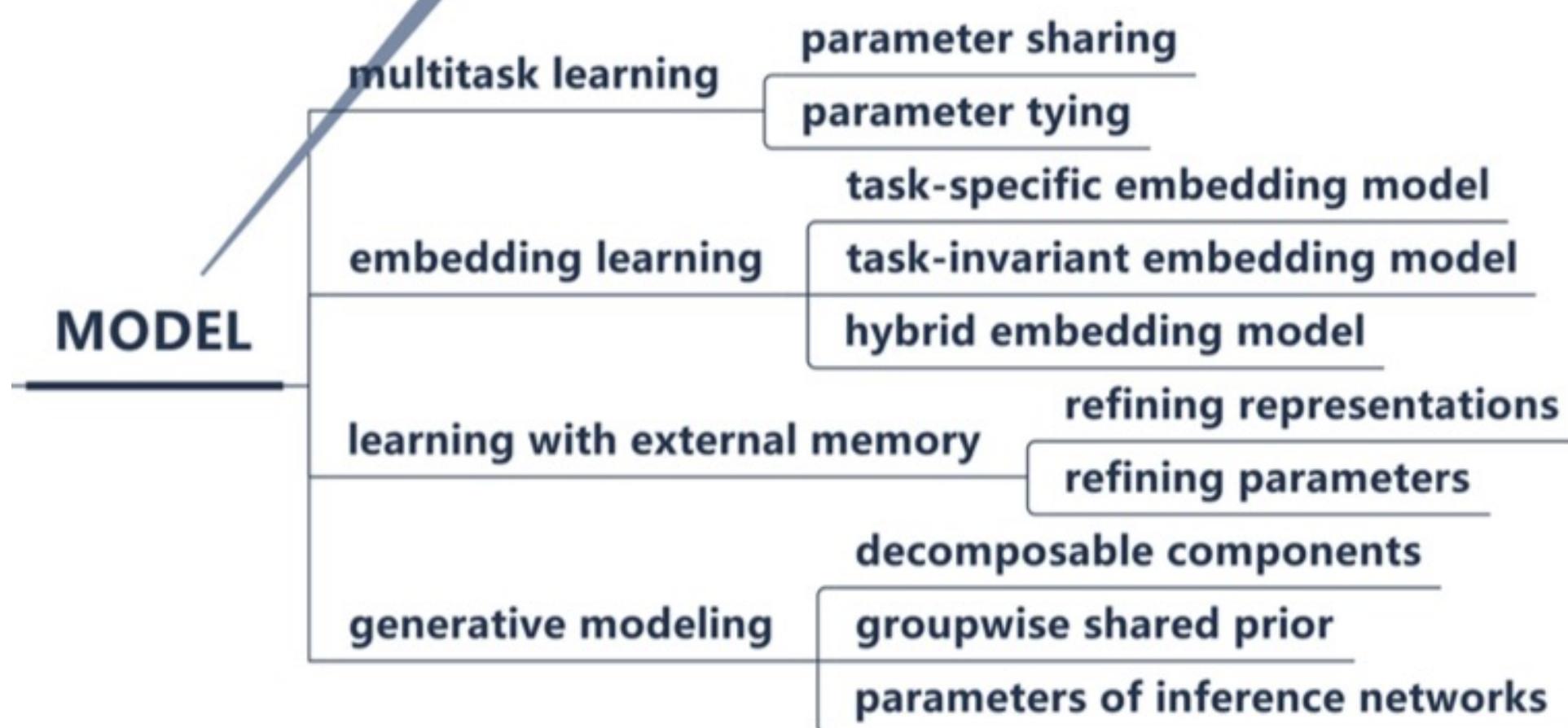
transforming samples from training set

transforming samples from a weakly labeled or unlabeled data set

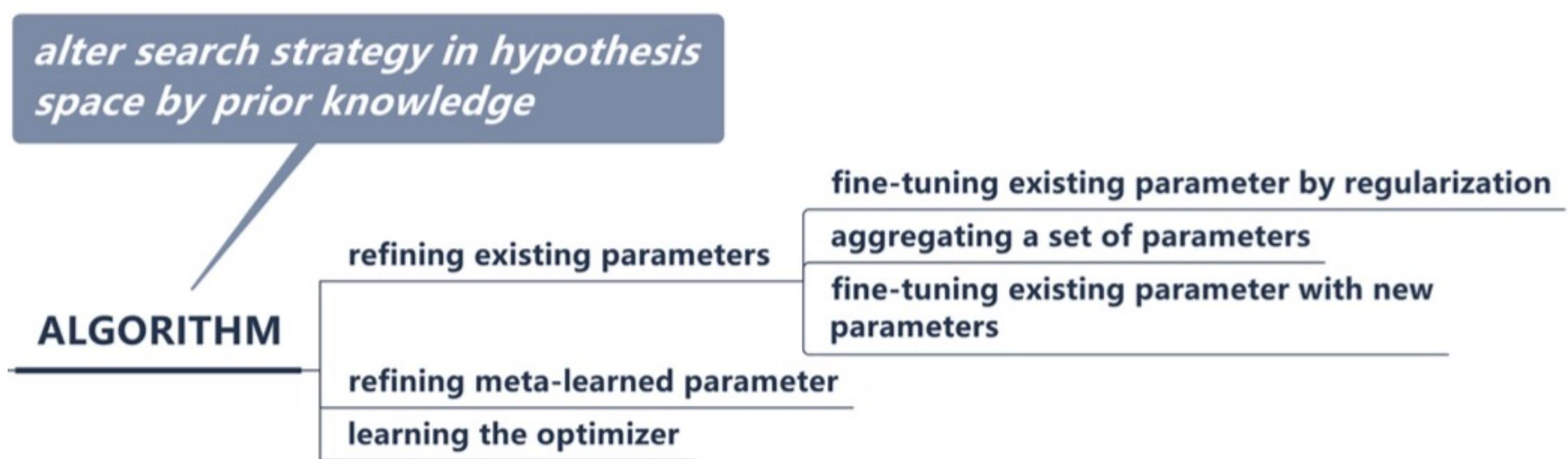
transforming samples from similar data sets

Few-Shot Learning (FSL)

constrain hypothesis space by prior knowledge

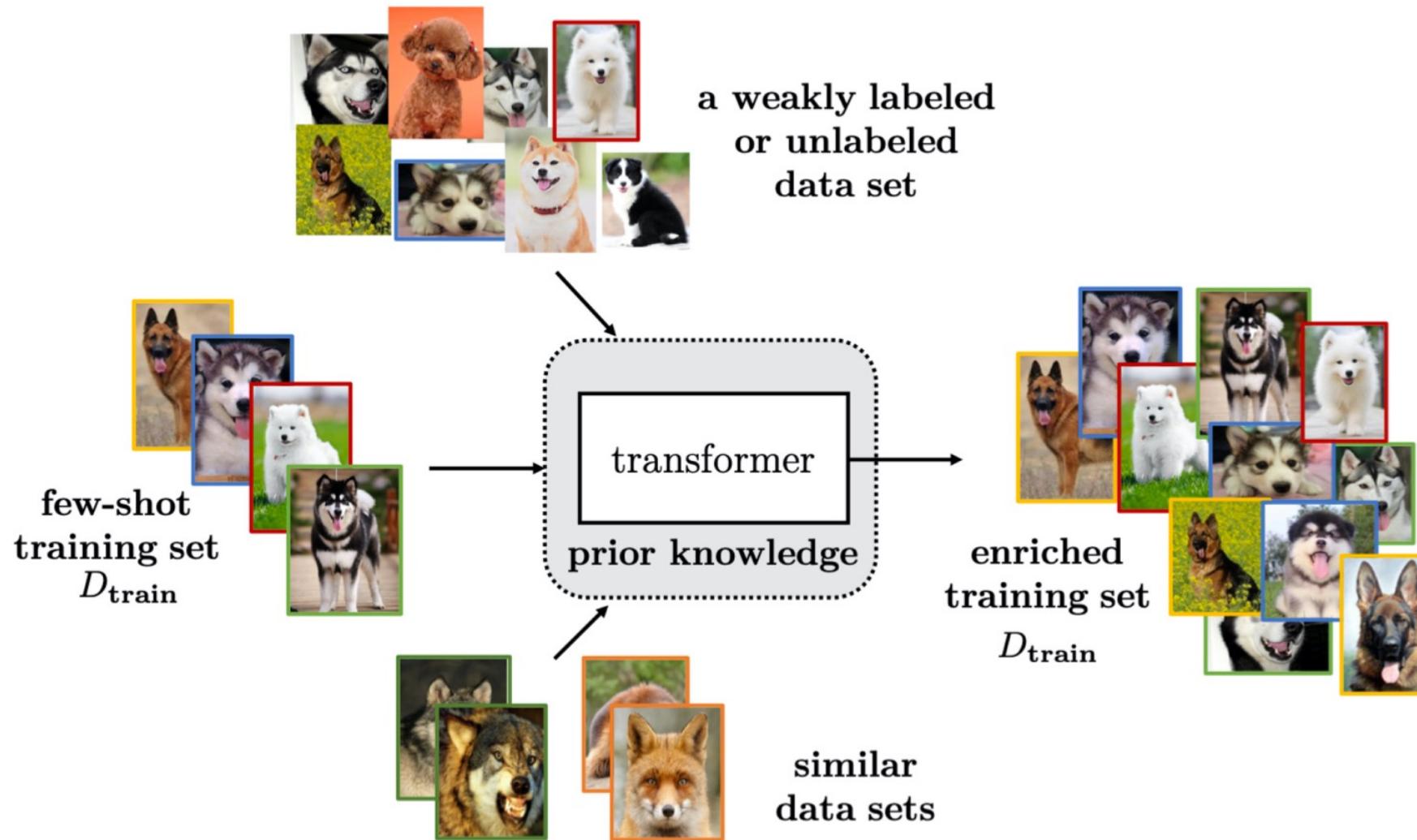


Few-Shot Learning (FSL)



Few-Shot Learning (FSL)

Solving the FSL problem by data augmentation



Few-Shot Learning (FSL)

Characteristics for FSL Methods Focusing on the Data Perspective

category	input (x, y)	transformer t	output (\tilde{x}, \tilde{y})
transforming samples from D_{train}	original (x_i, y_i)	learned transformation function on x_i	$(t(x_i), y_i)$
transforming samples from a weakly labeled or unlabeled data set	weakly labeled or unlabeled $(\bar{x}, -)$	a predictor trained from D_{train}	$(\bar{x}, t(\bar{x}))$
transforming samples from similar data sets	samples $\{(\hat{x}_j, \hat{y}_j)\}$ from similar data sets	an aggregator to combine $\{(\hat{x}_j, \hat{y}_j)\}$	$(t(\{\hat{x}_j\}), t(\{\hat{y}_j\}))$

The transformer $t(\cdot)$ takes input (x, y) and returns synthesized sample (\tilde{x}, \tilde{y}) to augment the few-shot D_{train} .

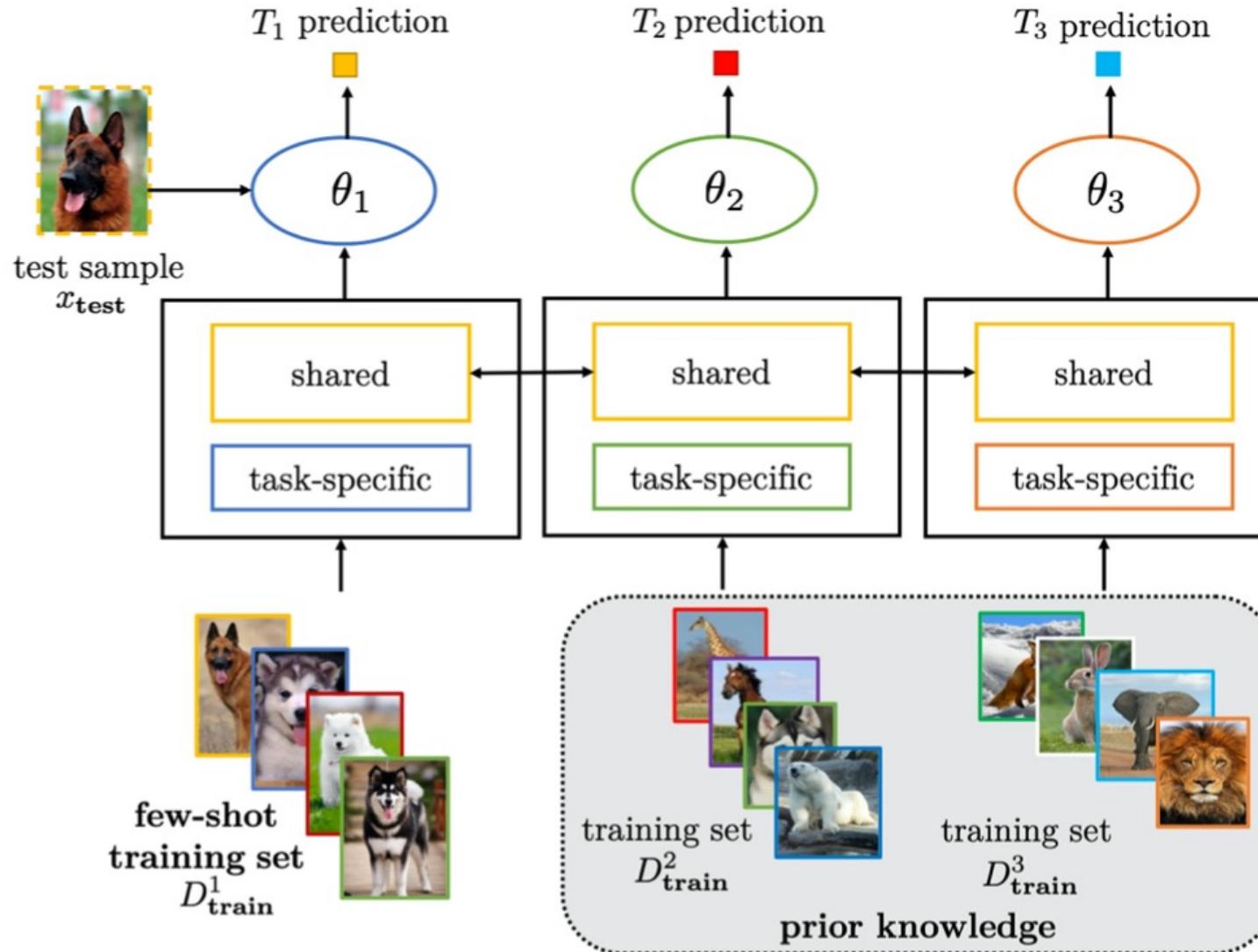
Few-Shot Learning (FSL)

Characteristics for FSL Methods Focusing on the Model Perspective

strategy	prior knowledge	how to constrain \mathcal{H}
multitask learning	other T 's with their data sets D 's	share/tie parameter
embedding learning	embedding learned from/together with other T 's	project samples to a smaller embedding space in which similar and dissimilar samples can be easily discriminated
learning with external memory	embedding learned from other T 's to interact with memory	refine samples using key-value pairs stored in memory
generative modeling	prior model learned from other T 's	restrict the form of distribution

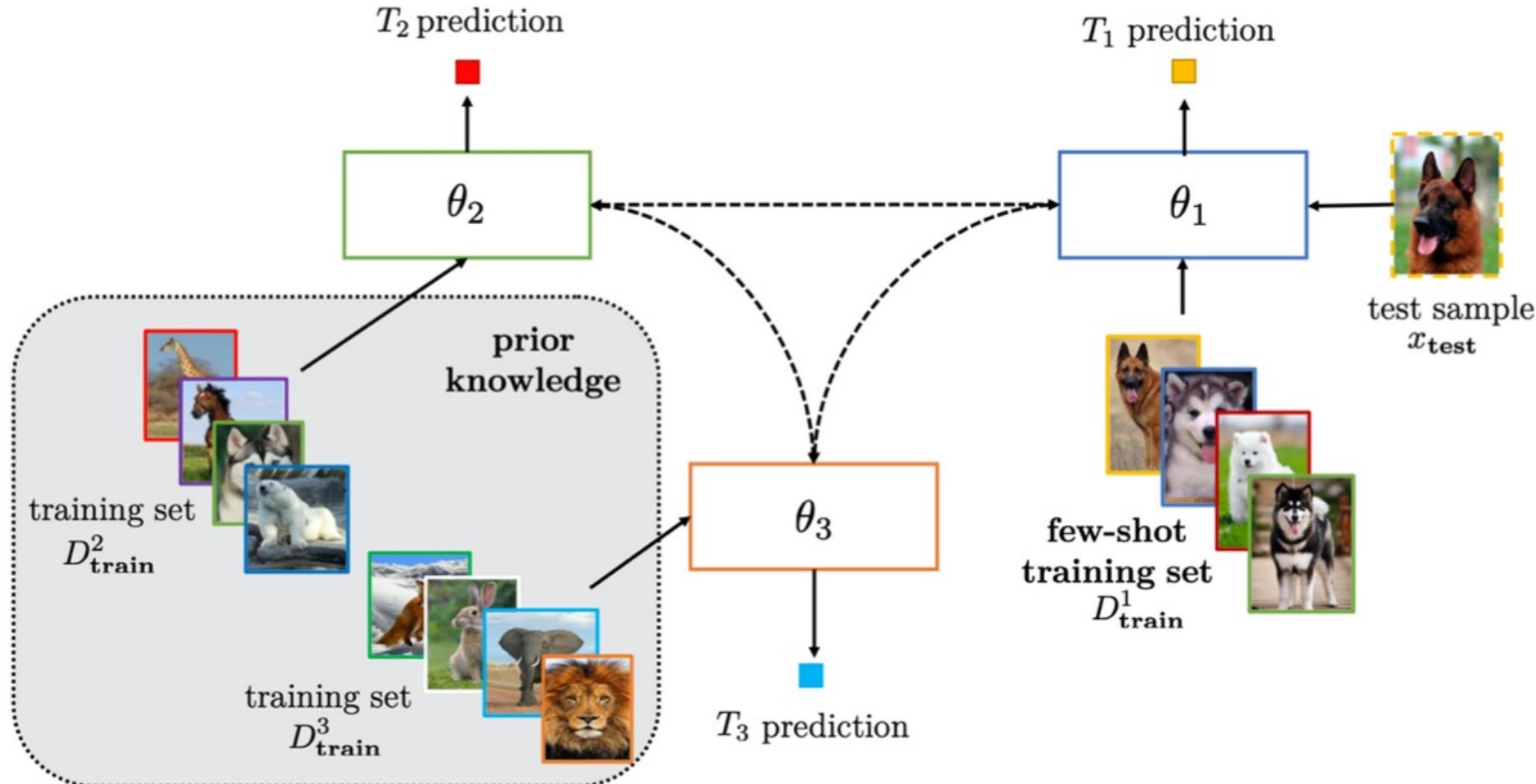
Few-Shot Learning (FSL)

Solving the FSL problem by multitask learning with parameter sharing



Few-Shot Learning (FSL)

Solving the FSL problem by multitask learning with parameter tying



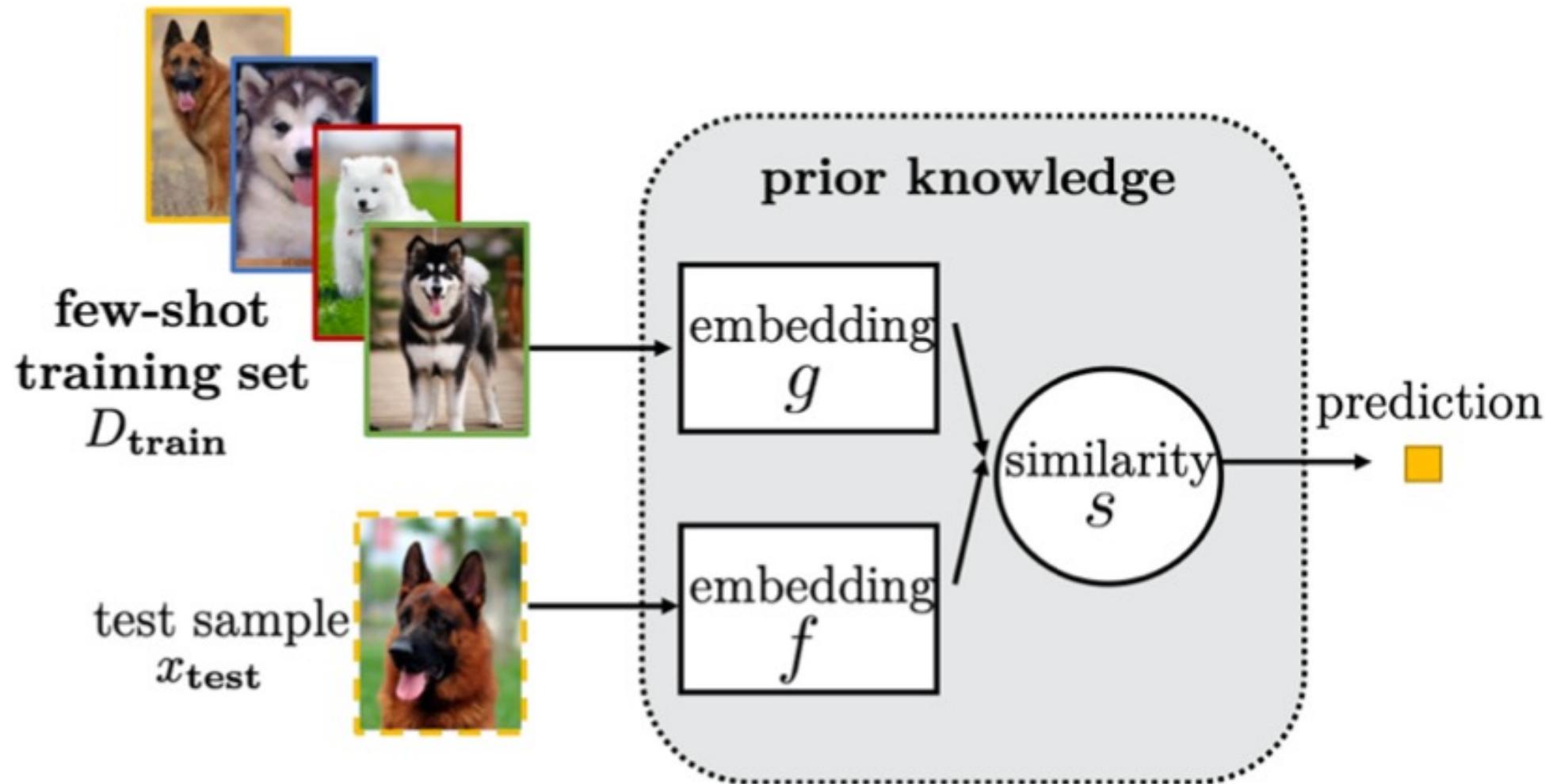
Few-Shot Learning (FSL)

Characteristics of Embedding Learning Methods

category	method	embedding function f for x_{test}	embedding function g for D_{train}	similarity measure s
task-specific	mAP-DLM/SSVM [130]	CNN	the same as f	cosine similarity
	class relevance pseudo-metric [36]	kernel	the same as f	squared ℓ_2 distance
	convolutional siamese net [70]	CNN	the same as f	weighted ℓ_1 distance
	Micro-Set [127]	logistic projection	the same as f	ℓ_2 distance
	Matching Nets [138]	CNN, LSTM	CNN, biLSTM	cosine similarity
	resLSTM [4]	GNN, LSTM	GNN, LSTM	cosine similarity
	Active MN [8]	CNN	biLSTM	cosine similarity
	SSMN [24]	CNN	another CNN	learned distance
	ProtoNet [121]	CNN	the same as f	squared ℓ_2 distance
	semi-supervised ProtoNet [108]	CNN	the same as f	squared ℓ_2 distance
task-invariant	PMN [141]	CNN, LSTM	CNN, biLSTM	cosine similarity
	ARC [119]	LSTM, biLSTM	the same as f	-
	Relation Net [126]	CNN	the same as f	-
	GNN [115]	CNN, GNN	the same as f	learned distance
	TPN [84]	CNN	the same as f	Gaussian similarity
	SNAIL [91]	CNN	the same as f	-
	Learnet [14]	adaptive CNN	CNN	weighted ℓ_1 distance
	DCCN [162]	adaptive CNN	CNN	-
	R2-D2 [13]	adaptive CNN	CNN	-
	TADAM [100]	adaptive CNN	the same as f	squared ℓ_2 distance
hybrid				

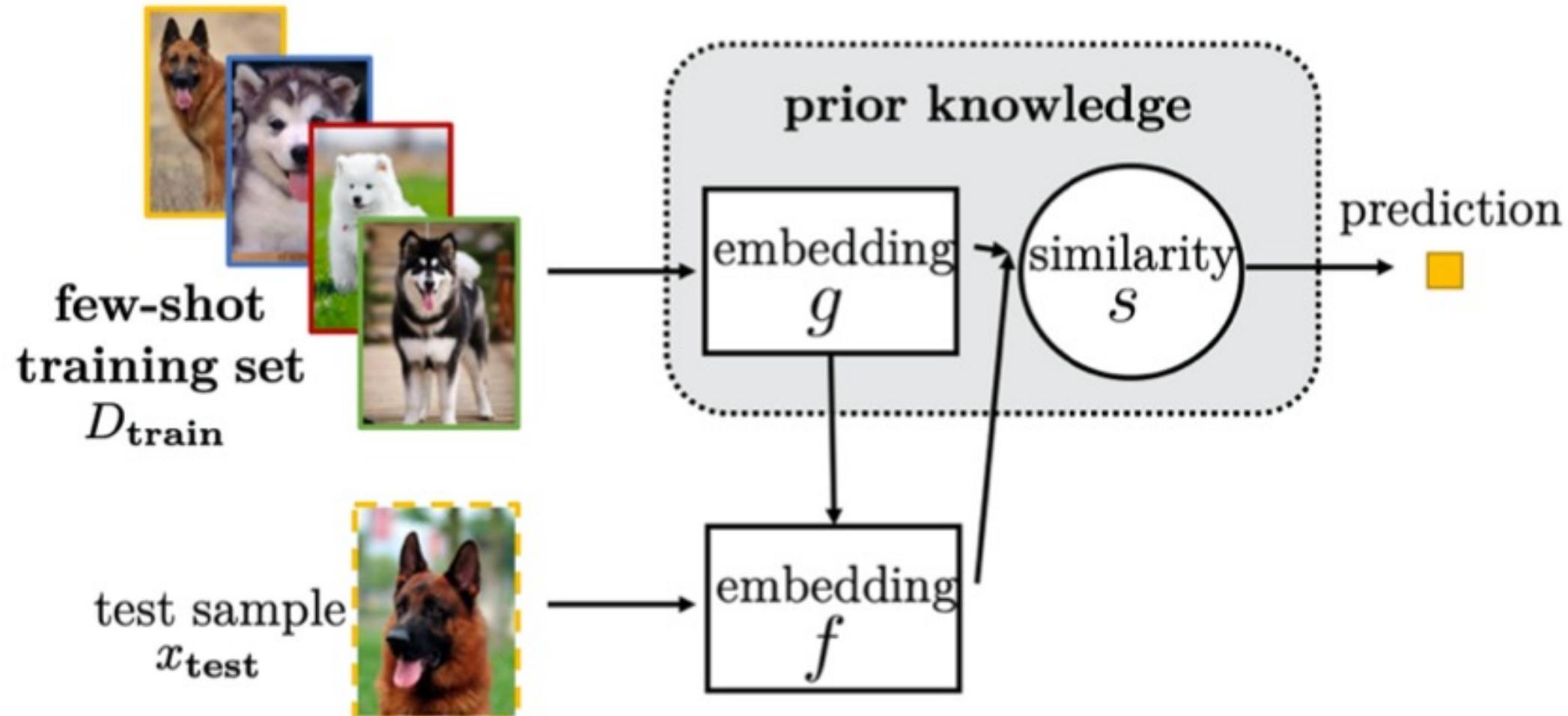
Few-Shot Learning (FSL)

Solving the FSL problem by task-invariant embedding model



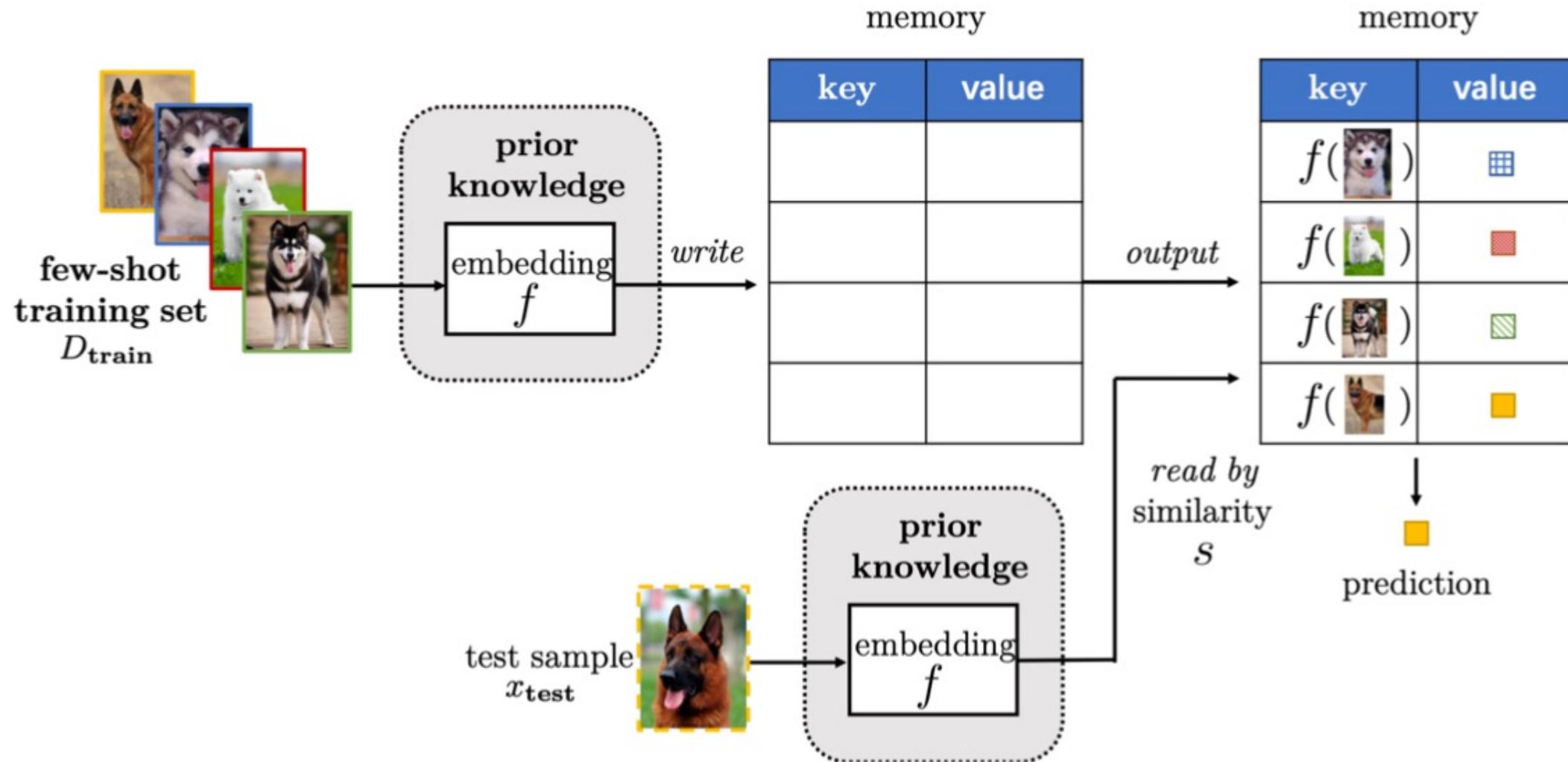
Few-Shot Learning (FSL)

Solving the FSL problem by hybrid embedding model



Few-Shot Learning (FSL)

Solving the FSL problem by learning with external memory



Few-Shot Learning (FSL)

Characteristics of FSL Methods

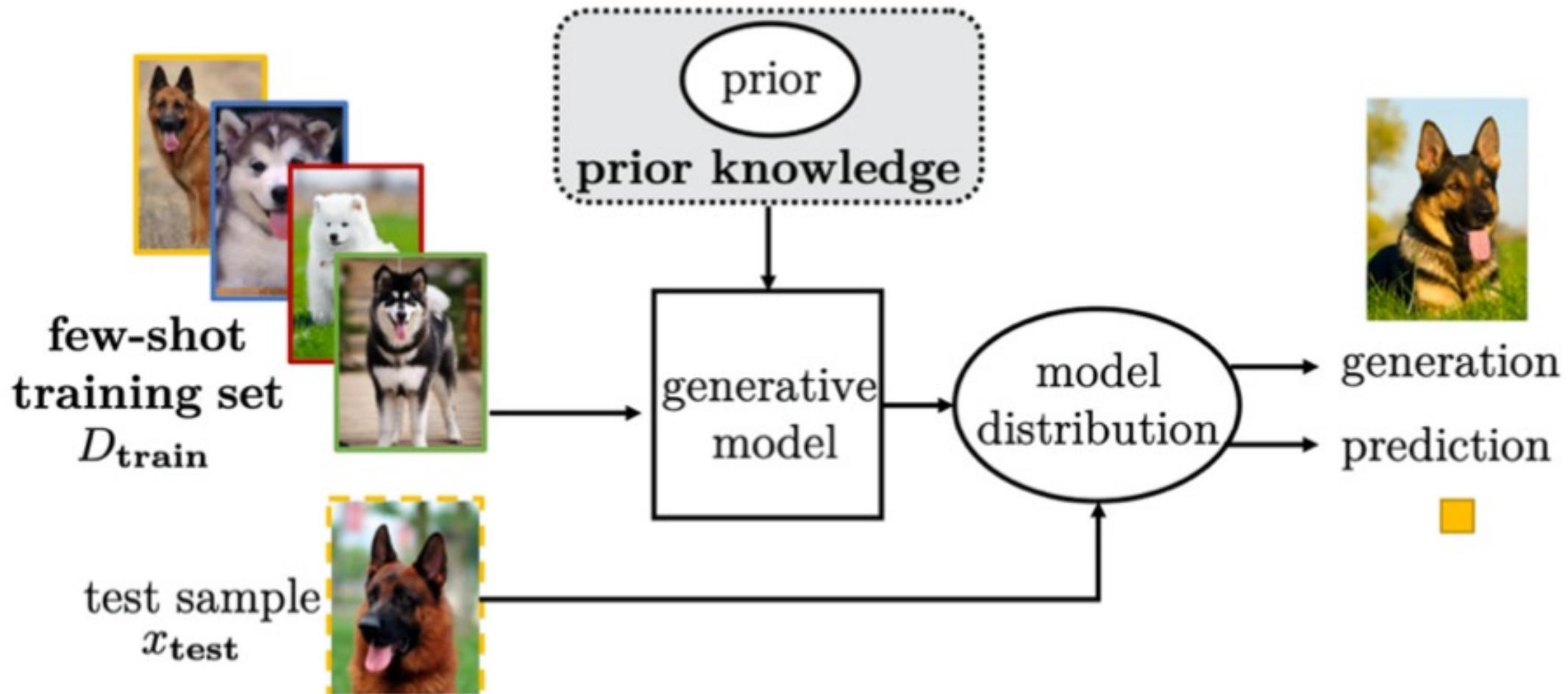
Based on Learning with External Memory

category	method	memory M		similarity s
		key M_{key}	value M_{value}	
refining representations	MANN [114]	$f(x_i, y_{i-1})$	$f(x_i, y_{i-1})$	cosine similarity
	APL [104]	$f(x_i)$	y_i	squared ℓ_2 distance
	abstraction memory [149]	$f(x_i)$	word embedding of y_i	dot product
	CMN [164]	$f(x_i)$	y_i , age	dot product
	life-long memory [65]	$f(x_i)$	y_i , age	cosine similarity
	Mem2Vec [125]	$f(x_i)$	word embedding of y_i , age	dot product
refining parameters	MetaNet [96]	$f(x_i)$	fast weight	cosine similarity
	CSNs [97]	$f(x_i)$	fast weight	cosine similarity
	MN-Net [22]	$f(x_i)$	y_i	dot product

Here, f is an embedding function usually pre-trained by CNN or LSTM.

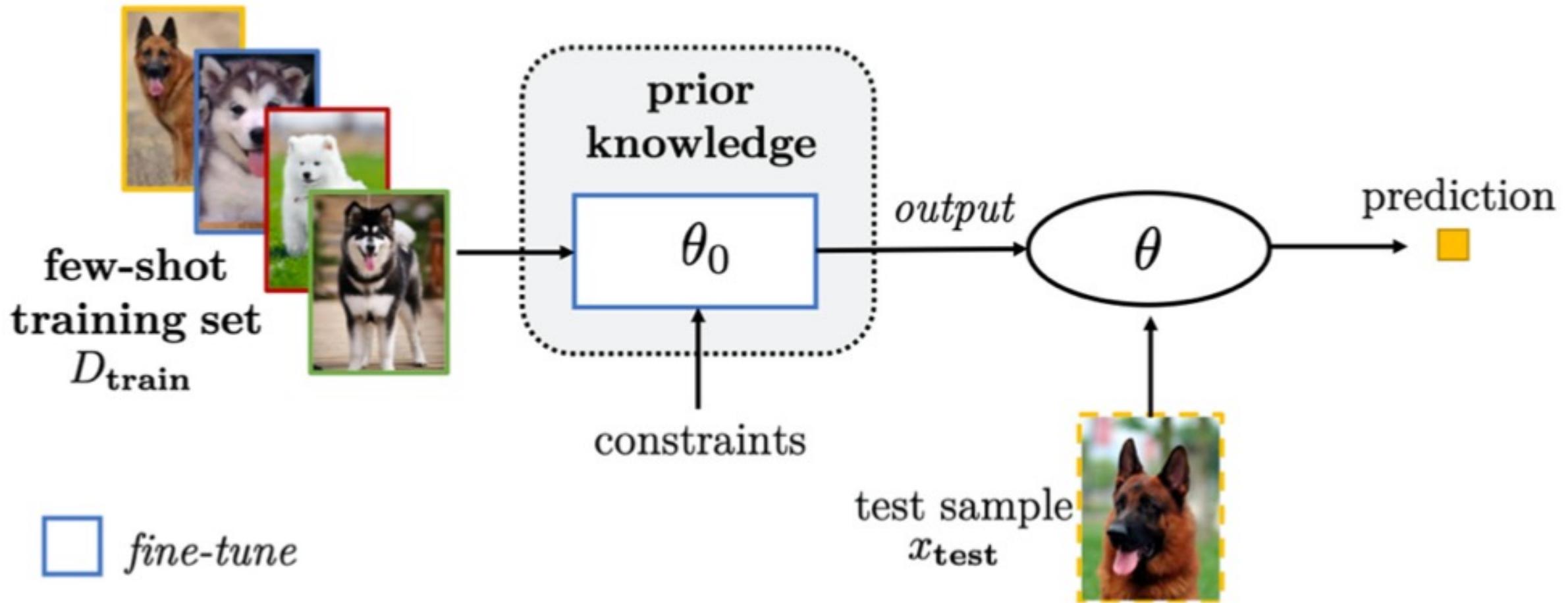
Few-Shot Learning (FSL)

Solving the FSL problem by generative modeling



Few-Shot Learning (FSL)

Solving the FSL problem by fine-tuning existing parameter θ_0 by regularization



Few-Shot Learning (FSL)

Characteristics for FSL Methods

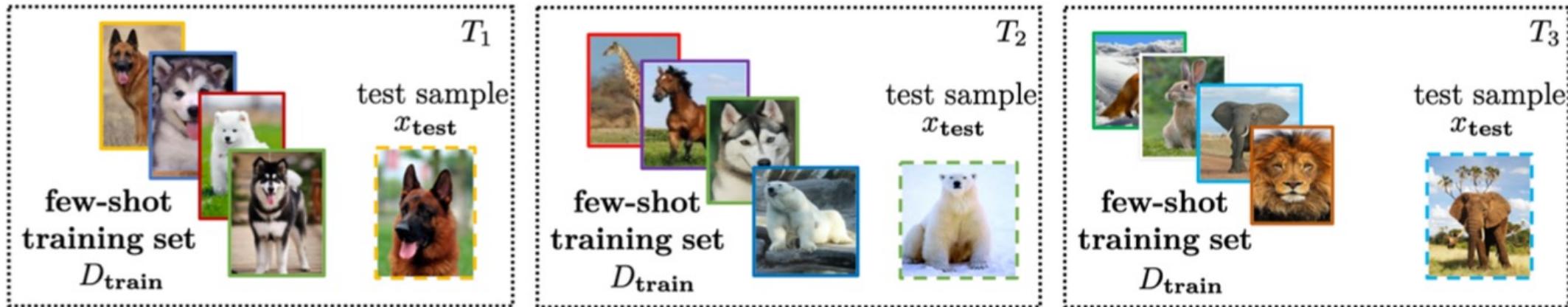
Focusing on the Algorithm Perspective

strategy	prior knowledge	how to search θ of the h^* in \mathcal{H}
refining existing parameters	learned θ_0	refine θ_0 by D_{train}
refining meta-learned parameters	meta-learner	refine θ_0 by D_{train}
learning the optimizer	meta-learner	use search steps provided by the meta-learner

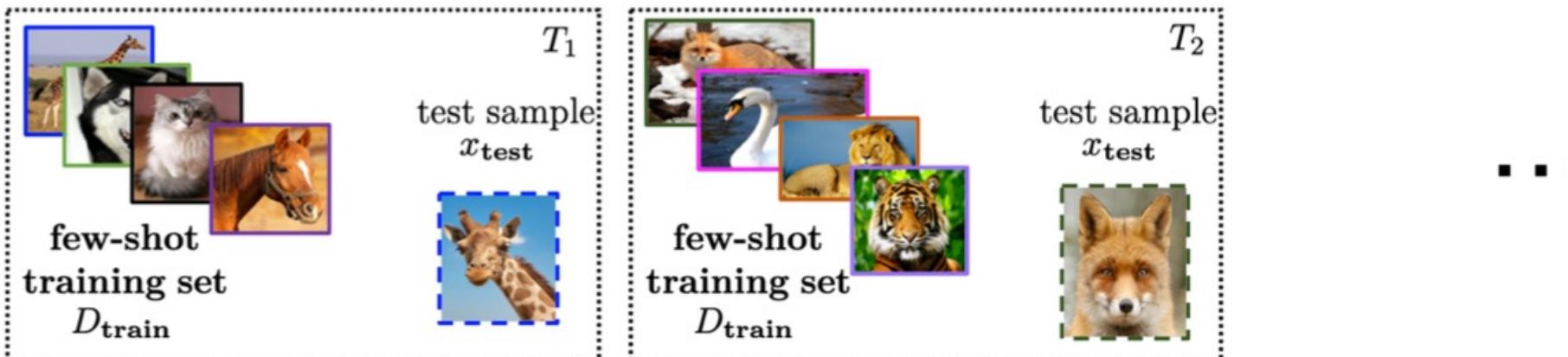
Few-Shot Learning (FSL)

Solving the FSL problem by meta-learning

meta-training tasks T_s 's



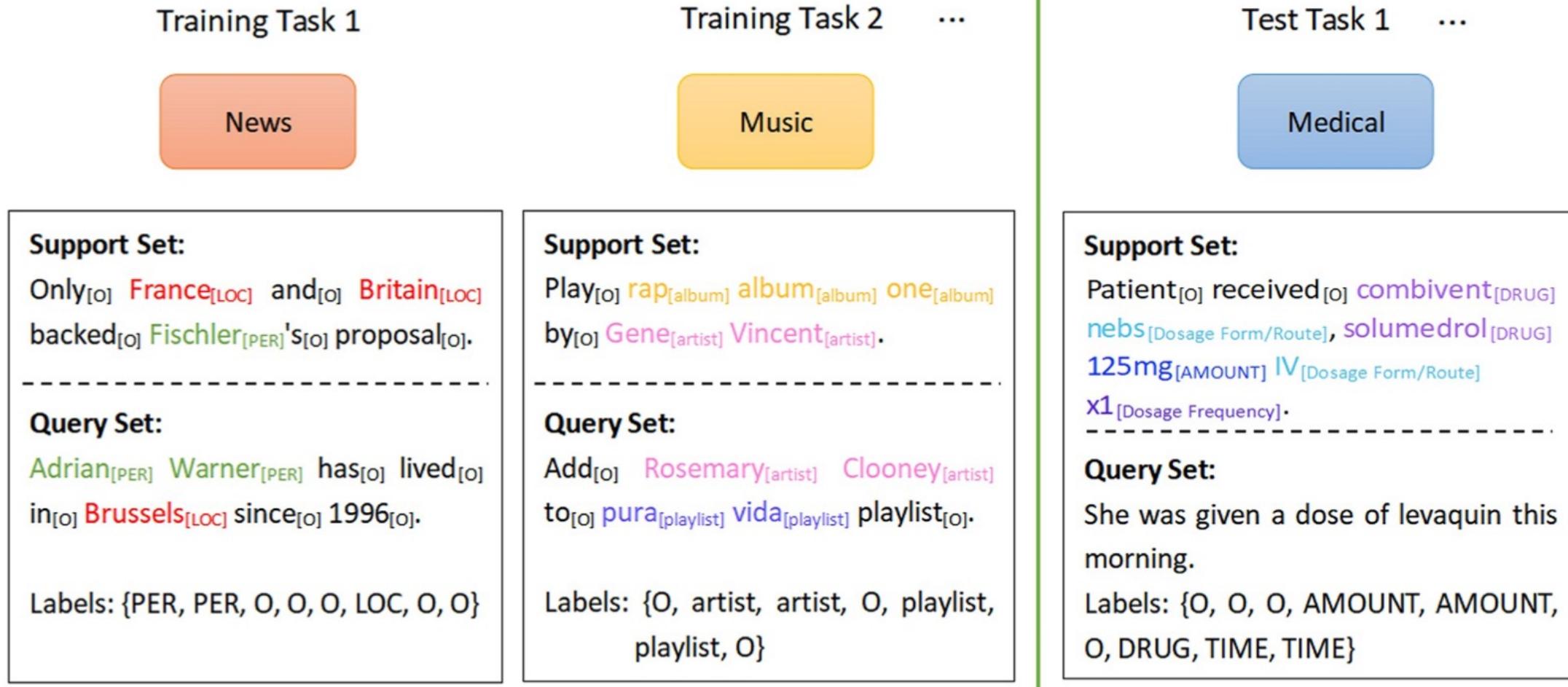
meta-testing tasks T_t 's



Few-Shot Learning (FSL)

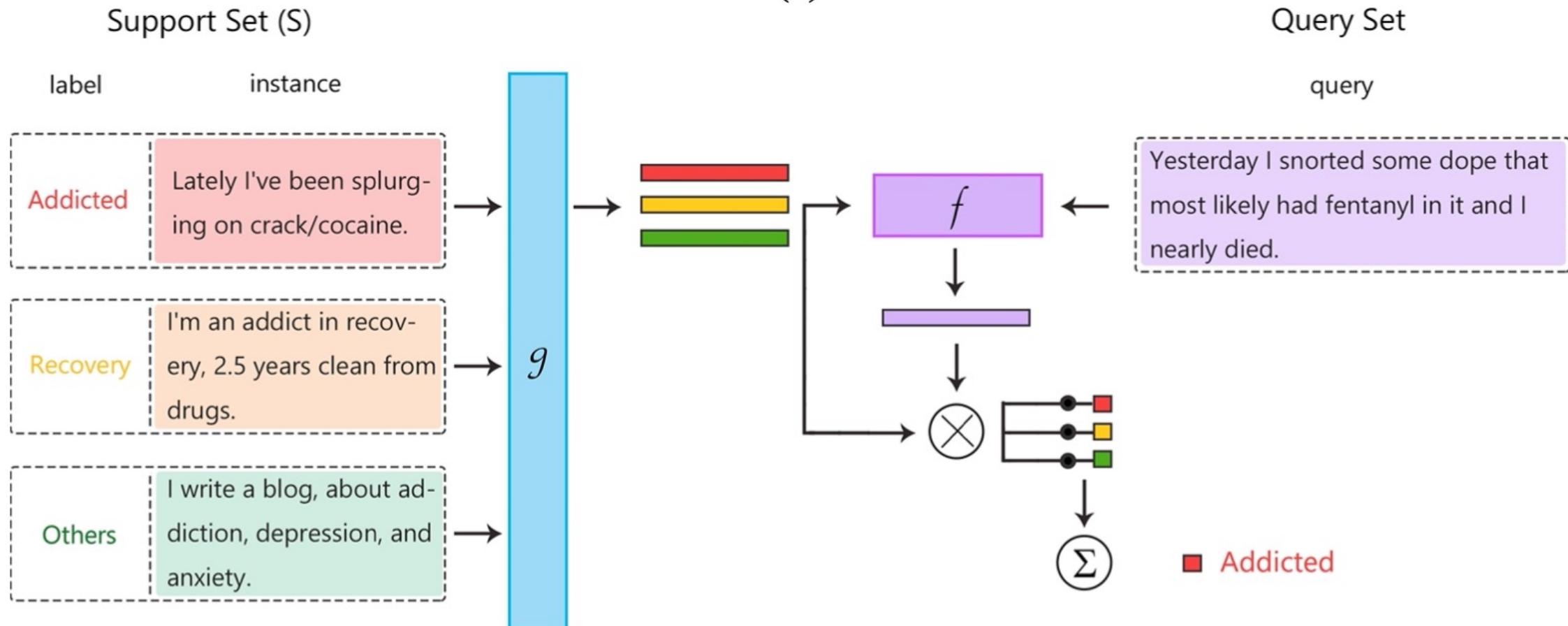
Meta-learning

Each task mimics the few-shot scenario, and can be completely non-overlapping.
Support sets are used to train; query sets are used to evaluate the model



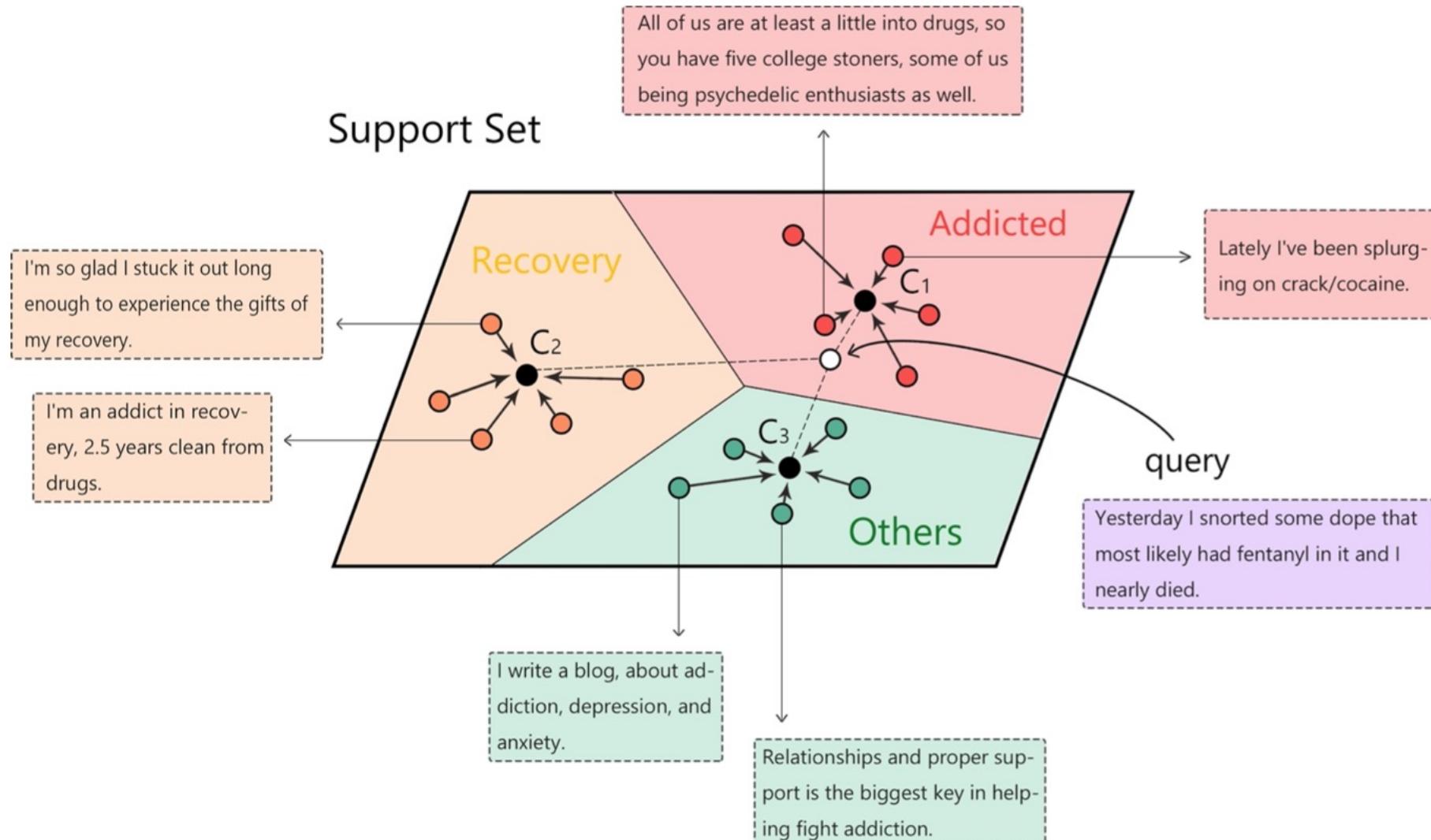
Few-Shot Learning (FSL)

Matching networks



Few-Shot Learning (FSL)

Prototypical network



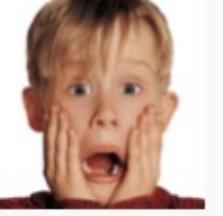
Few-Shot Learning (FSL) for medical text

Study	Year	Data source	Research aim	Size of training set	Number of entities / classes	Entity type of training domain	Entity type of test domain
Alicia Lara-Clares and Ana Garcia-Serrano ⁴⁴	2019	MEDDOCAN shared task dataset ⁴⁵	NER	500 clinical cases, with no reconstruction	29	Clinical	Clinical
Ferré et al. ⁴⁶	2019	BB-norm dataset from the Bacteria Biotope 2019 Task ⁴⁷	Entity Normalization	Original dataset with no reconstruction and zero-shot	Not mentioned *	Biological	Biological
Hou et al. ⁴⁸	2020	Snips dataset ⁴⁹	Slot Tagging (NER)	1-shot and 5-shot	7	Six of Weather, Music, PlayList, Book (including biomedical), Search Screen (including biomedical), Restaurant and Creative Work.	The remaining one
Sharaf et al. ⁵⁰	2020	ten different datasets collected from the Open Parallel Corpus (OPUS) ⁵¹	Neural Machine Translation (NMT)	Sizes ranging from 4k to 64k training words (200 to 3200 sentences), but reconstructed	N/A †	Bible, European Central Bank, KDE, Quran, WMT news test sets, Books, European Medicines Agency (EMEA), Global Voices, Medical (ufal-Med), TED talks	Bible, European Central Bank, KDE, Quran, WMT news test sets, Books, European Medicines Agency (EMEA), Global Voices, Medical (ufal-Med), TED talks
Lu et al. ⁵²	2020	MIMIC II ²² and MIMIC III ²³ , and EU legislation dataset ⁵³	Multi-label Text Classification	5-shot for MIMIC II and III, 50-shot for EU legislation	MIMIC II: 9 MIMIC III: 15 EU legislation: 5	Medical	Medical

Few-Shot Learning (FSL) for medical text

Study	Year	Data source	Research aim	Size of training set	Number of entities / classes	Entity type of training domain	Entity type of test domain
Lu et al. ⁸⁰	2021	Constructed and shared a novel dataset ^{††} based on Weibo for the research of few-shot rumor detection, and use PHEME dataset ⁸¹	Rumor Detection (NER)	For the Weibo dataset: 2-way 3-event 5-shot 9-query; for PHEME dataset: 2-way 2-event 5-shot 9-query	Weibo: 14 PHEME: 5	Source posts and comments from Sina Weibo related to COVID-19	Source posts and comments from Sina Weibo related to COVID-19
Ma et al. ⁸²	2021	CCLE, CERES-corrected CRISPR gene disruption scores, GDSC1000 dataset, PDTc dataset and PDX dataset ^{‡‡}	Drug-response Predictions	1-shot, 2-shot, 5-shot and 10-shot	N/A [†]	Biomedical	Biomedical
Kormilitzin et al. ⁸³	2021	MIMIC-III ²³ and UK-CRIS datasets ^{30,31}	NER	25%, 50%, 75% and 100% of the training set, with no reconstruction	7	Electronic health record	Electronic health record
Guo et al. ⁸⁴	2021	Abstracts of biomedical literatures (from relation extraction task of BioNLP Shared Task 2011 and 2019 ⁴⁷) and structured biological datasets	NER	100%, 75%, 50%, 25%, 0% of training set, with no reconstruction	Not mentioned *	Biomedical entities	Biomedical entities
Lee et al. ⁸⁵	2021	COVID19-Scientific ⁸⁶ , COVID19-Social ⁸⁷ (fact-checked by journalists from a website called Politi-fact.com), FEVER ⁸⁸ (Fact Extraction and Verification, generated by altering sentences extracted from Wikipedia to promote research on fact-checking systems)	Fact-Checking (close to Text Classification)	2-shot, 10-shot and 50-shot	Not mentioned *	Facts about COVID-19	Facts about COVID-19

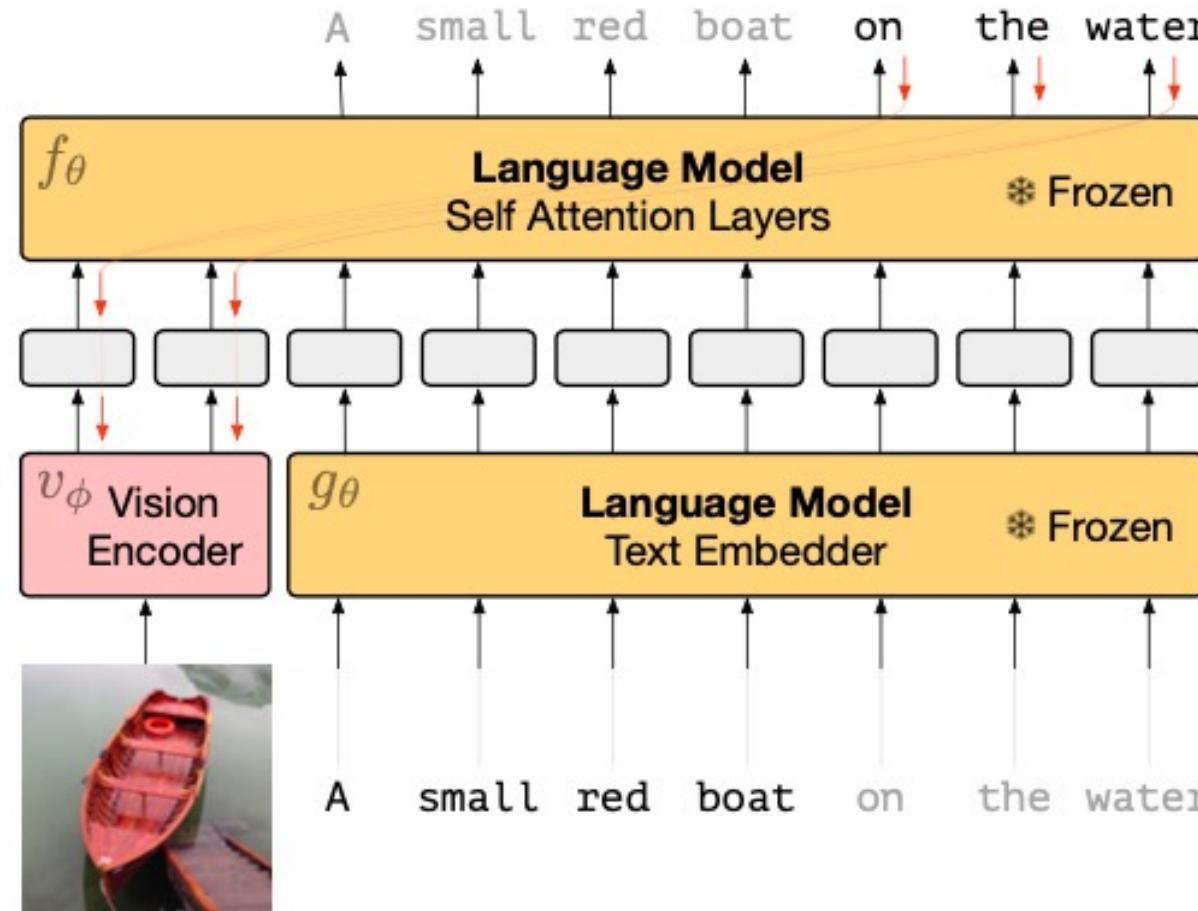
Multimodal Few-Shot Learning with Frozen Language Models

	This person is like 😊.		This person is like 😒.		This person is like	Model Completion 选拨. <EOS>
	This was invented by Zacharias Janssen.		This was invented by Thomas Edison.		This was invented by	Model Completion the Wright brothers. <EOS>
	With one of these I can drive around a track, overtaking other cars and taking corners at speed		With one of these I can take off from a city and fly across the sky to somewhere on the other side of the world		With one of these I can	Model Completion break into a secure building, unlock the door and walk right in <EOS>

Curated samples with about five seeds required to get past well-known language model failure modes of either repeating text for the prompt or emitting text that does not pertain to the image.

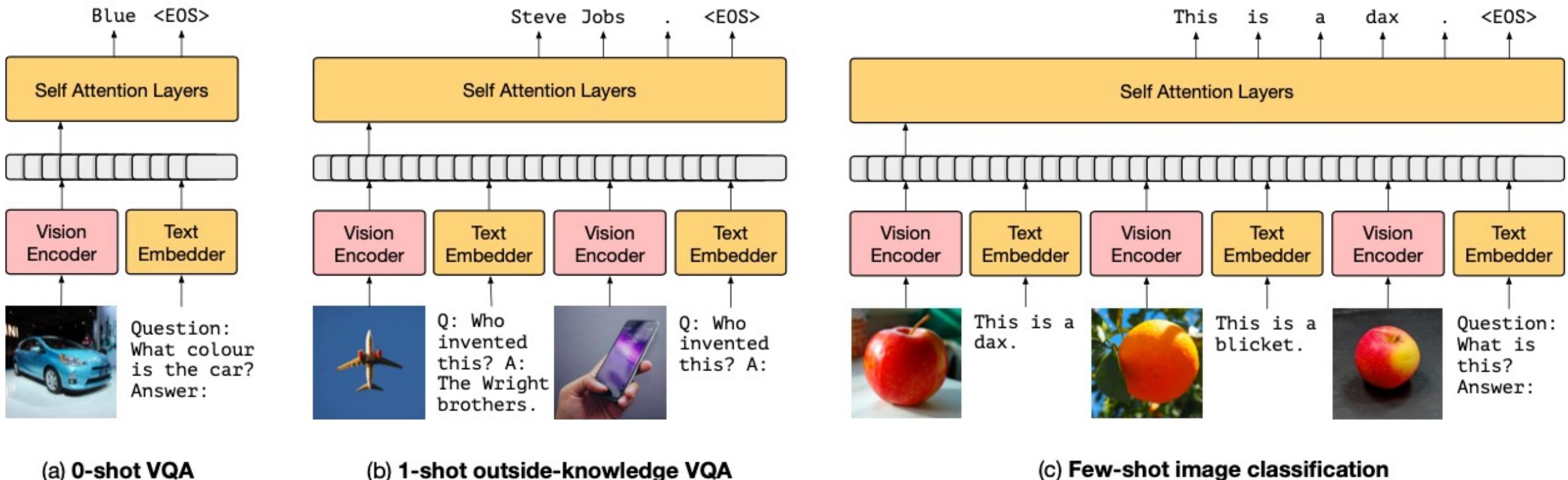
These samples demonstrate the ability to generate open-ended outputs that adapt to both images and text, and to make use of facts that it has learned during language-only pre-training.

Multimodal Few-Shot Learning with Frozen Language Models



Gradients through a frozen language model's self attention layers are used to train the vision encoder.

Multimodal Few-Shot Learning with Frozen Language Models



Inference-Time interface for *Frozen*. The figure demonstrates how we can support (a) visual question answering, (b) outside-knowledge question answering and (c) few-shot image classification via in-context learning.

Source: Maria Tsimpoukelli, Jacob L. Menick, Serkan Cabi, S. M. Eslami, Oriol Vinyals, and Felix Hill (2021). "Multimodal few-shot learning with frozen language models."

Advances in Neural Information Processing Systems 34 (2021): 200-212.

Multimodal Few-Shot Learning with Frozen Language Models

(a) minilmageNet



(b) Fast VQA



Examples of (a) the Open-Ended minilmageNet evaluation (b) the Fast VQA evaluation.

Source: Maria Tsimpoukelli, Jacob L. Menick, Serkan Cabi, S. M. Eslami, Oriol Vinyals, and Felix Hill (2021). "Multimodal few-shot learning with frozen language models."

Advances in Neural Information Processing Systems 34 (2021): 200-212.

GPT-3: Language Models are Few-Shot Learners

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

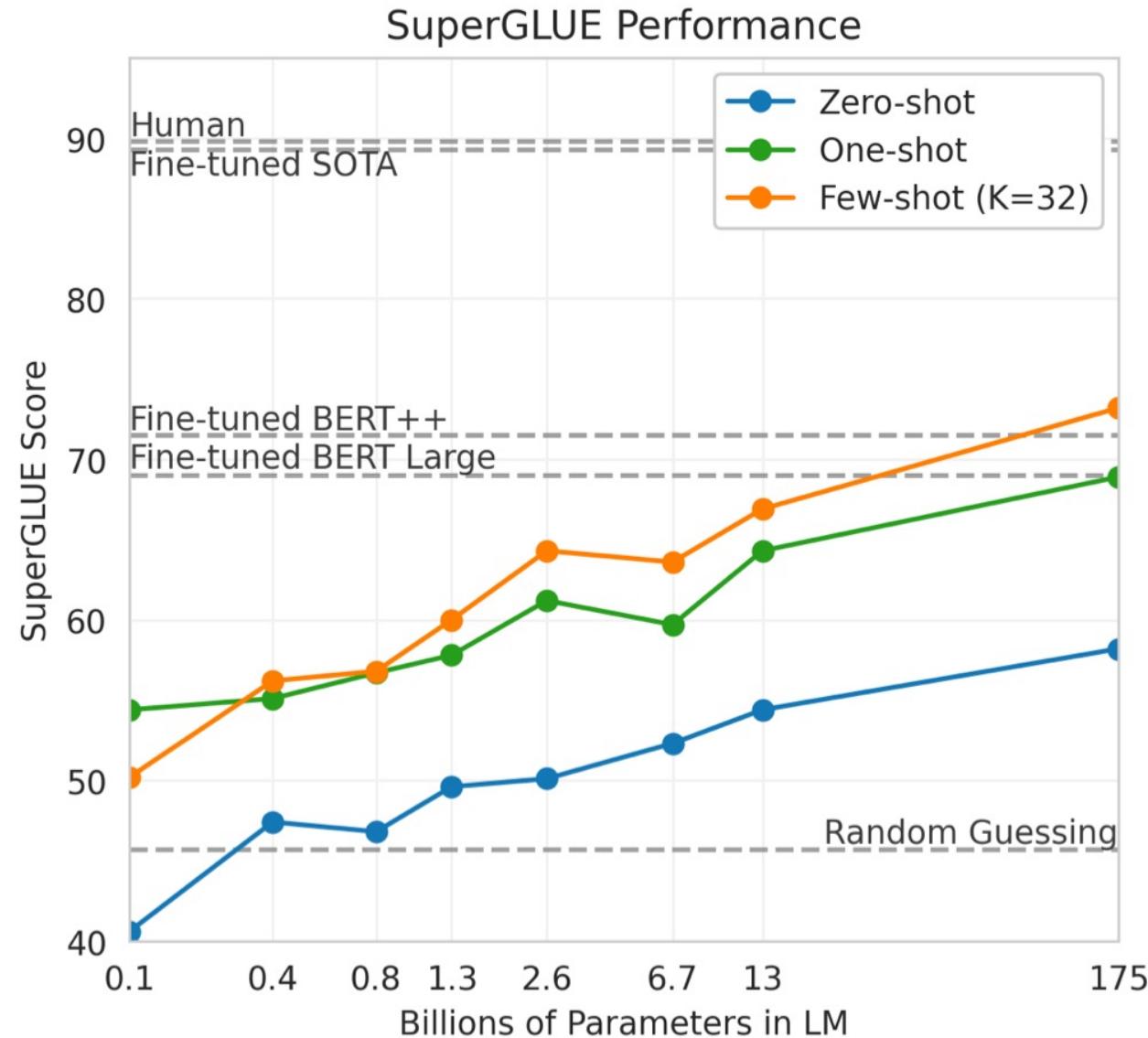
Alec Radford

Ilya Sutskever

Dario Amodei

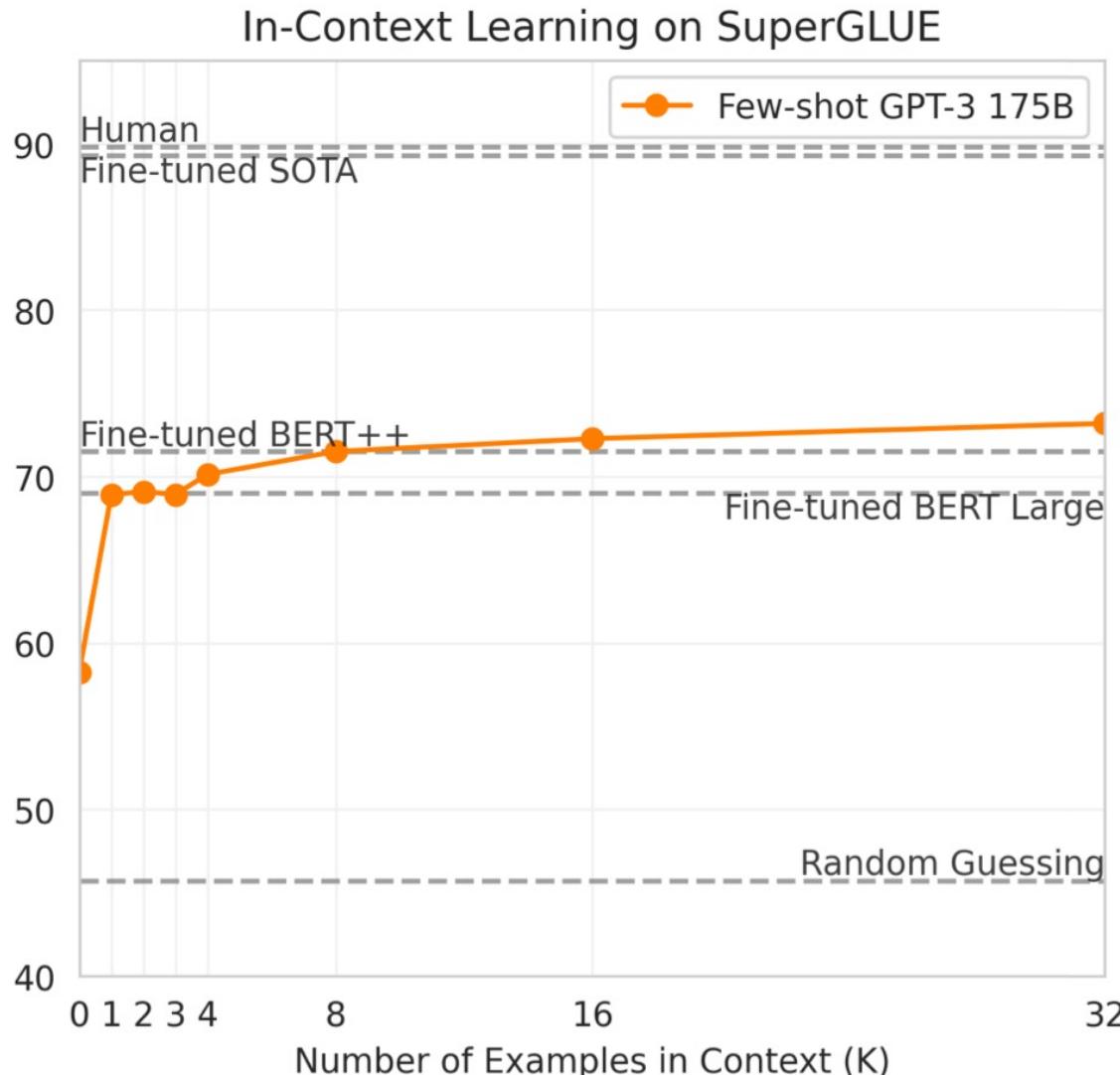
This work was funded by [OpenAI](#). All models were trained on [V100](#) GPU's on part of a high-bandwidth cluster provided by Microsoft.

GPT-3: Language Models are Few-Shot Learners



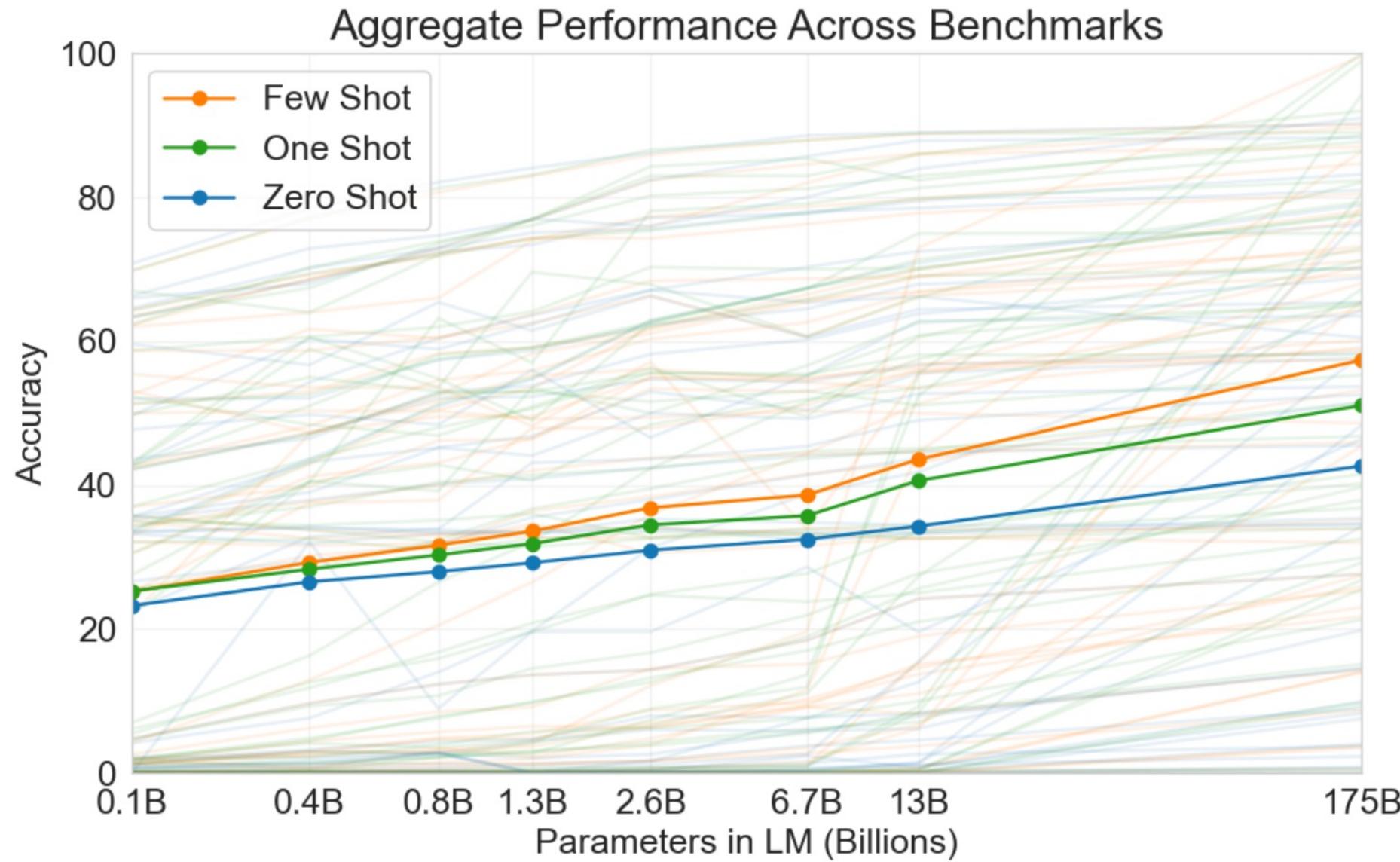
Source: Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan et al. (2020) "Language models are few-shot learners." Advances in neural information processing systems 33 (2020): 1877-1901

GPT-3: Language Models are Few-Shot Learners



Performance on SuperGLUE increases with model size. A value of $K = 32$ means that our model was shown 32 examples per task, for 256 examples total divided across the 8 tasks in SuperGLUE.

GPT-3: Language Models are Few-Shot Learners



Source: Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan et al. (2020) "Language models are few-shot learners." Advances in neural information processing systems 33 (2020): 1877-1901

GPT-3: Language Models are Few-Shot Learners

Performance on cloze and completion tasks.

Setting	LAMBADA (acc)	LAMBADA (ppl)	StoryCloze (acc)	HellaSwag (acc)
SOTA	68.0 ^a	8.63 ^b	91.8 ^c	85.6 ^d
GPT-3 Zero-Shot	76.2	3.00	83.2	78.9
GPT-3 One-Shot	72.5	3.35	84.7	78.1
GPT-3 Few-Shot	86.4	1.92	87.7	79.3

GPT-3 significantly improves SOTA on LAMBADA while achieving respectable performance on two difficult completion prediction datasets.

GPT-3: Language Models are Few-Shot Learners

Results on three Open-Domain QA tasks

Setting		NaturalQS	WebQS	TriviaQA
RAG (Fine-tuned, Open-Domain) [LPP ⁺ 20]	44.5	45.5	68.0	
T5-11B+SSM (Fine-tuned, Closed-Book) [RRS20]	36.6	44.7	60.5	
T5-11B (Fine-tuned, Closed-Book)	34.5	37.4	50.1	
GPT-3 Zero-Shot	14.6	14.4	64.3	
GPT-3 One-Shot	23.0	25.3	68.0	
GPT-3 Few-Shot	29.9	41.5	71.2	

GPT-3 is shown in the few-, one-, and zero-shot settings, as compared to prior SOTA results for closed book and open domain settings.

TriviaQA few-shot result is evaluated on the wiki split test server.

GPT-3: Language Models are Few-Shot Learners

GPT-3 results on a selection of QA / RC tasks.

Setting	ARC (Easy)	ARC (Challenge)	CoQA	DROP
Fine-tuned SOTA	92.0^a	78.5^b	90.7^c	89.1^d
GPT-3 Zero-Shot	68.8	51.4	81.5	23.6
GPT-3 One-Shot	71.2	53.2	84.0	34.3
GPT-3 Few-Shot	70.1	51.5	85.0	36.5

CoQA and DROP are F1 while ARC reports accuracy.

See the appendix for additional experiments. a[KKS+20] b[KKS+20] c[JZC+19] d [JN20]

GPT-3: Language Models are Few-Shot Learners

Setting	En→Fr	Fr→En	En→De	De→En	En→Ro	Ro→En
SOTA (Supervised)	45.6^a	35.0 ^b	41.2^c	40.2 ^d	38.5^e	39.9^e
XLM [LC19]	33.4	33.3	26.4	34.3	33.3	31.8
MASS [STQ ⁺ 19]	<u>37.5</u>	34.9	28.3	35.2	<u>35.2</u>	33.1
mBART [LGG ⁺ 20]	-	-	<u>29.8</u>	34.0	<u>35.0</u>	30.5
GPT-3 Zero-Shot	25.2	21.2	24.6	27.2	14.1	19.9
GPT-3 One-Shot	28.3	33.7	26.2	30.4	20.6	38.6
GPT-3 Few-Shot	32.6	<u>39.2</u>	29.7	<u>40.6</u>	21.0	<u>39.5</u>

Few-shot GPT-3 outperforms previous unsupervised NMT work by 5 BLEU when translating into English reflecting its strength as an English LM.

GPT-3: Language Models are Few-Shot Learners

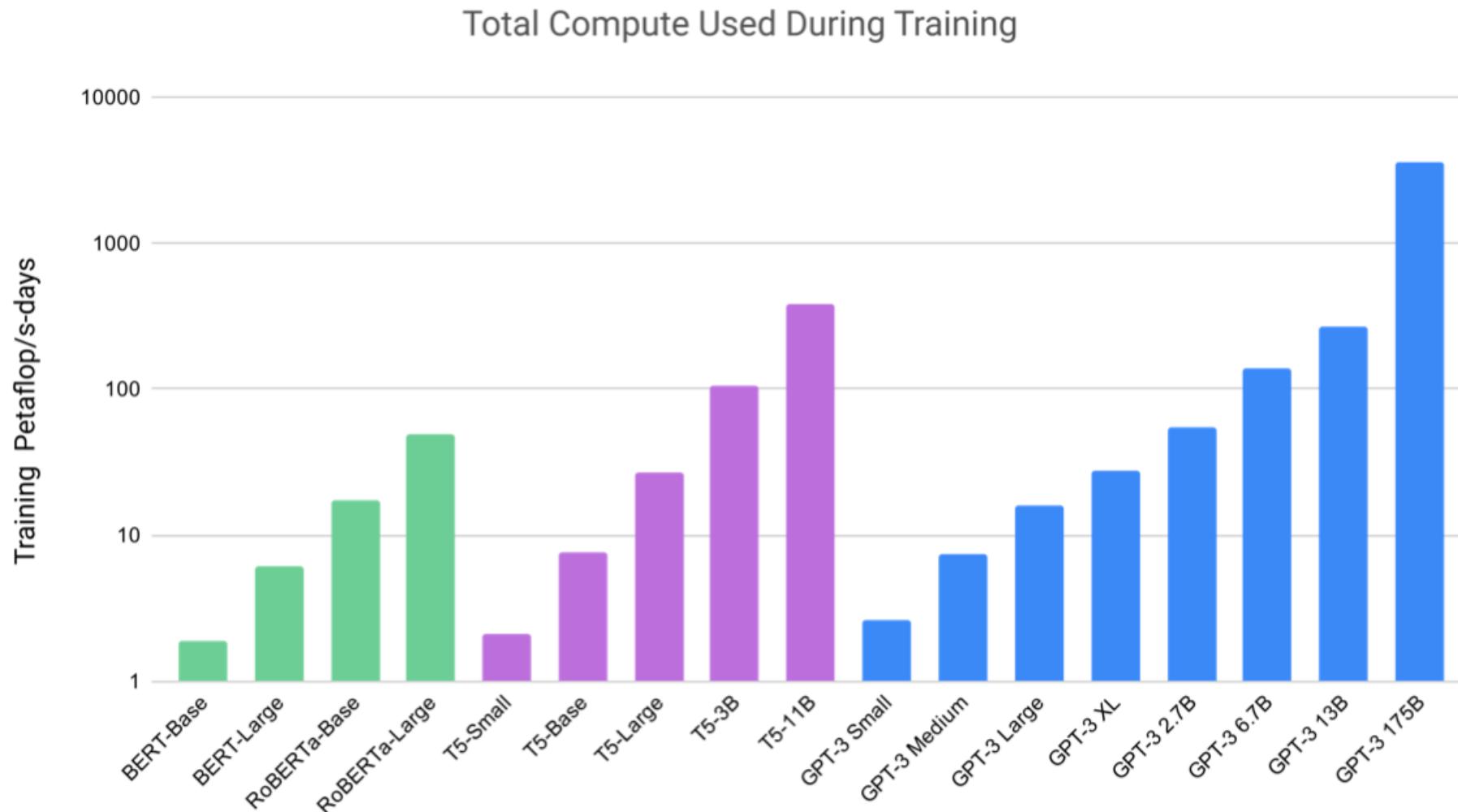
Performance of GPT-3 on SuperGLUE compared to fine-tuned baselines and SOTA

	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

GPT-3 few-shot is given a total of 32 examples within the context of each task and performs no gradient updates.

GPT-3: Language Models are Few-Shot Learners



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

GPT-3: Language Models are Few-Shot Learners

Human accuracy in identifying
whether short (~200 word) news articles are model generated

	Mean accuracy	95% Confidence Interval (low, hi)	t compared to control (p-value)	“I don’t know” assignments
Control (deliberately bad model)	86%	83%–90%	-	3.6 %
GPT-3 Small	76%	72%–80%	3.9 (2e-4)	4.9%
GPT-3 Medium	61%	58%–65%	10.3 (7e-21)	6.0%
GPT-3 Large	68%	64%–72%	7.3 (3e-11)	8.7%
GPT-3 XL	62%	59%–65%	10.7 (1e-19)	7.5%
GPT-3 2.7B	62%	58%–65%	10.4 (5e-19)	7.1%
GPT-3 6.7B	60%	56%–63%	11.2 (3e-21)	6.2%
GPT-3 13B	55%	52%–58%	15.3 (1e-32)	7.1%
GPT-3 175B	52%	49%–54%	16.9 (1e-34)	7.8%

This table compares mean accuracy between five different models, and shows the results of a two-sample T-Test for the difference in mean accuracy between each model and the control model (an unconditional GPT-3 Small model with increased output randomness).

GPT-3: Language Models are Few-Shot Learners

The GPT-3 generated news article that humans had the greatest difficulty distinguishing from a human written article (accuracy: 12%)

Title: United Methodists Agree to Historic Split

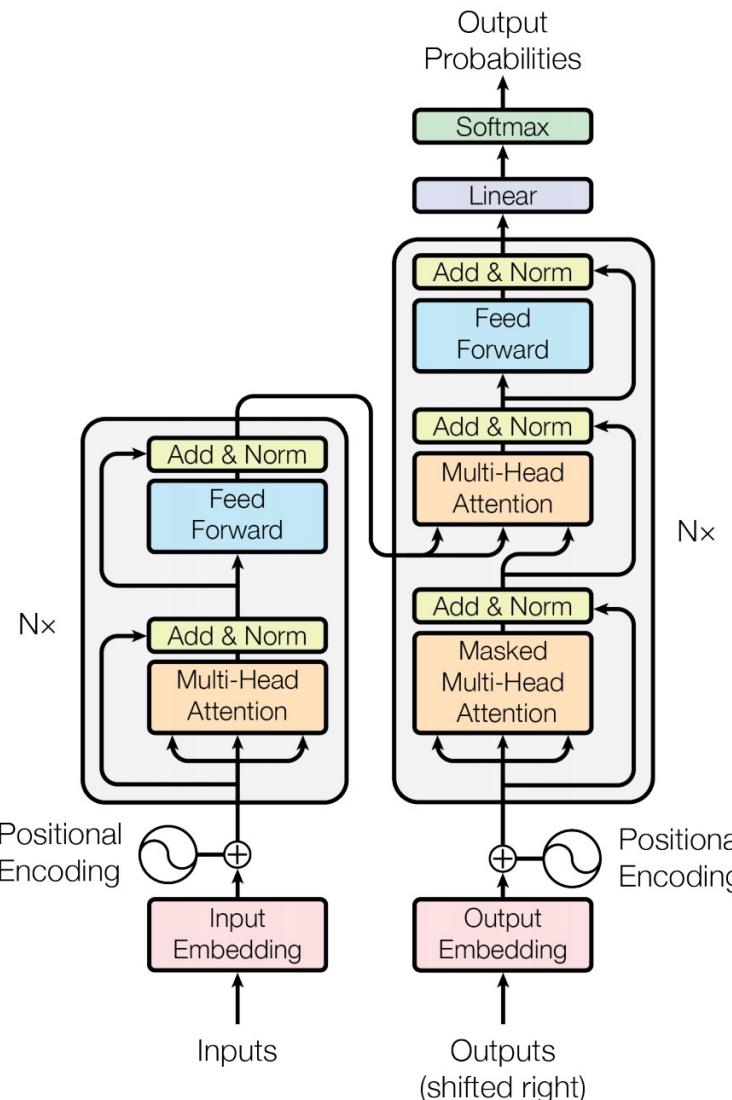
Subtitle: Those who oppose gay marriage will form their own denomination

Article: After two days of intense debate, the United Methodist Church has agreed to a historic split - one that is expected to end in the creation of a new denomination, one that will be "theologically and socially conservative," according to The Washington Post. The majority of delegates attending the church's annual General Conference in May voted to strengthen a ban on the ordination of LGBTQ clergy and to write new rules that will "discipline" clergy who officiate at same-sex weddings. But those who opposed these measures have a new plan: They say they will form a separate denomination by 2020, calling their church the Christian Methodist denomination.

The Post notes that the denomination, which claims 12.5 million members, was in the early 20th century the "largest Protestant denomination in the U.S.," but that it has been shrinking in recent decades. The new split will be the second in the church's history. The first occurred in 1968, when roughly 10 percent of the denomination left to form the Evangelical United Brethren Church. The Post notes that the proposed split "comes at a critical time for the church, which has been losing members for years," which has been "pushed toward the brink of a schism over the role of LGBTQ people in the church." Gay marriage is not the only issue that has divided the church. In 2016, the denomination was split over ordination of transgender clergy, with the North Pacific regional conference voting to ban them from serving as clergy, and the South Pacific regional conference voting to allow them.

Transformer (Attention is All You Need)

(Vaswani et al., 2017)

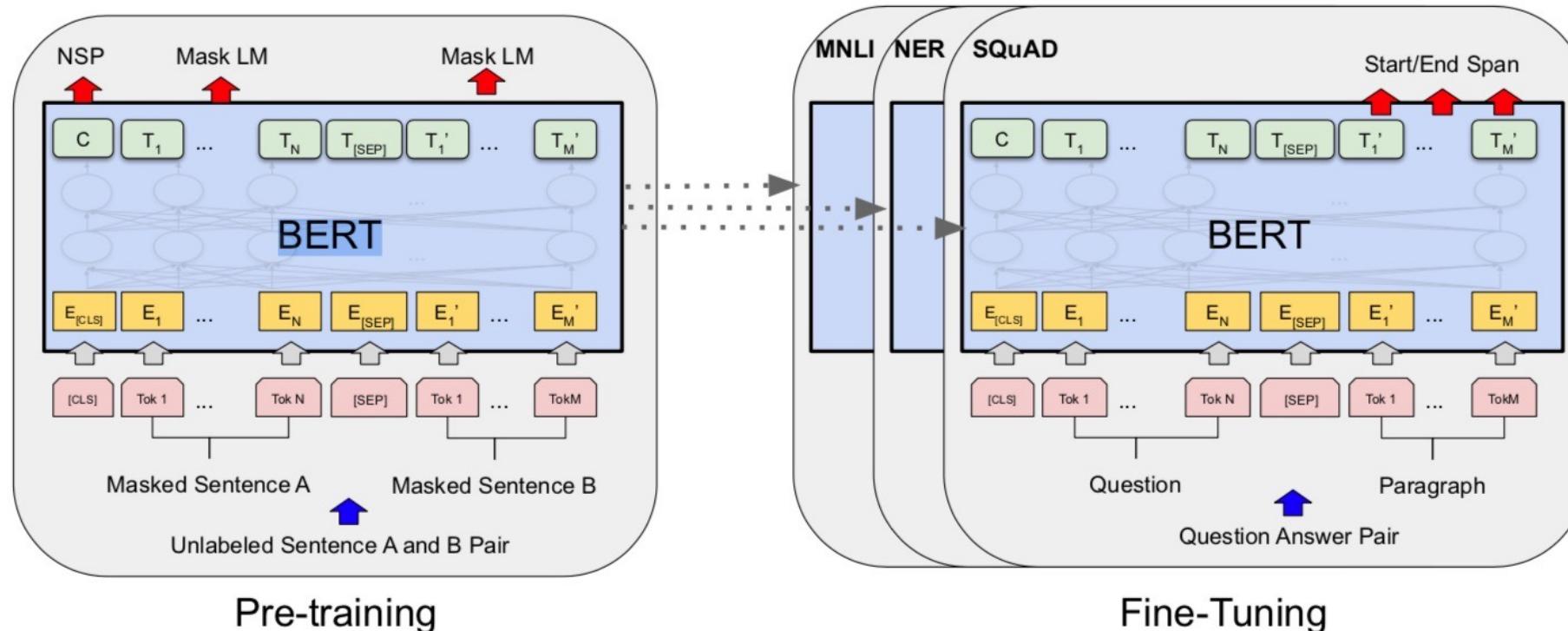


Source: Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin.
"Attention is all you need." In *Advances in neural information processing systems*, pp. 5998-6008. 2017.

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

BERT (Bidirectional Encoder Representations from Transformers)

Overall pre-training and fine-tuning procedures for BERT

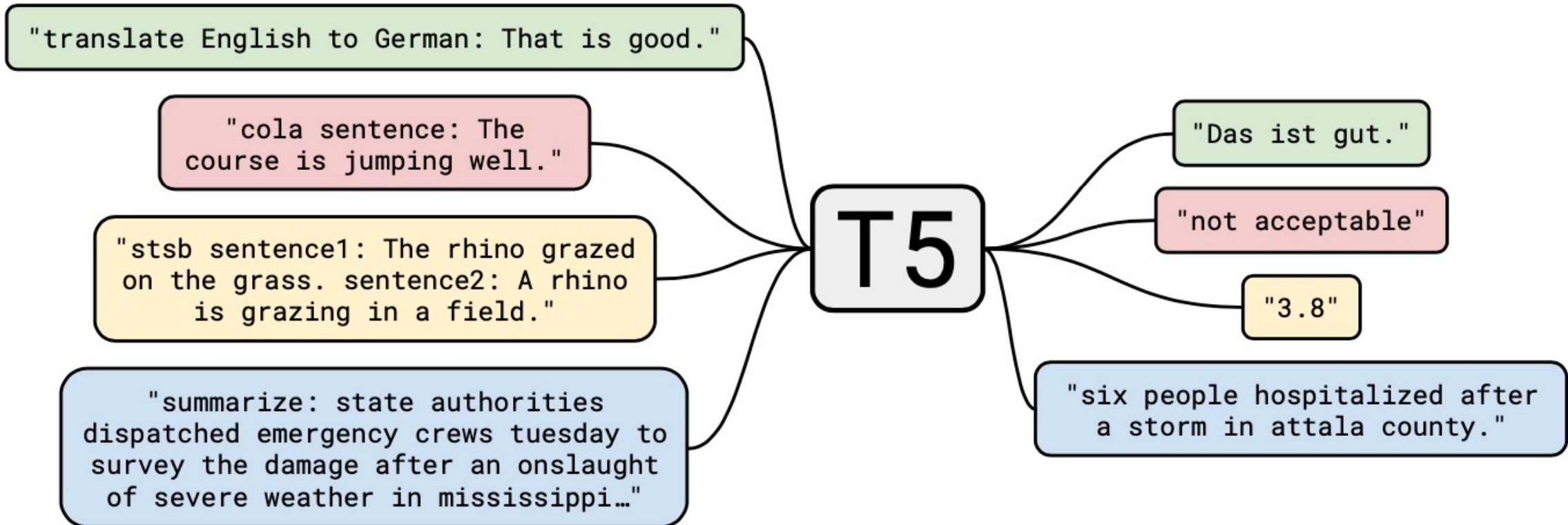


Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).

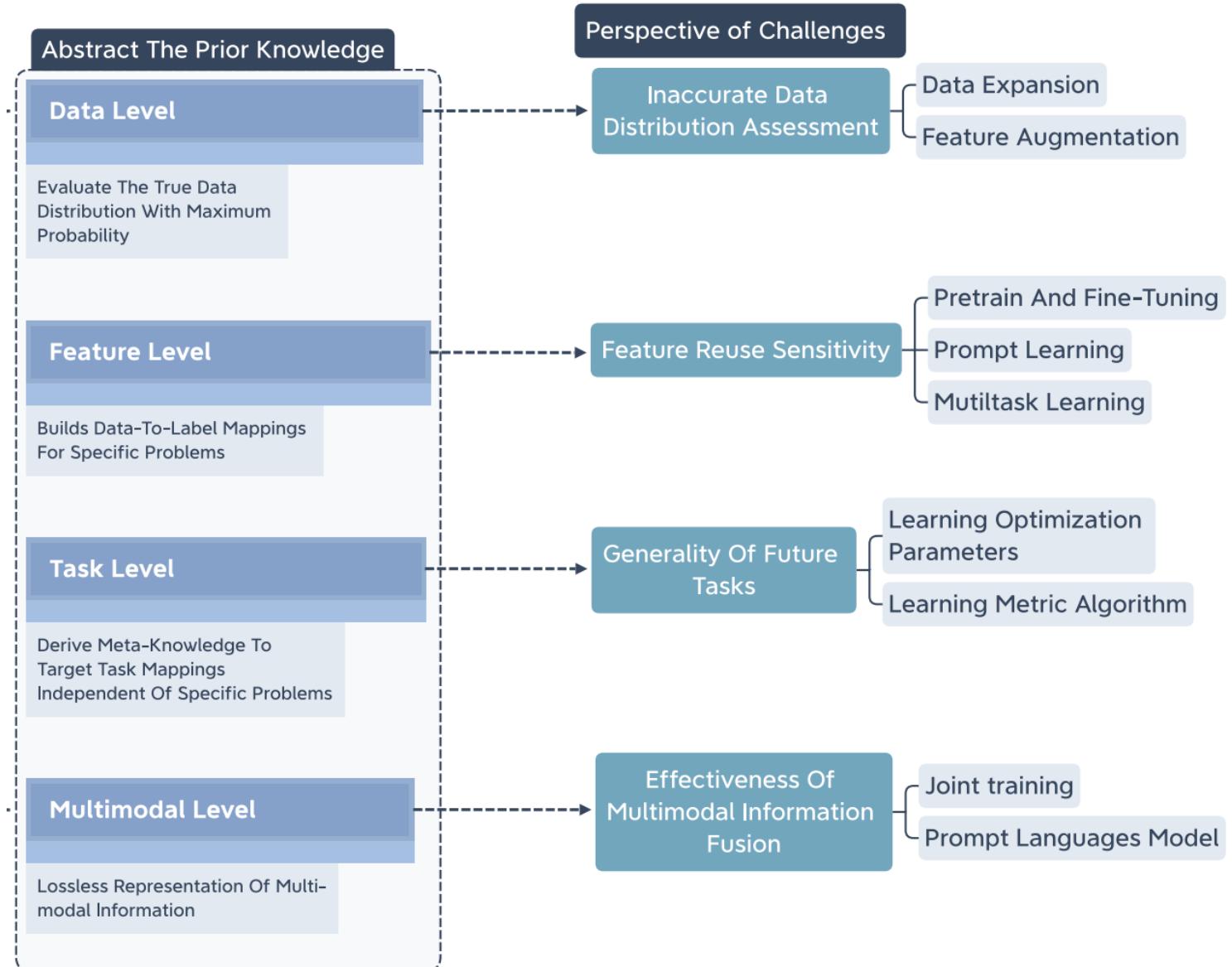
"Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805.

T5

Text-to-Text Transfer Transformer



Few-Shot Learning (FSL)



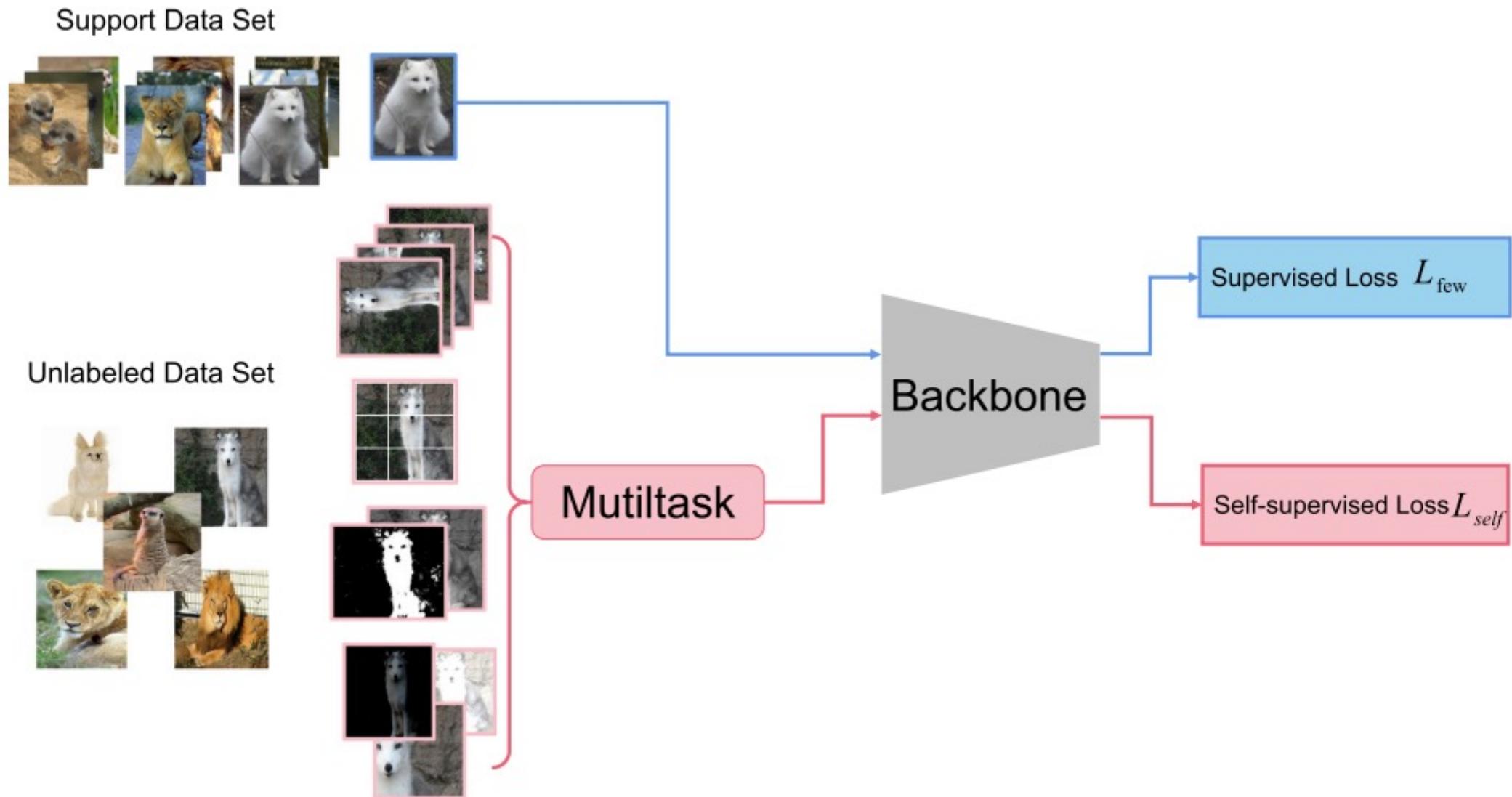
Pre-training and Fine-tuning Methods

Method	Venue	Backbone	Parameters	Tricks		
				Refining Parameters	Knowledge distillation	Attention
Squad [101]	EMNLP'16	Logistic Regression	-	✓		
Attention is All you Need [102]	NIPS'17	Transformer	110M			✓
Baseline [92]	ICLR'19	Conv64F	0.22M			
Baseline++ [92]	ICML'19	Conv64F	0.22M		✓	✓
CAN [98]	NIPS'19	ResNet12	12.47M			✓
GLUE [103]	ICLR'19	ELMo	94M	✓		
RFS-distill [95]	ECCV'20	ResNet-12	12.47M		✓	
LEE et al. [104]	WPI'20	GPT-2	1500M	✓		
CTX [97]	NIPS'20	Transformer	110M		✓	
SKD [96]	CVPR'20	ResNet12	12.47M			✓
P-Transfer [93]	AAAI'21	ResNet-12	12.47M	✓		
BERT Fine-tuning [105]	CL'21	BERT	110M	✓		✓
COSOC [106]	NIPS'21	ResNet-12	12.47M			✓
P>M>F [100]	CVPR'22	ViT	307M			✓
ReFine [94]	CVPR'22	ResNet-10	9.86M	✓		

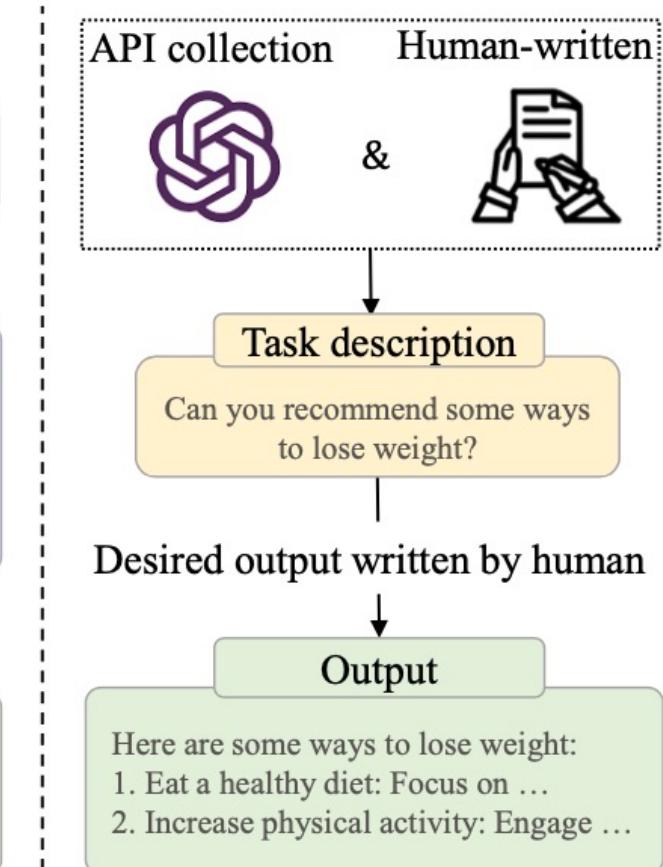
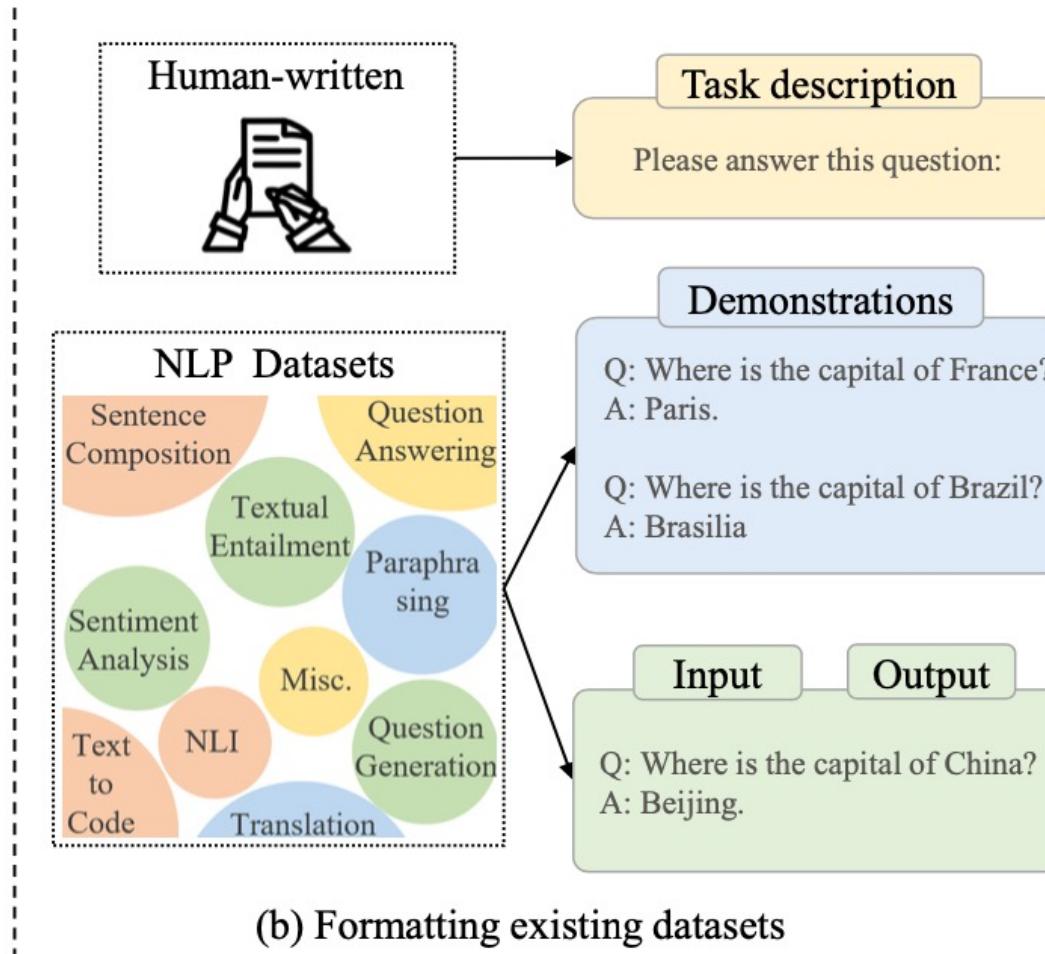
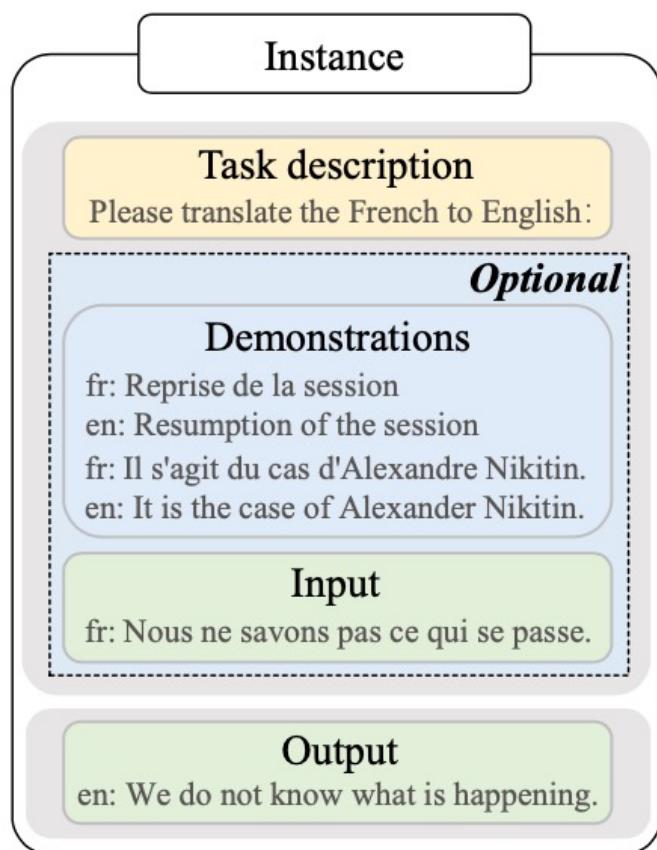
Few-Shot Learning (FSL) Prompt Learning in Natural Language Processing

Method	Venue	Backbone	Type	Shape	Highlight
PT [110]	ACL'21	GPT-3	Manual prompt	Prefix Prompt	Combine input vector and prefix prompts
Brown [88]	NIPS'20	GPT-3	Manual Prompt	Prefix Prompt	Design task-related prompt
Petroni et al. [87]	EMNLP'19	BERT	Manual Prompt	Cloze-style	Use cloze-style prompt to learn prior knowledge
Cui et al. [107]	ACL-IJCNLP'21	BERT	Manual prompt	Cloze-style	Use cloze to construct tasks in entity classification
PET [112]	EACL'21	PLM	Manual Prompt	Cloze-style	Introduces fine-tuning and knowledge distillation
LPAQA [113]	MIT'20	LMS	Auto-Prompt	Cloze-style	Collect manual templates as template library
AutoPrompt [114]	ACL'20	MLMs	Auto-Prompt	Cloze-style	Automatically search template by collect prior prompt
LM-BFF [115]	ACL'21	GPT-3	Auto-Prompt	Cloze-style	Restore the span of the mask sentences
PT* [108]	ACL'21	GPT-2	Latent Prompt	Prefix Prompt	Add prefix ID in front of the input text
CP-Tuning [116]	ACL'22	PLMs	Latent Prompt	Prefix Prompt	Continuous prompt embedding instead of manually designed prompt
OptiPrompt [117]	NAACL'21	BERT	Latent Prompt	Cloze-style	Initialize with text prompt templates and fine-tuning
GPT understands [118]	Arxiv'21	GPTs	Latent Prompt	Prefix Prompt	Model output is regarded as hidden vector to optimize
PPT [119]	ACL'22	PLMS	Latent Prompt	Cloze-style	Add soft prompt during pre training

Contrast Learning is involved in FSL multitask



Instance Formatting and Two Different Methods for Constructing the Instruction-formatted Instances



(a) Instance format

(b) Formatting existing datasets

(c) Formatting human needs

In-context Learning (ICL) and Chain-of-thought (CoT) Prompting

In-Context Learning

Answer the following mathematical reasoning questions:

Q: If you have 12 candies and you give 4 candies to your friend, how many candies do you have left?

A: The answer is 8.

Q: If a rectangle has a length of 6 cm and a width of 3 cm, what is the perimeter of the rectangle?

A: The answer is 18 cm.

Q: Sam has 12 marbles. He gives 1/4 of them to his sister. How many marbles does Sam have left?

A: The answer is 9.

Chain-of-Thought Prompting

Answer the following mathematical reasoning questions:

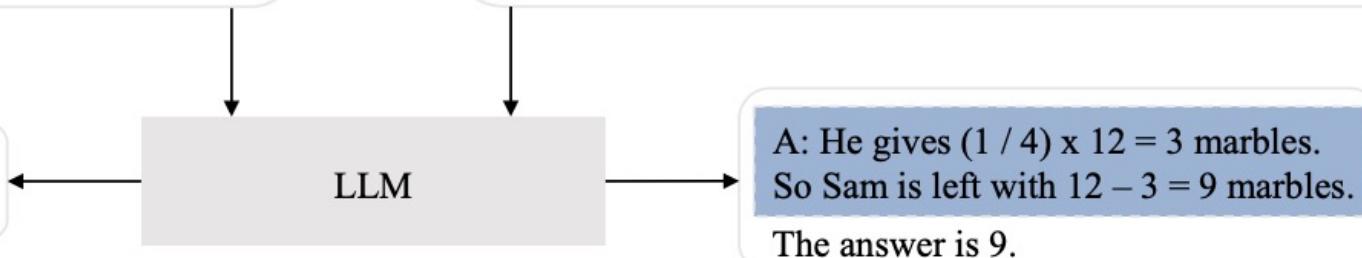
Q: If a rectangle has a length of 6 cm and a width of 3 cm, what is the perimeter of the rectangle?

A: For a rectangle, add up the length and width and double it. So, the perimeter of this rectangle is $(6 + 3) \times 2 = 18$ cm.

The answer is 18 cm.

Q: Sam has 12 marbles. He gives 1/4 of them to his sister. How many marbles does Sam have left?

A: He gives $(1 / 4) \times 12 = 3$ marbles. So Sam is left with $12 - 3 = 9$ marbles.
The answer is 9.



: Task description

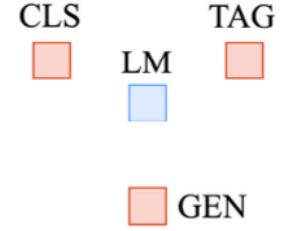
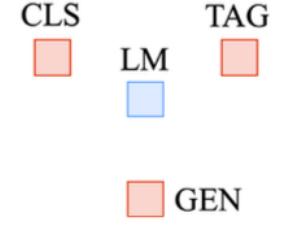
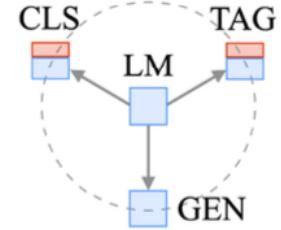
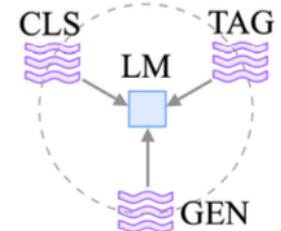
: Demonstration

: Chain-of-Thought

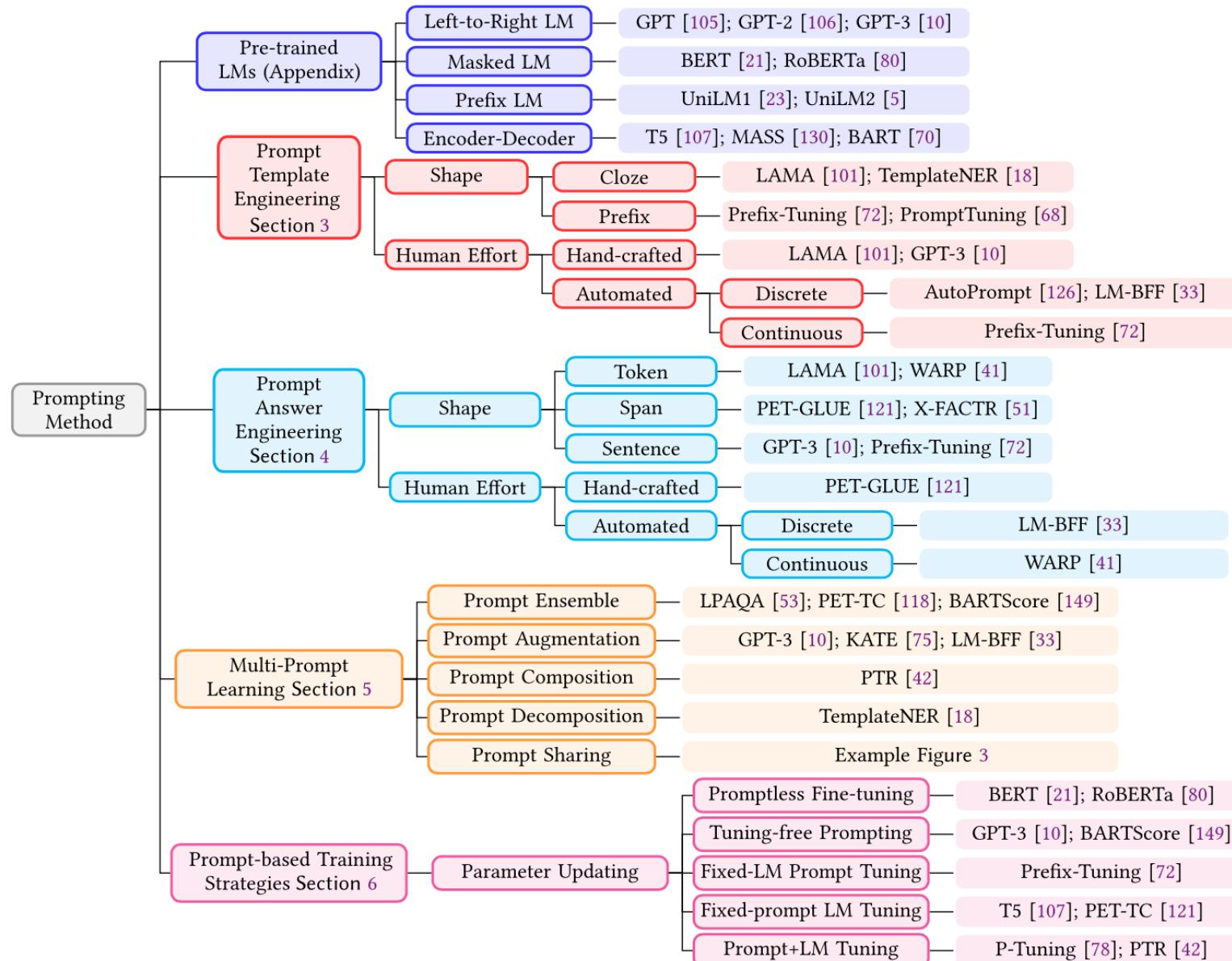
: Query

Pre-train, Prompt, and Predict: Prompting Methods in Natural Language Processing (LLMs)

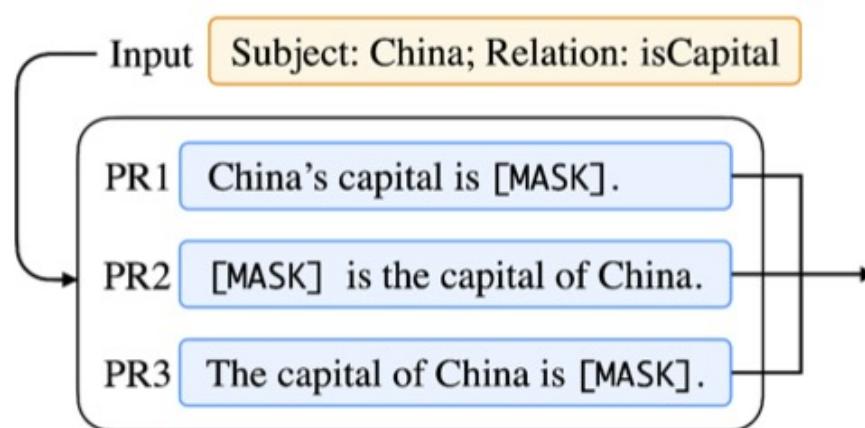
Four Paradigms in NLP (LM)

Paradigm	Engineering	Task Relation
a. Fully Supervised Learning (Non-Neural Network)	Feature (e.g. word identity, part-of-speech, sentence length)	
b. Fully Supervised Learning (Neural Network)	Architecture (e.g. convolutional, recurrent, self-attentional)	
c. Pre-train, Fine-tune	Objective (e.g. masked language modeling, next sentence prediction)	
Transfer Learning: Pre-training, Fine-Tuning (FT)		
d. Pre-train, Prompt, Predict	Prompt (e.g. cloze, prefix)	

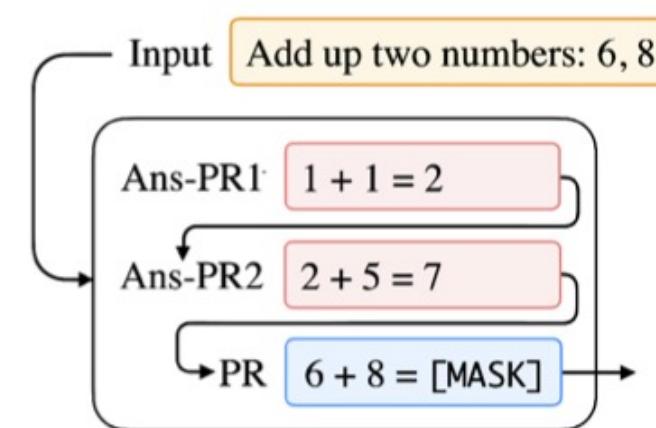
Typology of Prompting Methods



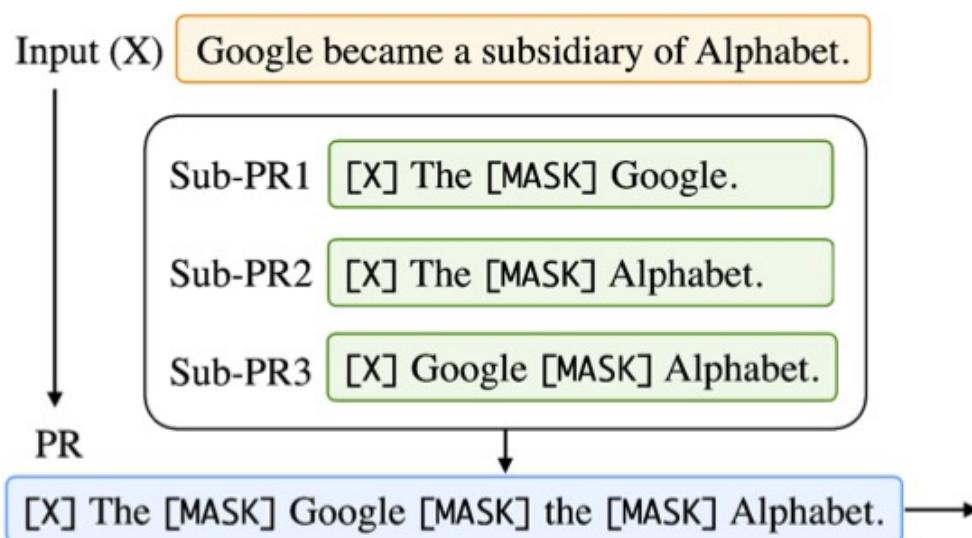
Different Multi-Prompt Learning Strategies



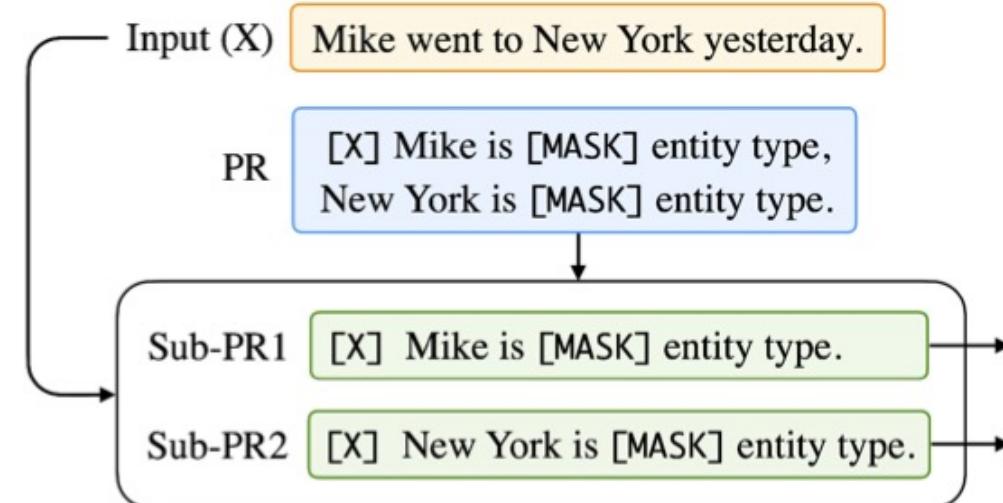
(a) Prompt Ensembling.



(b) Prompt Augmentation.



(c) Prompt Composition.



(d) Prompt Decomposition.

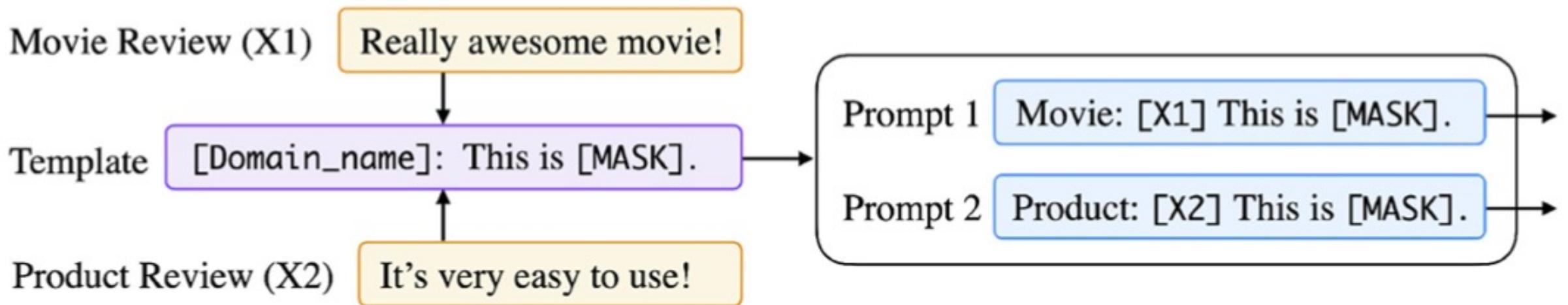
Examples of Input, Template, and Answer for Different Tasks

Type	Task Example	Input ([X])	Template	Answer ([Z])
Text Classification	Sentiment	I love this movie.	[X] The movie is [Z].	great fantastic ...
	Topics	He prompted the LM.	[X] The text is about [Z].	sports science ...
	Intention	What is taxi fare to Denver?	[X] The question is about [Z].	quantity city ...
Text-span Classification	Aspect Sentiment	Poor service but good food.	[X] What about service? [Z].	Bad Terrible ...
				Yes No ...
				...
Text-pair Classification	Natural Language Inference	[X1]: An old man with ...	[X1]? [Z], [X2]	Yes No ...
		[X2]: A man walks
		[X1]: Mike went to Paris.		organization location ...
Tagging	Named Entity Recognition	[X2]: Paris	[X1][X2] is a [Z] entity.	...
		[X1]: Mike went to Paris.		The victim ... A woman
		[X2]: Paris		...
Text Generation	Summarization	Las Vegas police ...	[X] TL;DR: [Z]	The victim ... A woman
				...
				I love you. I fancy you. ...
Regression	Textual Similarity	[X1]: A man is smoking.	[X1] [Z], [X2]	Yes No ...
		[X2]: A man is skating.		...
		[X1]: A man is smoking.		...

Characteristics of Different Tuning Strategies

Strategy	LM Params	Prompt Params		Example
		Additional	Tuned	
Promptless Fine-tuning	Tuned	—	—	ELMo [97], BERT [20], BART [69]
Tuning-free Prompting	Frozen	✗	✗	GPT-3 [9], AutoPrompt [125], LAMA [100]
Fixed-LM Prompt Tuning	Frozen	✓	Tuned	Prefix-Tuning [71], Prompt-Tuning [67]
Fixed-prompt LM Tuning	Tuned	✗	✗	PET-TC [117], PET-Gen [118], LM-BFF [32]
Prompt+LM Fine-tuning	Tuned	✓	Tuned	PADA [5], P-Tuning [77], PTR [41]

Multi-prompt Learning for Multi-task, Multi-domain, or Multi-lingual Learning



GPT Prompt Engineering for Question Answering

- Find the answer to the question from the given context.
- When the question cannot be answered with the given context, say "unanswerable".
- Just say the answer without repeating the question.
- Context: {context}
- Question:{question}
- Answer:

Prompts and QA Inference For FLAN T5

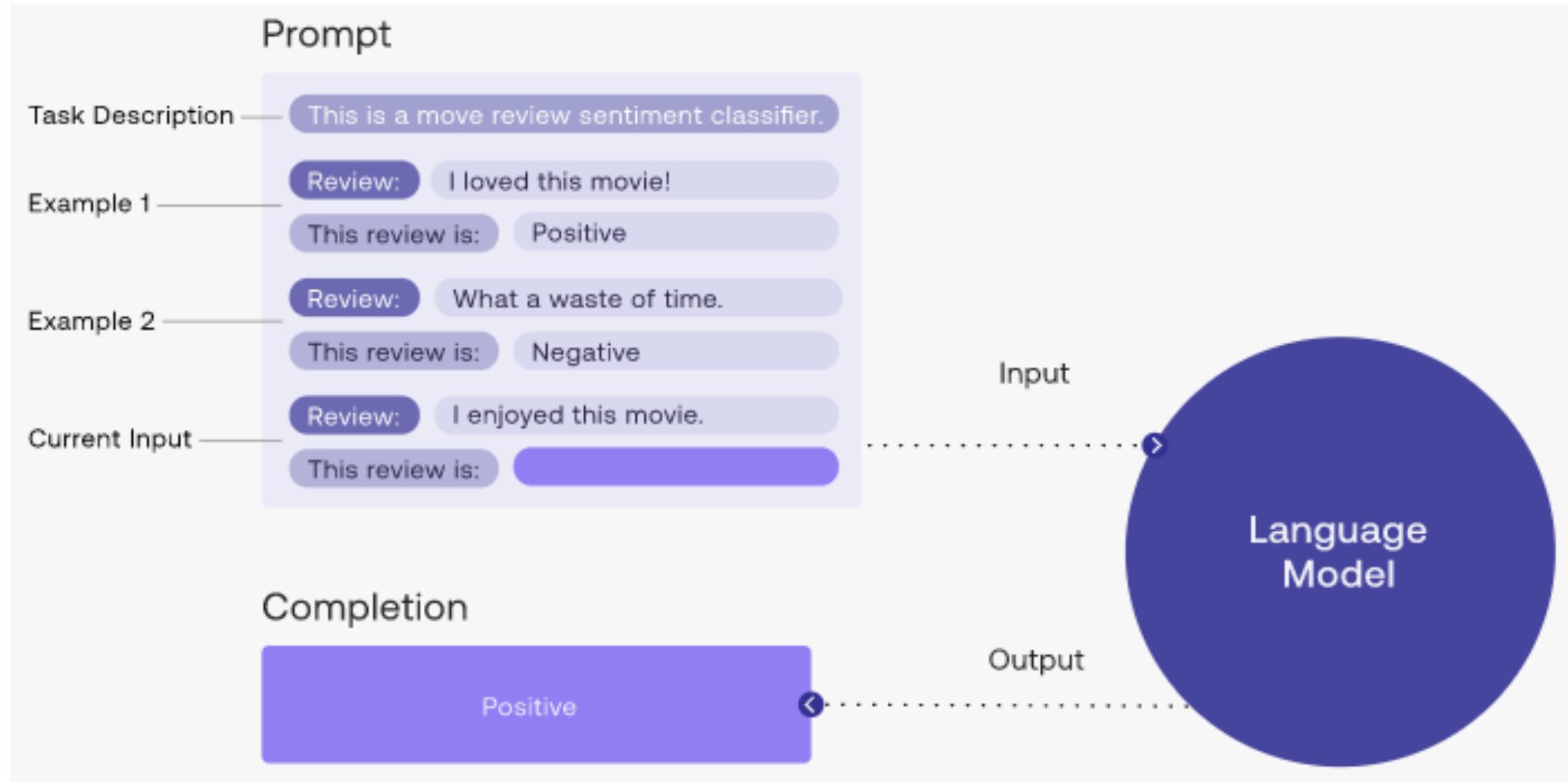
Question Answering

- Prompts and QA Inference For FLAN T5, we follow [41] and use the following prompt:
- Context: {context}\nQuestion: {question}\nAnswer:
- Context: {context}
- Question: {question}
- Answer:

Prompt Engineering

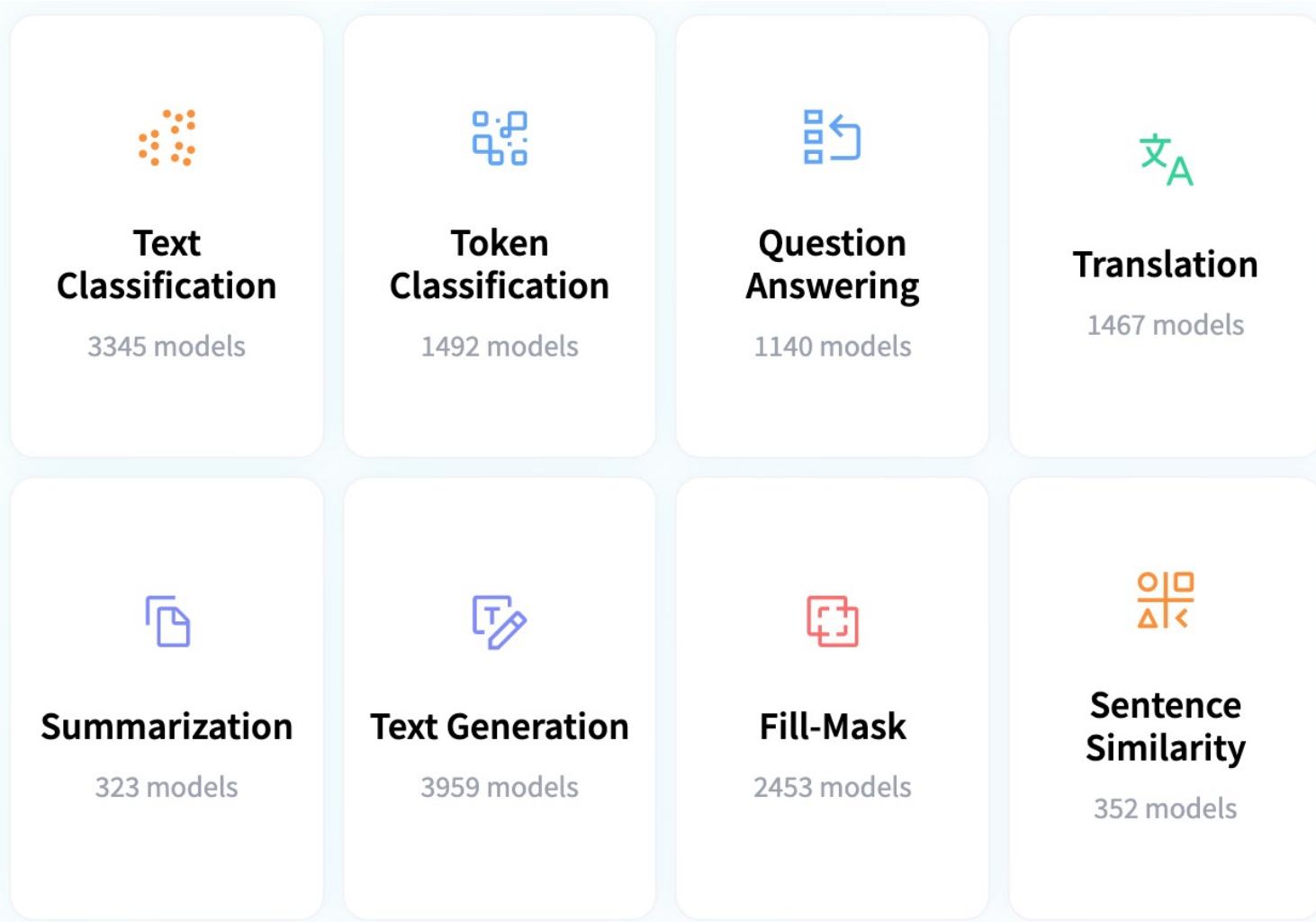
- **Prompts For FLAN models**
 - Longpre, S., Hou, L., Vu, T., Webson, A., Chung, H. W., Tay, Y., ... & Roberts, A. (2023). The flan collection: Designing data and methods for effective instruction tuning. *arXiv preprint arXiv:2301.13688*.
- **MNLI, NLI-FEVER, VitaminC:**
 - "Premise: {premise}\n\nHypothesis: {hypothesis}\n\nDoes the premise entail the hypothesis?\n\nA yes\nB it is not possible to tell\nC no"
- **ANLI:**
 - "{context}\n\nBased on the paragraph above can we conclude that\"{hypothesis}\"?\n\nA Yes\nB It's impossible to say\nC No"
- **SNLI:**
 - "If \"{premise}\", does this mean that \"{hypothesis}\"?\n\nA yes\nB it is not possible to tell\nC no"

Prompt Engineering with ChatGPT for NLP



Hugging Face Tasks

Natural Language Processing



NLP with Transformers Github

Why GitHub? Team Enterprise Explore Marketplace Pricing

Search / Sign in Sign up

[nlp-with-transformers / notebooks](#) Public

Notifications Fork 170 Star 1.1k

Code Issues Pull requests Actions Projects Wiki Security Insights

main 1 branch 0 tags Go to file Code

lewtn Merge pull request #21 from JingchaoZhang/patch-3 ... ae5b7c1 15 days ago 71 commits

.github/ISSUE_TEMPLATE Update issue templates 25 days ago

data Move dataset to data directory 4 months ago

images Add README last month

scripts Update issue templates 25 days ago

.gitignore Initial commit 4 months ago

01_introduction.ipynb Remove Colab badges & fastdoc refs 27 days ago

02_classification.ipynb Merge pull request #8 from nlp-with-transformers/remove-display-df 26 days ago

03_transformer-anatomy.ipynb [Transformers Anatomy] Remove cells with figure references 22 days ago

04_multilingual-ner.ipynb Merge pull request #8 from nlp-with-transformers/remove-display-df 26 days ago

05_text-generation.ipynb Merge pull request #8 from nlp-with-transformers/remove-display-df 26 days ago

About

Jupyter notebooks for the Natural Language Processing with Transformers book

[transformersbook.com/](#)

Readme Apache-2.0 License 1.1k stars 33 watching 170 forks

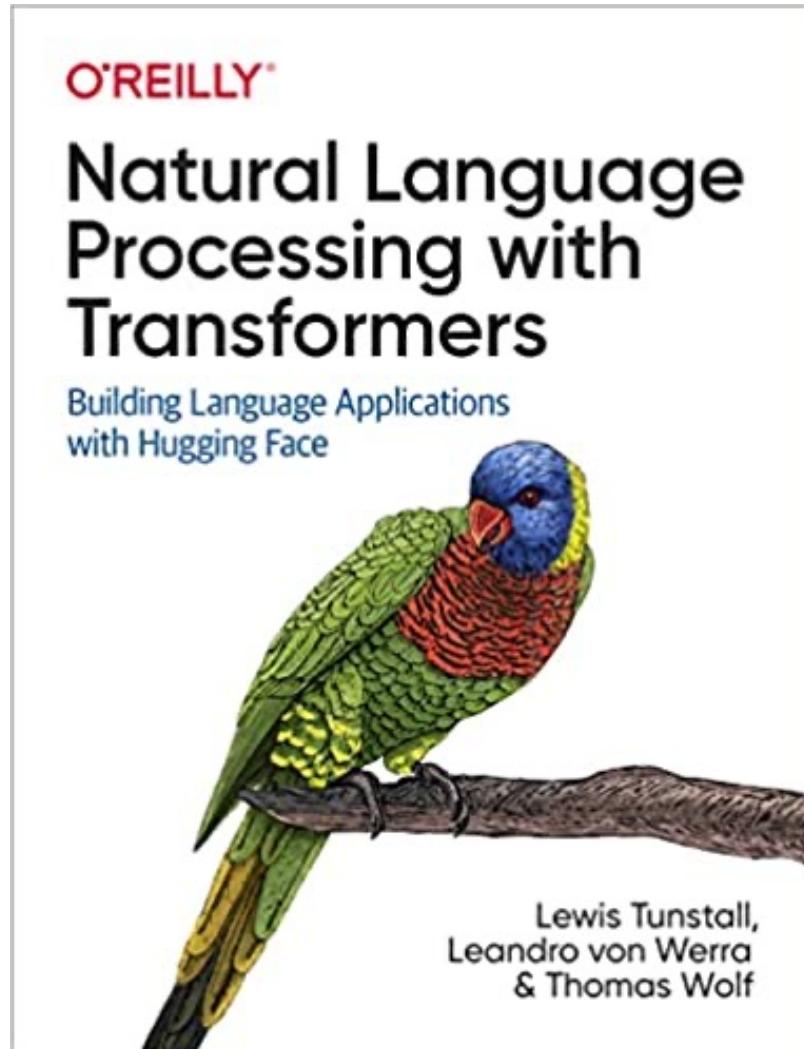
O'REILLY® Natural Language Processing with Transformers Building Language Applications with Hugging Face

Releases No releases published

Packages

<https://github.com/nlp-with-transformers/notebooks>

NLP with Transformers Github Notebooks



Running on a cloud platform

To run these notebooks on a cloud platform, just click on one of the badges in the table below:

Chapter	Colab	Kaggle	Gradient	Studio Lab
Introduction	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Text Classification	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Transformer Anatomy	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Multilingual Named Entity Recognition	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Text Generation	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Summarization	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Question Answering	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Making Transformers Efficient in Production	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Dealing with Few to No Labels	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Training Transformers from Scratch	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Future Directions	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab

Nowadays, the GPUs on Colab tend to be K80s (which have limited memory), so we recommend using [Kaggle](#), [Gradient](#), or [SageMaker Studio Lab](#). These platforms tend to provide more performant GPUs like P100s, all for free!

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>

The screenshot shows a Google Colab notebook interface. The title bar reads "python101.ipynb" and "All changes saved". The main content area has a red header "NLP with Transformers". On the left, there's a sidebar with a "Table of contents" section containing various topics like "Natural Language Processing with Transformers", "Text Classification", etc. The main content area has two sections: "Natural Language Processing with Transformers" and "Text Classification". The "Natural Language Processing with Transformers" section contains code cells for cloning a GitHub repository and running setup scripts. It also includes a text block with a story about Optimus Prime and Megatron. The "Text Classification" section contains code cells for importing the transformers library and creating a classifier pipeline.

NLP with Transformers

- Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
- Github: <https://github.com/nlp-with-transformers/notebooks>

```
[1]: 1 !git clone https://github.com/nlp-with-transformers/notebooks.git
2 %cd notebooks
3 from install import *
4 install_requirements()
```

```
[3]: 1 from utils import *
2 setup_chapter()
```

```
[12]: 1 text = """Dear Amazon, last week I ordered an Optimus Prime action figure \
2 from your online store in Germany. Unfortunately, when I opened the package, \
3 I discovered to my horror that I had been sent an action figure of Megatron \
4 instead! As a lifelong enemy of the Decepticons, I hope you can understand my \
5 dilemma. To resolve the issue, I demand an exchange of Megatron for the \
6 Optimus Prime figure I ordered. Enclosed are copies of my records concerning \
7 this purchase. I expect to hear from you soon. Sincerely, Bumblebee."""
```

Text Classification

```
[13]: 1 from transformers import pipeline
2 classifier = pipeline("text-classification")
```

```
[14]: 1 import pandas as pd
2 outputs = classifier(text)
3 pd.DataFrame(outputs)
```

<https://tinyurl.com/aintpuppython101>

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>

The screenshot shows a Google Colab notebook titled "Text Classification". The left sidebar contains a "Table of contents" with several sections under "Text Classification with Transformers", such as "The Dataset", "From Datasets to DataFrames", "From Text to Tokens", etc. The main area displays code cells and their outputs. Cell [10] shows a command to check GPU usage: !nvidia-smi. Cell [11] contains code to clone a GitHub repository and install requirements: # Uncomment and run this cell if you're on Colab or Kaggle, !git clone https://github.com/nlp-with-transformers/notebooks.git, %cd notebooks, from install import *, install_requirements(). Cell [12] hides imports: # hide, from utils import *, setup_chapter(). Cell [13] lists available datasets: from datasets import list_datasets, all_datasets = list_datasets(), print(f"There are {len(all_datasets)} datasets currently available on the Hub"), print(f"The first 10 are: {all_datasets[:10]}"). The output of this cell shows there are 3783 datasets and the first 10 are: ['acronym_identification', 'ade_corpus_v2', 'adversarial_qa', ...]. The top right corner shows "All changes saved" and various status indicators like RAM and Disk usage.

Text Classification

Table of contents

- Text Classification with Transformers
 - The Dataset
 - From Datasets to DataFrames
 - From Text to Tokens
 - Character Tokenization
 - Word Tokenization
 - Subword Tokenization
 - Tokenizing the Whole Dataset
 - Training a Text Classifier
 - Transformers as Feature Extractors
 - Extracting the last hidden states
 - Creating a feature matrix
 - Visualizing the training set
 - Training a simple classifier
 - Fine-Tuning Transformers
 - Loading a pretrained model
 - Defining the performance metrics
 - Training the model
 - Sidebar: Fine-Tuning with Keras
 - Error analysis
 - Saving and sharing the model

Text Classification with Transformers

- Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
- Github: <https://github.com/nlp-with-transformers/notebooks>

[10] 1 !nvidia-smi

[11] 1 # Uncomment and run this cell if you're on Colab or Kaggle
2 !git clone https://github.com/nlp-with-transformers/notebooks.git
3 %cd notebooks
4 from install import *
5 install_requirements()

[12] 1 # hide
2 from utils import *
3 setup_chapter()

The Dataset

[13] 1 from datasets import list_datasets
2 all_datasets = list_datasets()
3 print(f"There are {len(all_datasets)} datasets currently available on the Hub")
4 print(f"The first 10 are: {all_datasets[:10]}")

There are 3783 datasets currently available on the Hub
The first 10 are: ['acronym_identification', 'ade_corpus_v2', 'adversarial_qa', ...]

<https://tinyurl.com/aintpuppython101>

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>

File Edit View Insert Runtime Tools Help All changes saved

Named Entity Recognition (NER)

Comment Share A

+ Code + Text RAM Disk Editing

Multilingual Named Entity Recognition (NER)

- Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
- Github: <https://github.com/nlp-with-transformers/notebooks>

```
[ ] 1 #NER: https://huggingface.co/tasks/token-classification
2 !pip install transformers
3 from transformers import pipeline
4 classifier = pipeline("ner")
5 classifier("Hello I'm Omar and I live in Zürich.")

▶ 1 from transformers import pipeline
2 classifier = pipeline("ner")
3 classifier("Hello I'm Omar and I live in Zürich.")

[{"end": 14,
 'entity': 'I-PER',
 'index': 5,
 'score': 0.99770516,
 'start': 10,
 'word': 'Omar'},
 {'end': 35,
 'entity': 'I-LOC',
 'index': 10,
 'score': 0.9968976,
 'start': 29,
 'word': 'Zürich'}]
```

<https://tinyurl.com/aintpuppython101>

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>

The screenshot shows a Google Colab notebook interface. The title bar reads "python101.ipynb" and "Text Summarization". The main content area has a heading "Text Summarization" and a list of sources:

- Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
- Github: <https://github.com/nlp-with-transformers/notebooks>

Below the sources, there are two code cells. The first cell contains Python code for summarizing a text about Paris:

```
1 #Source: https://huggingface.co/tasks/summarization
2 !pip install transformers
3 from transformers import pipeline
4 classifier = pipeline("summarization")
5 text = "Paris is the capital and most populous city of France, with an estimated population of 2,175,601 residents as of 2018, in an area of more than
6 classifier(text, max_length=30)
```

The output of this cell shows the summarized text:

```
No model was supplied, defaulted to sshleifer/distilbart-cnn-12-6 (https://huggingface.co/sshleifer/distilbart-cnn-12-6)
Your min_length=56 must be inferior than your max_length=30.
[{'summary_text': 'Paris is the capital and most populous city of France, with an estimated population of 2,175,601 residents . The City of Paris'}]
```

The second cell contains Python code for summarizing a longer text about a package exchange:

```
1 #!pip install transformers
2 text = """Dear Amazon, last week I ordered an Optimus Prime action figure \
3 from your online store in Germany. Unfortunately, when I opened the package, \
4 I discovered to my horror that I had been sent an action figure of Megatron \
5 instead! As a lifelong enemy of the Decepticons, I hope you can understand my \
6 dilemma. To resolve the issue, I demand an exchange of Megatron for the \
7 Optimus Prime figure I ordered. Enclosed are copies of my records concerning \
8 this purchase. I expect to hear from you soon. Sincerely, Bumblebee."""
9 from transformers import pipeline
10 summarizer = pipeline("summarization")
11 outputs = summarizer(text, max_length=45, clean_up_tokenization_spaces=True)
12 print(outputs[0]['summary_text'])
```

<https://tinyurl.com/aintpuppython101>

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>

The screenshot shows a Google Colab notebook interface. The title bar reads "Text Generation". The left sidebar contains a tree view with a single node labeled "Text Generation" expanded. The main workspace displays two code cells. The first cell, numbered [9], contains Python code to initialize a text generation pipeline using the Hugging Face Transformers library. The second cell, numbered [18s], contains another snippet of code to generate text based on the input "Once upon a time". Both cells show the output of their execution.

```
[9] 1 #Source: https://huggingface.co/tasks/text-generation
2 #!pip install transformers
3 from transformers import pipeline
4 generator = pipeline('text-generation', model = 'gpt2')
5 generator("Hello, I'm a language model", max_length = 30, num_return_sequences=3)

Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
[{'generated_text': "Hello, I'm a language model.\n\nBut then, one day, I'm not trying to teach the language in my head.\n\n"}, {'generated_text': "Hello, I'm a language model. I'm an implementation for the type system. I'm working with types and programming language constructs. I am a programmer. As you know, languages are not a linear model. The thing that jumps out at"}, {'generated_text': "Hello, I'm a language modeler, not a programmer. As you know, languages are not a linear model. The thing that jumps out at"}]

[18s] 1 from transformers import pipeline
2 generator = pipeline('text-generation', model = 'gpt2')
3 outputs = generator("Once upon a time", max_length = 30, num_return_sequences=3)
4 print(outputs[0]['generated_text'])

Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Once upon a time, every person who ever saw Jesus, knew that He was Christ. And even though he might not have known Him, He was

[1] 1 from transformers import pipeline
```

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>

The screenshot shows a Google Colab notebook interface. The title of the notebook is "python101.ipynb". The main title of the page is "Question Answering and Dialogue Systems". The content is organized into sections: "Question Answering and Dialogue Systems" and "Question Answering". The code cell [3] contains Python code for setting up a question-answerer pipeline and querying it with a question about living in Taipei. The output cell shows the response from the pipeline, which includes the answer "Taipei", its start and end positions in the context, and a confidence score of approximately 0.973. A note at the bottom indicates that no specific model was supplied, defaulting to distilbert-base-cased-distilled-squad.

```
[3]: 1 !pip install transformers
2 from transformers import pipeline
3 qamodel = pipeline("question-answering")
4 question = "Where do I live?"
5 context = "My name is Michael and I live in Taipei."
6 qamodel(question = question, context = context)

1 from transformers import pipeline
2 qamodel = pipeline("question-answering")
3 question = "Where do I live?"
4 context = "My name is Michael and I live in Taipei."
5 qamodel(question = question, context = context)
6 #{'answer': 'Taipei', 'end': 39, 'score': 0.9730741381645203, 'start': 33}

C No model was supplied, defaulted to distilbert-base-cased-distilled-squad (https://huggingface.co/distilbert-base-cased-distilled-squad)
{'answer': 'Taipei', 'end': 39, 'score': 0.9730741381645203, 'start': 33}
```

<https://tinyurl.com/aintpuppython101>

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>

The screenshot shows the Google Colab interface with three code cells. The first cell (line 12) asks about precipitation causes and receives the answer 'gravity'. The second cell (line 13) asks about other forms of precipitation and receives the answer 'graupel'. The third cell (line 1) attempts to import the pipeline again and receives the answer 'within a cloud'.

```
[12]: 1 from transformers import pipeline
2 qamodel = pipeline("question-answering", model ='deepset/roberta-base-squad2')
3 question = "What causes precipitation to fall?"
4 context = """In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity.
5 output = qamodel(question = question, context = context)
6 print(output[ 'answer' ])
```

```
gravity
```

```
[13]: 1 from transformers import pipeline
2 qamodel = pipeline("question-answering", model ='deepset/roberta-base-squad2')
3 question = "What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?"
4 context = """In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity.
5 output = qamodel(question = question, context = context)
6 print(output[ 'answer' ])
```

```
graupel
```

```
1 #from transformers import pipeline
2 #qamodel = pipeline("question-answering", model ='deepset/roberta-base-squad2')
3 question = "Where do water droplets collide with ice crystals to form precipitation?"
4 context = """In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity.
5 output = qamodel(question = question, context = context)
6 print(output[ 'answer' ])
```

```
within a cloud
```

<https://tinyurl.com/aintpuppython101>

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>

The screenshot shows a Google Colab interface with the following details:

- Title:** python101.ipynb
- Toolbar:** File, Edit, View, Insert, Runtime, Tools, Help, All changes saved.
- Table of contents:**
 - Semantic Analysis
 - Named Entity Recognition (NER)
 - NER with CRF
 - NER with CRF RandomizedSearchCV
 - Sentiment Analysis
 - Sentiment Analysis - Unsupervised Lexical
 - Sentiment Analysis - Supervised Machine Learning
 - Sentiment Analysis - Supervised Deep Learning Models
 - Sentiment Analysis - Advanced Deep Learning
 - Deep Learning and Universal Sentence-Embedding Models
 - Universal Sentence Encoder (USE)
 - Universal Sentence Encoder Multilingual (USEM)
 - Section:** Question Answering and Dialogue Systems
 - Question Answering (QA)
 - BERT for Question Answering**
 - Content:**

Source: Apoorv Nandan (2020), BERT (from HuggingFace Transformers) for Text Extraction, https://keras.io/examples/nlp/text_extraction_with_bert/

Description: Fine tune pretrained BERT from HuggingFace Transformers on SQuAD.

Introduction:

This demonstration uses SQuAD (Stanford Question-Answering Dataset). In SQuAD, an input consists of a question, and a paragraph for context. The goal is to find the span of text in the paragraph that answers the question. We evaluate our performance on this data with the "Exact Match" metric, which measures the percentage of predictions that exactly match any one of the ground-truth answers.

We fine-tune a BERT model to perform this task as follows:

 1. Feed the context and the question as inputs to BERT.
 2. Take two vectors S and T with dimensions equal to that of hidden states in BERT.
 3. Compute the probability of each token being the start and end of the answer span. The probability of a token being the start of the answer is given by a dot product between S and the representation of the token in the last layer of BERT, followed by a softmax over all tokens. The probability of a token being the end of the answer is computed similarly with the vector T.
 4. Fine-tune BERT and learn S and T along the way.

References:

 - [BERT](#)
 - [SQuAD](#)

Question Answering and Dialogue Systems

<https://tinyurl.com/aintpuppython101>

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>

The screenshot shows a Google Colab notebook interface. The left sidebar contains a 'Table of contents' with sections like RandomizedSearchCV, Sentiment Analysis, Deep Learning and Universal Sentence-Embedding Models, Question Answering and Dialogue Systems, and Data Visualization. A specific section titled 'BERT for Question Answering' is highlighted. The main workspace displays code execution results, including progress bars for file downloads and a detailed table of model layers and their properties.

Table of contents

- RandomizedSearchCV
- Sentiment Analysis
 - Sentiment Analysis - Unsupervised
 - Lexical
 - Sentiment Analysis - Supervised
 - Machine Learning
 - Sentiment Analysis - Supervised
 - Deep Learning Models
 - Sentiment Analysis - Advanced Deep
 - Learning
- Deep Learning and Universal Sentence-Embedding Models
 - Universal Sentence Encoder (USE)
 - Universal Sentence Encoder Multilingual (USEM)
- Question Answering and Dialogue Systems
 - Question Answering (QA)
 - BERT for Question Answering**
 - Dialogue Systems
 - Joint Intent Classification and Slot Filling with Transformers
- Data Visualization

+ Code + Text

Comment Share A

RAM Disk ✓ Editing ^

Downloading: 100% 433/433 [00:29<00:00, 14.5B/s]

Downloading: 100% 536M/536M [00:29<00:00, 18.3MB/s]

Model: "model"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[None, 384]	0	
input_3 (InputLayer)	[None, 384]	0	
input_2 (InputLayer)	[None, 384]	0	
tf_bert_model (TFBertModel)	((None, 384, 768), (109482240	input_1[0][0]	
start_logit (Dense)	(None, 384, 1)	768	tf_bert_model[0][0]
end_logit (Dense)	(None, 384, 1)	768	tf_bert_model[0][0]
flatten (Flatten)	(None, 384)	0	start_logit[0][0]
flatten_1 (Flatten)	(None, 384)	0	end_logit[0][0]
activation_7 (Activation)	(None, 384)	0	flatten[0][0]
activation_8 (Activation)	(None, 384)	0	flatten_1[0][0]

Total params: 109,483,776
Trainable params: 109,483,776
Non-trainable params: 0

CPU times: user 20.8 s, sys: 7.75 s, total: 28.5 s
Wall time: 1min 42s

<https://tinyurl.com/aintpuppython101>

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>

The screenshot shows a Google Colab notebook interface. The title bar indicates the file is named "python101.ipynb". The menu bar includes File, Edit, View, Insert, Runtime, Tools, Help, and a status message "All changes saved". The top right features a Comment button, a Share button, a settings gear, and a user profile icon.

The left sidebar contains a "Table of contents" panel with the following structure:

- RandomizedSearchCV
- Sentiment Analysis
 - Sentiment Analysis - Unsupervised
 - Lexical
 - Sentiment Analysis - Supervised
 - Machine Learning
 - Sentiment Analysis - Supervised
 - Deep Learning Models
 - Sentiment Analysis - Advanced Deep
 - Learning
- Deep Learning and Universal Sentence-Embedding Models
 - Universal Sentence Encoder (USE)
 - Universal Sentence Encoder Multilingual (USEM)
- Question Answering and Dialogue Systems
 - Question Answering (QA)
 - BERT for Question Answering
 - Dialogue Systems
 - Joint Intent Classification and Slot Filling with Transformers** (highlighted with a yellow bar)
- Data Visualization

The main content area has a header "Dialogue Systems" in red. Below it, a code cell shows two lines of Python code:

```
[ ] 1 #Source: Olivier Grisel (2020), Transformers (BERT fine-tuning): Joint Intent Classification and S  
2 #https://github.com/m2dsupdlclass/lectures-labs/blob/master/labs/06\_deep\_nlp/Transformers\_Joint\_I
```

A collapsible section titled "Joint Intent Classification and Slot Filling with Transformers" is expanded, containing the following text:

The goal of this notebook is to fine-tune a pretrained transformer-based neural network model to convert a user query expressed in English into a representation that is structured enough to be processed by an automated service.

Here is an example of interpretation computed by such a Natural Language Understanding system:

```
>>> nlu("Book a table for two at Le Ritz for Friday night",  
        tokenizer, joint_model, intent_names, slot_names)
```

```
{  
    'intent': 'BookRestaurant',  
    'slots': {  
        'party_size_number': 'two',  
        'restaurant_name': 'Le Ritz',  
        'timeRange': 'Friday night'  
    }  
}
```

Intent classification is a simple sequence classification problem. The trick is to treat the structured knowledge extraction part ("Slot Filling") as token-level classification problem using BIO-annotations:

<https://tinyurl.com/aintpuppython101>

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>

The screenshot shows the Google Colab interface with a notebook titled "python101.ipynb". The left sidebar contains a "Table of contents" with various sections related to NLP and ML, including "RandomizedSearchCV", "Sentiment Analysis", "Sentiment Analysis - Unsupervised Lexical", "Sentiment Analysis - Supervised Machine Learning", "Sentiment Analysis - Supervised Deep Learning Models", "Sentiment Analysis - Advanced Deep Learning", "Deep Learning and Universal Sentence-Embedding Models", "Universal Sentence Encoder (USE)", "Universal Sentence Encoder Multilingual (USEM)", "Question Answering and Dialogue Systems", "Question Answering (QA)", "BERT for Question Answering", "Dialogue Systems", and "Joint Intent Classification and Slot Filling with Transformers". The main workspace shows a code cell with the following Python code:

```
1 def show_predictions(text, tokenizer, model, intent_names, slot_names):
2     inputs = tf.constant(tokenizer.encode(text))[None, :]
3     outputs = model(inputs)
4     slot_logits, intent_logits = outputs
5     slot_ids = slot_logits.numpy().argmax(axis=-1)[0, 1:-1]
6     intent_id = intent_logits.numpy().argmax(axis=-1)[0]
7     print("Text:", text)
8     print("Intent:", intent_names[intent_id])
9     print("Slots:")
10    for token, slot_id in zip(tokenizer.tokenize(text), slot_ids):
11        print(f"{token}>10} : {slot_names[slot_id]}")
12
13 show_predictions("Book a table for two at Le Ritz for Friday night!",
14                   tokenizer, joint_model, intent_names, slot_names)
```

The output of the code cell is displayed below, showing the processed text and its corresponding slots and intents:

```
Text: Book a table for two at Le Ritz for Friday night!
Intent: BookRestaurant
Slots:
    Book : O
        a : O
    table : O
        for : O
    two : B-party_size_number
        at : O
    Le : B-restaurant_name
        R : I-restaurant_name
    #itz : I-restaurant_name
        for : O
    Friday : B-timeRange
        night : O
        ! : O
```

<https://tinyurl.com/aintpuppython101>

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>

The screenshot shows the Google Colab interface with a notebook titled "python101.ipynb". The left sidebar contains a "Table of contents" with several sections: NER with CRF, Sentiment Analysis (with sub-sections for Unsupervised, Supervised Machine Learning, Supervised Deep Learning Models, and Advanced Deep Learning), Deep Learning and Universal Sentence-Embedding Models (with sub-sections for Universal Sentence Encoder (USE) and Multilingual (USEM)), Question Answering and Dialogue Systems (with sub-sections for Question Answering (QA) and BERT for Question Answering), Dialogue Systems, and Joint Intent Classification and Slot Filling with Transformers. The main workspace displays a block of Python code. The code defines a function `nlu` that takes text, a tokenizer, a model, intent names, and slot names. It encodes the text, runs it through the model, and decodes the results into predictions. A specific call to `nlu` is shown for booking a table at Le Ritz for Friday night.

```
# NAIVE BIO: handling: treat B- and I- the same...
new_slot_name = current_word_slot_name[2:]
if active_slot_name is None:
    active_slot_words.append(word)
    active_slot_name = new_slot_name
elif new_slot_name == active_slot_name:
    active_slot_words.append(word)
else:
    collected_slots[active_slot_name] = " ".join(active_slot_words)
    active_slot_words = [word]
    active_slot_name = new_slot_name
if active_slot_name:
    collected_slots[active_slot_name] = " ".join(active_slot_words)
info["slots"] = collected_slots
return info

def nlu(text, tokenizer, model, intent_names, slot_names):
    inputs = tf.constant(tokenizer.encode(text))[None, :] # batch_size = 1
    outputs = model(inputs)
    slot_logits, intent_logits = outputs
    slot_ids = slot_logits.numpy().argmax(axis=-1)[0, 1:-1]
    intent_id = intent_logits.numpy().argmax(axis=-1)[0]

    return decode_predictions(text, tokenizer, intent_names, slot_names,
                             intent_id, slot_ids)

nlu("Book a table for two at Le Ritz for Friday night",
    tokenizer, joint_model, intent_names, slot_names)

{'intent': 'BookRestaurant',
 'slots': {'party_size_number': 'two',
           'restaurant_name': 'Le Ritz',
           'timeRange': 'Friday night'}}
```

<https://tinyurl.com/aintpuppython101>

NLP with Transformers

```
!git clone https://github.com/nlp-with-transformers/notebooks.git  
%cd notebooks  
from install import *  
install_requirements()
```

```
from utils import *  
setup_chapter()
```

Text Classification

```
text = """Dear Amazon, last week I ordered an Optimus Prime action figure \
from your online store in Germany. Unfortunately, when I opened the package, \
I discovered to my horror that I had been sent an action figure of Megatron \
instead! As a lifelong enemy of the Decepticons, I hope you can understand my \
dilemma. To resolve the issue, I demand an exchange of Megatron for the \
Optimus Prime figure I ordered. Enclosed are copies of my records concerning \
this purchase. I expect to hear from you soon. Sincerely, Bumblebee."""
```

Text Classification

```
text = """Dear Amazon, last week I ordered an Optimus Prime action figure \
from your online store in Germany. Unfortunately, when I opened the package, \
I discovered to my horror that I had been sent an action figure of Megatron \
instead! As a lifelong enemy of the Decepticons, I hope you can understand my \
dilemma. To resolve the issue, I demand an exchange of Megatron for the \
Optimus Prime figure I ordered. Enclosed are copies of my records concerning \
this purchase. I expect to hear from you soon. Sincerely, Bumblebee."""
```

```
from transformers import pipeline
classifier = pipeline("text-classification")

import pandas as pd
outputs = classifier(text)
pd.DataFrame(outputs)
```

	label	score
0	NEGATIVE	0.901546

Text Classification

```
from transformers import pipeline
classifier = pipeline("text-classification")

import pandas as pd
outputs = classifier(text)
pd.DataFrame(outputs)
```

	label	score
0	NEGATIVE	0.901546

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.

<https://github.com/nlp-with-transformers/notebooks>

Named Entity Recognition

```
ner_tagger = pipeline("ner", aggregation_strategy="simple")
outputs = ner_tagger(text)
pd.DataFrame(outputs)
```

	entity_group	score	word	start	end
0	ORG	0.879010	Amazon	5	11
1	MISC	0.990859	Optimus Prime	36	49
2	LOC	0.999755	Germany	90	97
3	MISC	0.556570	Mega	208	212
4	PER	0.590256	##tron	212	216
5	ORG	0.669692	Decept	253	259
6	MISC	0.498349	##icons	259	264
7	MISC	0.775362	Megatron	350	358
8	MISC	0.987854	Optimus Prime	367	380
9	PER	0.812096	Bumblebee	502	511

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.

<https://github.com/nlp-with-transformers/notebooks>

Question Answering

```
reader = pipeline("question-answering")
question = "What does the customer want?"
outputs = reader(question=question, context=text)
pd.DataFrame([outputs])
```

	score	start	end	answer
0	0.631292	335	358	an exchange of Megatron

Summarization

```
summarizer = pipeline("summarization")
outputs = summarizer(text, max_length=45, clean_up_tokenization_spaces=True)
print(outputs[0]['summary_text'])
```

Bumblebee ordered an Optimus Prime action figure from your online store in Germany. Unfortunately, when I opened the package, I discovered to my horror that I had been sent an action figure of Megatron instead.

Text Summarization

```
text = """Dear Amazon, last week I ordered an Optimus Prime action figure \
from your online store in Germany. Unfortunately, when I opened the package, \
I discovered to my horror that I had been sent an action figure of Megatron \
instead! As a lifelong enemy of the Decepticons, I hope you can understand my \
dilemma. To resolve the issue, I demand an exchange of Megatron for the \
Optimus Prime figure I ordered. Enclosed are copies of my records concerning \
this purchase. I expect to hear from you soon. Sincerely, Bumblebee."""
```

```
from transformers import pipeline
summarizer = pipeline("summarization")
outputs = summarizer(text, max_length=45, clean_up_tokenization_spaces=True)
print(outputs[0]['summary_text'])
```

Bumblebee ordered an Optimus Prime action figure from your online store in Germany. Unfortunately, when I opened the package, I discovered to my horror that I had been sent an action figure of Megatron instead.

Translation

```
translator = pipeline("translation_en_to_de",
                     model="Helsinki-NLP/opus-mt-en-de")
outputs = translator(text, clean_up_tokenization_spaces=True, min_length=100)
print(outputs[0]['translation_text'])
```

Sehr geehrter Amazon, letzte Woche habe ich eine Optimus Prime Action Figur aus Ihrem Online-Shop in Deutschland bestellt. Leider, als ich das Paket öffnete, entdeckte ich zu meinem Entsetzen, dass ich stattdessen eine Action Figur von Megatron geschickt worden war! Als lebenslanger Feind der Decepticons, Ich hoffe, Sie können mein Dilemma verstehen. Um das Problem zu lösen, Ich fordere einen Austausch von Megatron für die Optimus Prime Figur habe ich bestellt. Anbei sind Kopien meiner Aufzeichnungen über diesen Kauf. Ich erwarte, bald von Ihnen zu hören. Aufrichtig, Bumblebee.

Text Generation

```
from transformers import set_seed
set_seed(42) # Set the seed to get reproducible results

generator = pipeline("text-generation")
response = "Dear Bumblebee, I am sorry to hear that your order was mixed up."
prompt = text + "\n\nCustomer service response:\n" + response
outputs = generator(prompt, max_length=200)
print(outputs[0]['generated_text'])
```

Customer service response:

Dear Bumblebee, I am sorry to hear that your order was mixed up. The order was completely mislabeled, which is very common in our online store, but I can appreciate it because it was my understanding from this site and our customer service of the previous day that your order was not made correct in our mind and that we are in a process of resolving this matter. We can assure you that your order

Text Generation

Dear Amazon, last week I ordered an Optimus Prime action figure from your online store in Germany. Unfortunately, when I opened the package, I discovered to my horror that I had been sent an action figure of Megatron instead! As a lifelong enemy of the Decepticons, I hope you can understand my dilemma. To resolve the issue, I demand an exchange of Megatron for the Optimus Prime figure I ordered. Enclosed are copies of my records concerning this purchase. I expect to hear from you soon. Sincerely, Bumblebee.

Customer service response:

Dear Bumblebee, I am sorry to hear that your order was mixed up. The order was completely mislabeled, which is very common in our online store, but I can appreciate it because it was my understanding from this site and our customer service of the previous day that your order was not made correct in our mind and that we are in a process of resolving this matter. We can assure you that your order

Question Answering

```
!pip install transformers
from transformers import pipeline
qamodel = pipeline("question-answering")
question = "Where do I live?"
context = "My name is Michael and I live in Taipei."
qamodel(question = question, context = context)
```

```
{'answer': 'Taipei', 'end': 39, 'score': 0.9730741381645203, 'start': 33}
```

Question Answering

```
from transformers import pipeline
qamodel = pipeline("question-answering", model ='deepset/roberta-base-squad2')
question = "Where do I live?"
context = "My name is Michael and I live in Taipei."
output = qamodel(question = question, context = context)
print(output['answer'])
```

Taipei

Text Generation with LLM (zephyr-7b-beta)

```
# Install transformers from source - only needed for versions <= v4.34
!pip install git+https://github.com/huggingface/transformers.git
!pip install accelerate
```

Text Generation with LLM (zephyr-7b-beta)

```
import torch
from transformers import pipeline

pipe = pipeline("text-generation",
model="HuggingFaceH4/zephyr-7b-beta",
torch_dtype=torch.bfloat16,
device_map="auto")
```

Text Generation with LLM (zephyr-7b-beta)

```
# We use the tokenizer's chat template to format each message - see
# https://huggingface.co/docs/transformers/main/en/chat_templating
messages = [
    {
        "role": "system",
        "content": "You are a friendly chatbot who always responds in the style of a pirate",
    },
    {"role": "user", "content": "How many helicopters can a human eat in one sitting?"},
]
prompt = pipe.tokenizer.apply_chat_template(messages,
tokenize=False, add_generation_prompt=True)

outputs = pipe(prompt, max_new_tokens=256,
do_sample=True, temperature=0.7, top_k=50, top_p=0.95)
print(outputs[0]["generated_text"])
```

Text Generation with LLM (zephyr-7b-beta)

```
import torch
from transformers import pipeline

pipe = pipeline("text-generation", model="HuggingFaceH4/zephyr-7b-beta",
torch_dtype=torch.bfloat16, device_map="auto")

# We use the tokenizer's chat template to format each message - see
# https://huggingface.co/docs/transformers/main/en/chat_templating
messages = [
    {
        "role": "system",
        "content": "You are a friendly chatbot who always responds in the style of a pirate",
    },
    {"role": "user", "content": "How many helicopters can a human eat in one sitting?"},
]

prompt = pipe.tokenizer.apply_chat_template(messages, tokenize=False,
add_generation_prompt=True)
outputs = pipe(prompt, max_new_tokens=256, do_sample=True, temperature=0.7,
top_k=50, top_p=0.95)
print(outputs[0]["generated_text"])
```

Summary

- Deep Learning
 - Generative AI
 - Pre-train, Prompt, and Predict (Prompting)
 - Transfer Learning
 - Pre-training, Fine-Tuning (FT)
- Few-Shot Learning (FSL)
 - Meta Learning: Learn to Learn
- One-Shot Learning (1SL)
- Zero-Shot Learning (OSL)(ZSL)

References

- Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
- Denis Rothman (2021), Transformers for Natural Language Processing: Build innovative deep neural network architectures for NLP with Python, PyTorch, TensorFlow, BERT, RoBERTa, and more, Packt Publishing.
- Savaş Yıldırım and Meysam Asgari-Chenaghlou (2021), Mastering Transformers: Build state-of-the-art models from scratch with advanced natural language processing techniques, Packt Publishing.
- Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D., & Finn, C. (2023). Direct preference optimization: Your language model is secretly a reward model. arXiv preprint arXiv:2305.18290.
- Tunstall, Lewis, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang et al. "Zephyr: Direct Distillation of LM Alignment." arXiv preprint arXiv:2310.16944 (2023).
- Yihan Cao, Siyu Li, Yixin Liu, Zhiling Yan, Yutong Dai, Philip S. Yu, and Lichao Sun (2023). "A Comprehensive Survey of AI-Generated Content (AIGC): A History of Generative AI from GAN to ChatGPT." arXiv preprint arXiv:2303.04226.
- Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. (2023) "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing." ACM Computing Surveys 55, no. 9 (2023): 1-35.
- Yisheng Song, Ting Wang, Puyu Cai, Subrota K. Mondal, and Jyoti Prakash Sahoo. (2023) "A comprehensive survey of few-shot learning: Evolution, applications, challenges, and opportunities." ACM Computing Surveys (2023).
- Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min et al. (2023) "A Survey of Large Language Models." arXiv preprint arXiv:2303.18223.
- Junliang Wang, Chuqiao Xu, Jie Zhang, and Ray Zhong (2022). "Big data analytics for intelligent manufacturing systems: A review." Journal of Manufacturing Systems 62 (2022): 738-752.
- Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., ... & Lowe, R. (2022). Training language models to follow instructions with human feedback. arXiv preprint arXiv:2203.02155.
- Gozalo-Brizuela, Roberto, and Eduardo C. Garrido-Merchan (2023). "ChatGPT is not all you need. A State of the Art Review of large Generative AI models." arXiv preprint arXiv:2301.04655 (2023).
- Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. (2023) "InstructBLIP: Towards General-purpose Vision-Language Models with Instruction Tuning." arXiv preprint arXiv:2305.06500 (2023).
- Shahab Saquib Sohail, Faiza Farhat, Yassine Himeur, Mohammad Nadeem, Dag Øivind Madsen, Yashbir Singh, Shadi Atalla, and Wathiq Mansoor (2023). "The Future of GPT: A Taxonomy of Existing ChatGPT Research, Current Challenges, and Possible Future Directions." Current Challenges, and Possible Future Directions (April 8, 2023) (2023).
- Longbing Cao (2022). "Decentralized ai: Edge intelligence and smart blockchain, metaverse, web3, and desci." IEEE Intelligent Systems 37, no. 3: 6-19.
- Qinglin Yang, Yetong Zhao, Huawei Huang, Zehui Xiong, Jiawen Kang, and Zibin Zheng (2022). "Fusing blockchain and AI with metaverse: A survey." IEEE Open Journal of the Computer Society 3 : 122-136.
- Russell Belk, Mariam Humayun, and Myriam Brouard (2022). "Money, possessions, and ownership in the Metaverse: NFTs, cryptocurrencies, Web3 and Wild Markets." Journal of Business Research 153: 198-205.
- Thien Huynh-The, Quoc-Viet Pham, Xuan-Qui Pham, Thanh Thi Nguyen, Zhu Han, and Dong-Seong Kim (2022). "Artificial Intelligence for the Metaverse: A Survey." arXiv preprint arXiv:2202.10336.
- Thippa Reddy Gadekallu, Thien Huynh-The, Weizheng Wang, Gokul Yenduri, Pasika Ranaweera, Quoc-Viet Pham, Daniel Benevides da Costa, and Madhusanka Liyanage (2022). "Blockchain for the Metaverse: A Review." arXiv preprint arXiv:2203.09738.
- Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., ... & Lowe, R. (2022). Training language models to follow instructions with human feedback. arXiv preprint arXiv:2203.02155.
- Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv preprint arXiv:1810.04805.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan et al. (2020) "Language models are few-shot learners." Advances in neural information processing systems 33 (2020): 1877-1901
- Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni (2020). "Generalizing from a few examples: A survey on few-shot learning." ACM computing surveys (csur) 53, no. 3 (2020): 1-34.
- Yao Ge, Yuting Guo, Yuan-Chi Yang, Mohammed Ali Al-Garadi, and Abeed Sarker (2022). "Few-shot learning for medical text: A systematic review." arXiv preprint arXiv:2204.14081 (2022).
- Maria Tsimpoukelli, Jacob L. Menick, Serkan Cabi, S. M. Eslami, Oriol Vinyals, and Felix Hill (2021). "Multimodal few-shot learning with frozen language models." Advances in Neural Information Processing Systems 34 (2021): 200-212.
- OpenAI (2023), A Survey of Techniques for Maximizing LLM Performance, <https://www.youtube.com/watch?v=ahnGLM-RC1Y>
- The Super Duper NLP Repo, <https://notebooks.quantumstat.com/>
- NLP with Transformer, <https://github.com/nlp-with-transformers/notebooks>
- Min-Yuh Day (2023), Python 101, <https://tinyurl.com/aintpupython101>