PAWEŁ GOLIK & MATEUSZ JASTRZĘBIOWSKI

Prompt Learning

A new paradigm in NLP

Created at Canva.com

Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig

ACM Computing Surveys | Volume 55 | Issue 9 | September 2023 | Article No.: 195 | pp 1–35

Four paradigms in NLP

- 2011

Fully Supervised Learning
(Non-Neural Network)
Feature engineering (e.g. word identity, part-of-speech, sentence length)

2011

Fully Supervised Learning
(Neural Network)
Architecture engineering
(e.g., convolutional, recurrent, self-attentional)

2017

Pre-train, Fine-tune
Objective engineering
(e.g., masked language modeling, next sentence prediction)

2019

Pre-train, Prompt, Predict
?

Traditional supervised learning system for NLP

We take input ***x,*** and **predict** output ***y*** based on a model P(***y*** | ***x***;θ)

x =“I love this movie.”

y ϵ *Y* = *{++, +, ~, -, --}*

text classification task

x =“Good morning”

y =“*Dzień dobry*”

conditional text generation

1 Fully-supervised learning without neural networks

< 2011

1 Fully-supervised learning without neural networks

Feature extraction: Bag of Words

Labeled dataset

Inputs x	Labels
I love this movie.	++
It is OK.	+
⋮	⋮
This movie is horrible.	--

Labeled dataset
(with extracted features)

I	is	love	OK	this	movie	horrible	Labels
1	0	1	0	1	1	0	++
0	1	0	1	0	1	0	+
				⋮			⋮
0	1	0	0	1	1	1	--

Challenges

1

labeled dataset

2

handcrafting features (feature engineering)

2 Fully-supervised learning with neural networks

2011 - 2017

2 Fully-supervised learning with neural networks

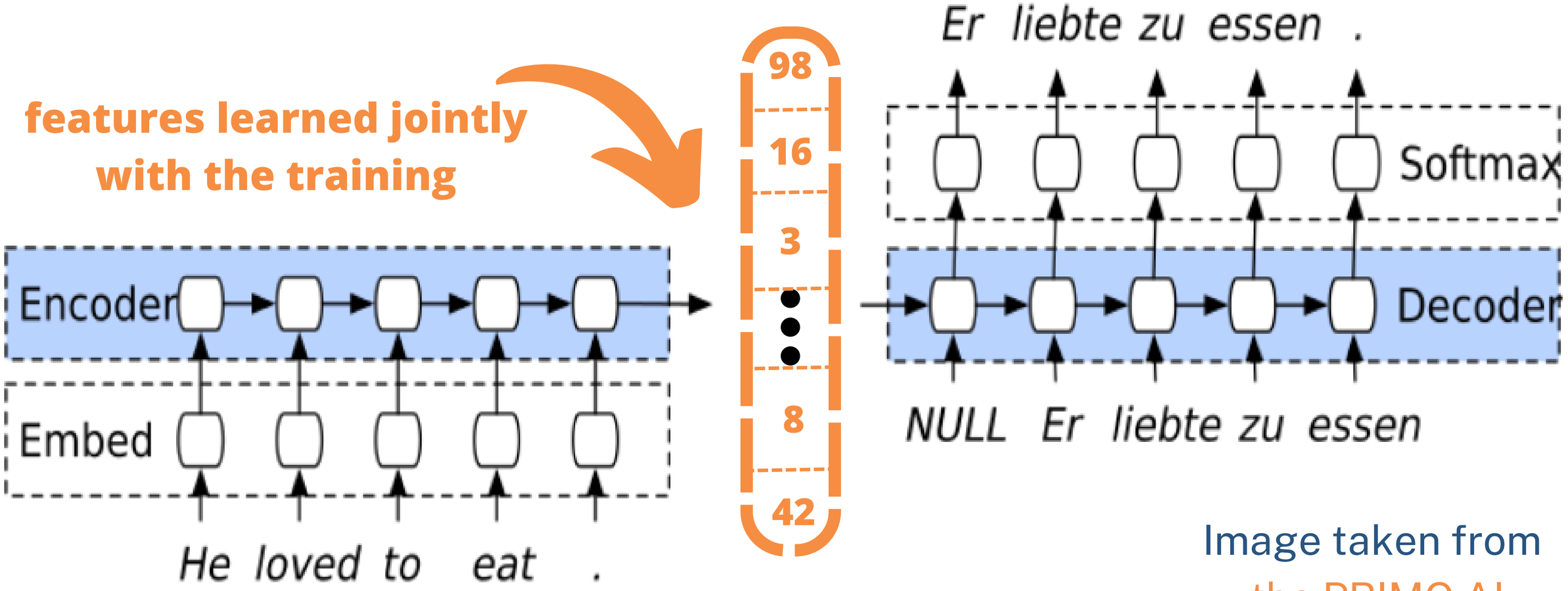


Image taken from the PRIMO.AI

Challenges

1

labeled dataset

2

handcrafting features

3

architecture engineering

3 Pre-train and fine-tune

2017 - 2019

3 Pre-train and fine-tune

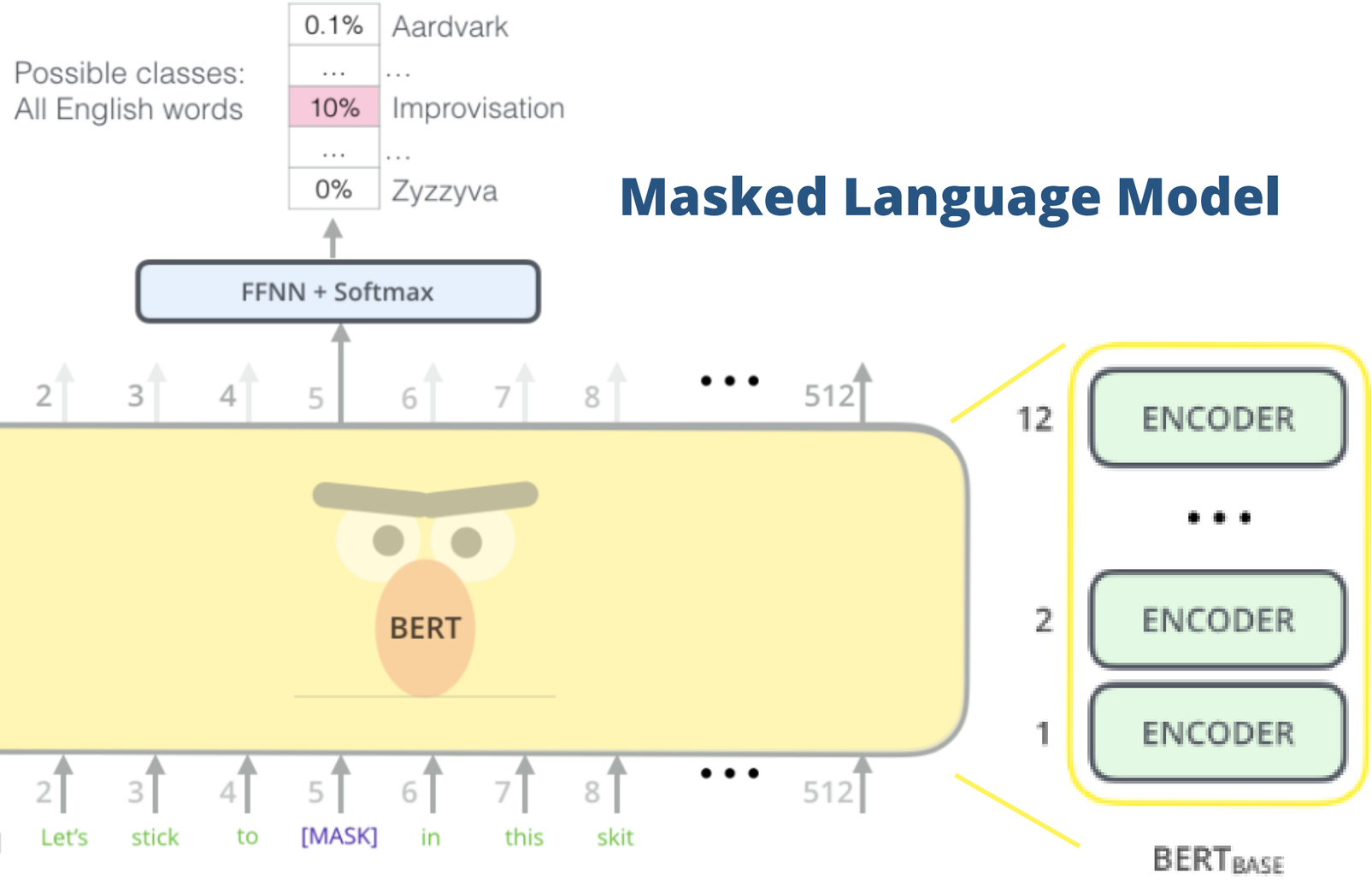
A change in NLP approaches

Supervised models learn how **the answer (y)** is **dependent** on **the input (x)**

A **language model (LM)**, predicting the **probability** of observed textual **input data (x)**

3 Pre-train and fine-tune

Use the output of the masked word's position to predict the masked word

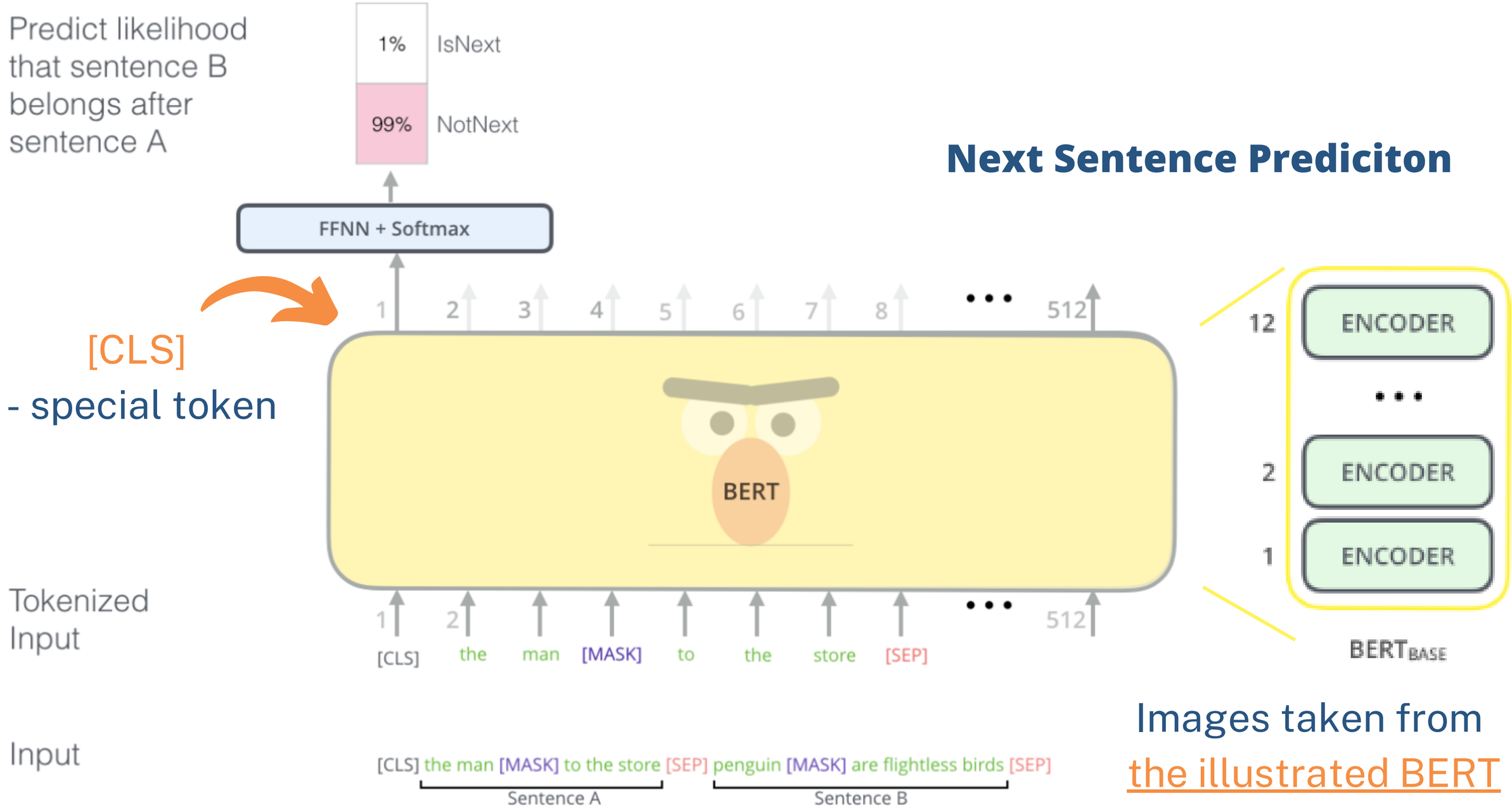


Masked Language Model

An embedding for each token

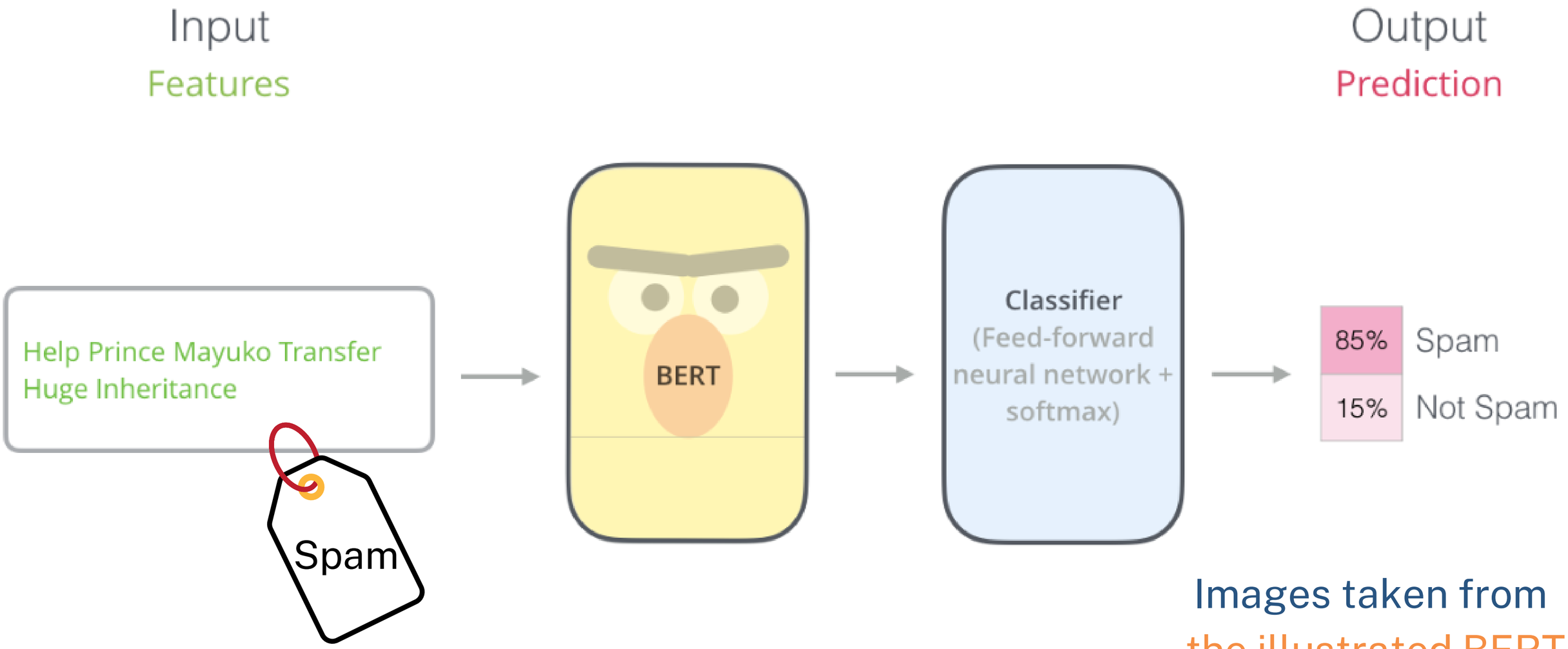
Images taken from the illustrated BERT

3 Pre-train and fine-tune



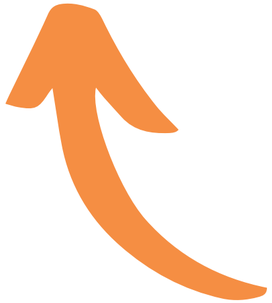
Images taken from [the illustrated BERT]

3 Pre-train and fine-tune



Images taken from
the illustrated BERT

3 Pre-train and fine-tune



Unsupervised!
no labels needed

$P(x;\theta)$

Language Model (LM)
pre-training:

- unsupervised
- abundance of data

Supervised -
labels required

$P(y|x;\theta)$

Language Model (LM)
fine-tuning for a down-stream task:

- supervised
- transfer learning

Cross-encoder

Downstream task:

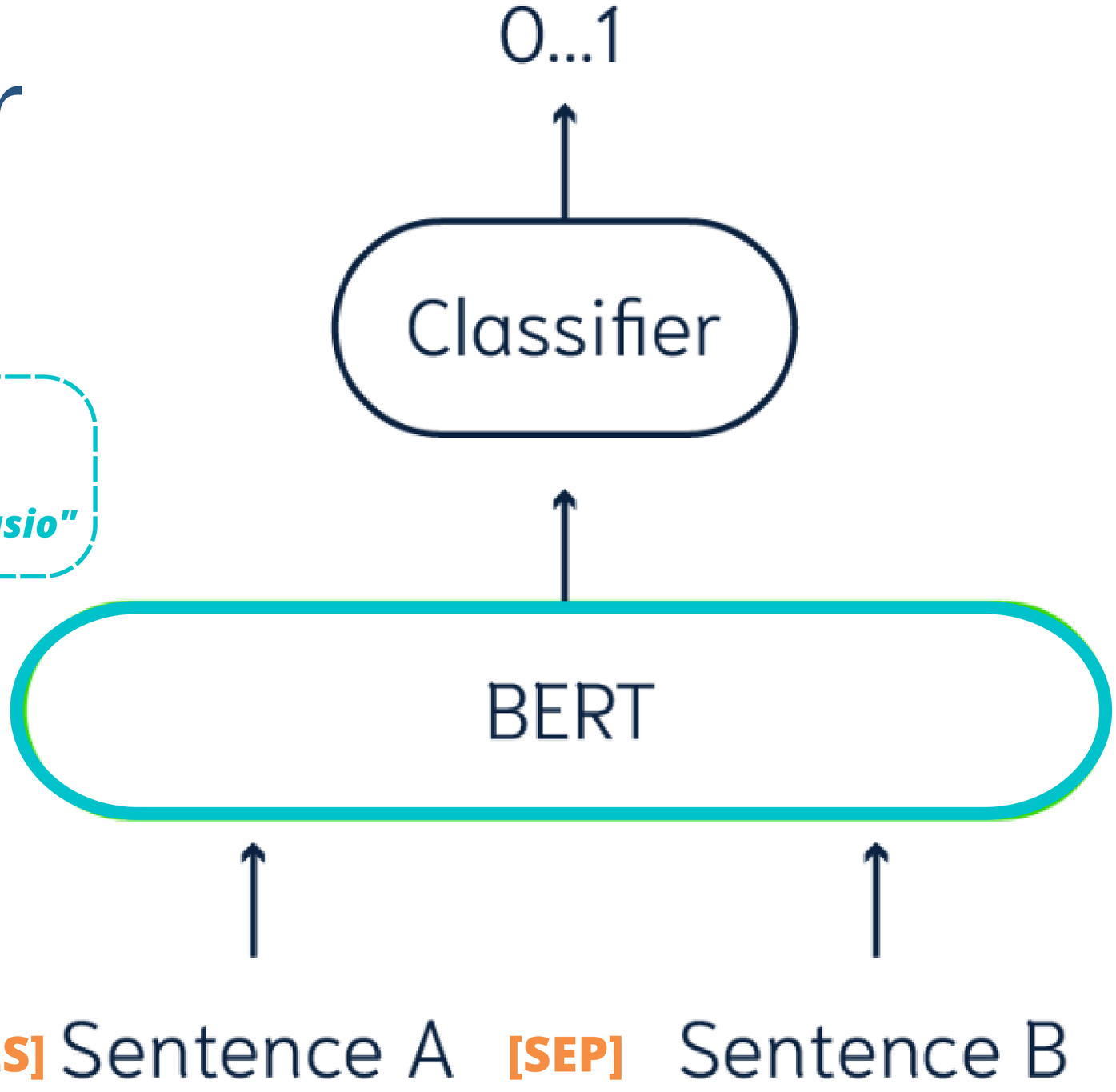
E-commerce product matching

A positive pair of offers:
Sentence A: ***"CASIO LA680WEA-1EF"***
Sentence B: ***"Unisex Watch LA680WEA1EF Casio"***

1

A negative pair of offers:
Sentence A: ***"Rolex Day-Date"***
Sentence B: ***"DateJust Rolex $16,995.00"***

0



Bi-encoder

Downstream task:

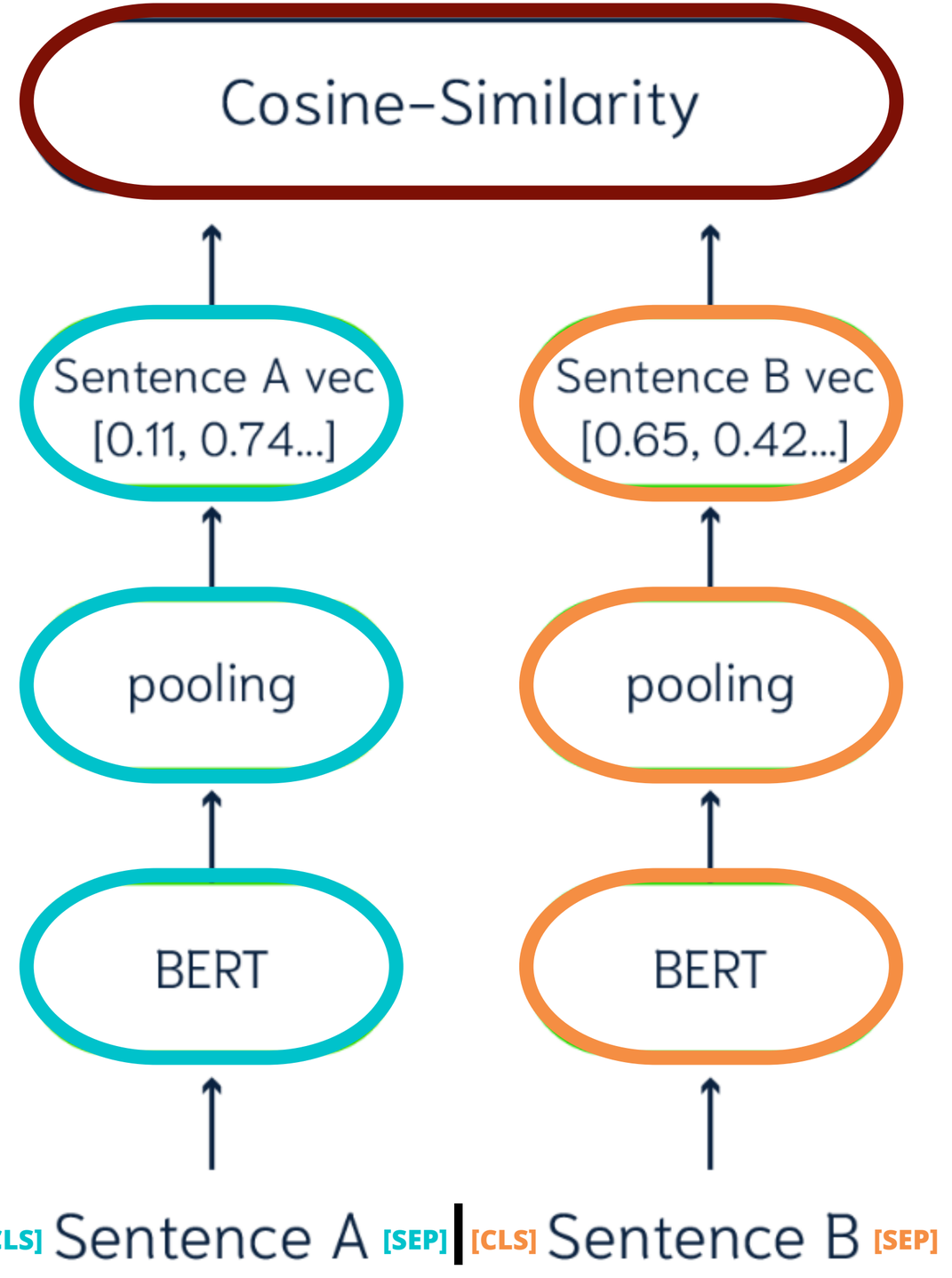
E-commerce product matching

A positive pair of offers:
Sentence A: ***"CASIO LA680WEA-1EF"***
Sentence B: ***"Unisex Watch LA680WEA1EF Casio"***

1

A negative pair of offers:
Sentence A: ***"Rolex Day-Date"***
Sentence B: ***"DateJust Rolex $16,995.00"***

0



Promptless Fine-tuning

Advantages

Well-known supervised learning

Tuning all the Language Model parameters » **fit to larger** datasets

Disadvantages

For large models very **resource-consuming**

A **separate model** for each task

LMs may **overfit** or not learn stably on smaller datasets

Catastrophic forgetting

4 Pre-train, Prompt, Predict

2019 - ?

Prompting

LM predicts **P(x;θ)**

let's get rid of **P(y|x;θ)**!

fine-tuning - **adapting LM**

adapting input of a downstream task

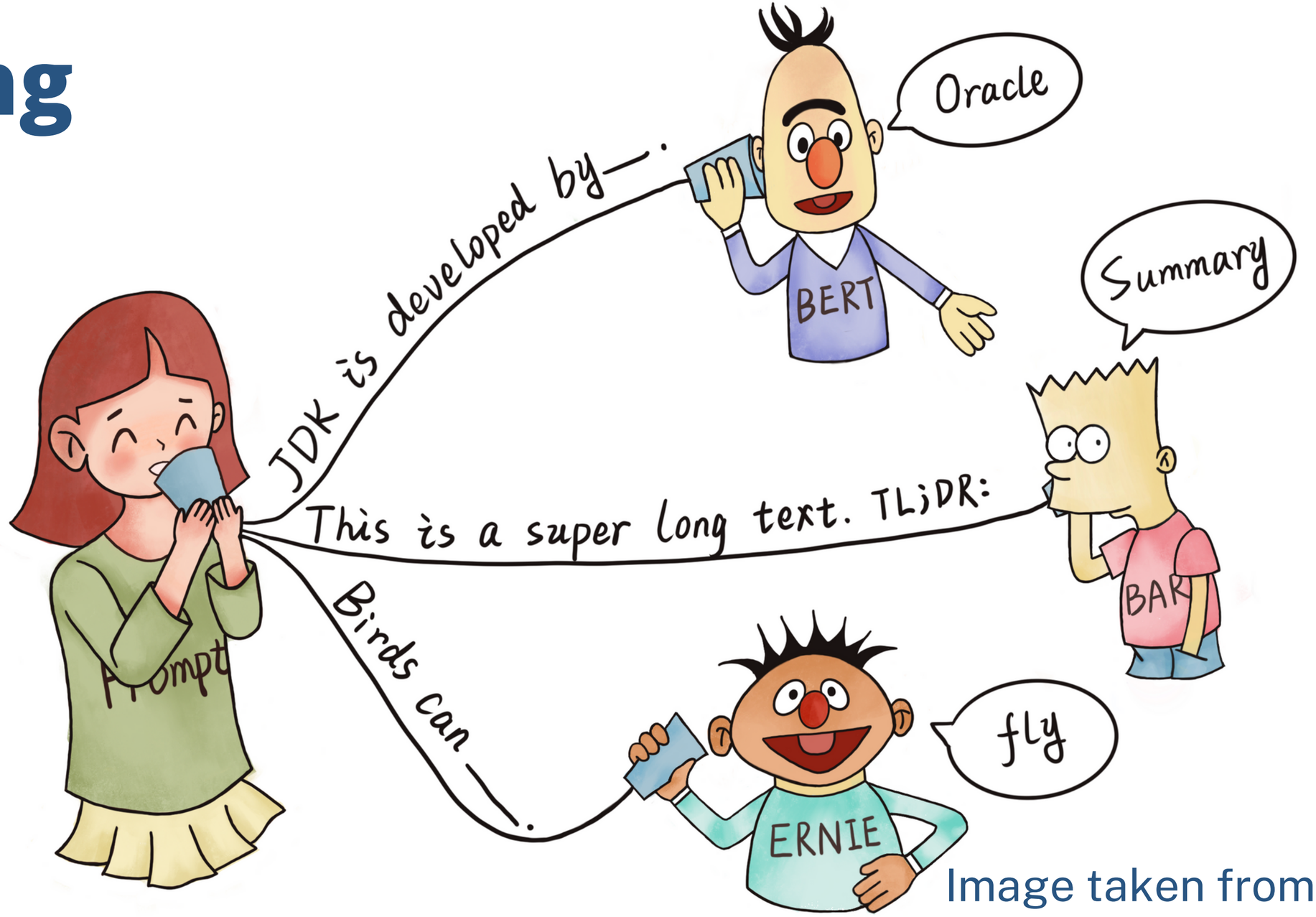


Image taken from the survey

Prompting

target: Y = {++, +, ~, -, --}

We take input ***x*** *(e.g., x =* ***"I love this movie."****)*

Prompting

target: Y = {++, +, ~, -, --}

We take input ***x*** *(e.g., x = **"I love this movie."**)*

We create a ***template x'**: [X] Overall, it was a [Z] movie.*

Prompting

target: Y = {++, +, ~, -, --}

intermediate **answer space:** $\mathcal{Z}$ = {“excellent”, “good”, “OK”, “bad”, “horrible”}

We take input ***x*** *(e.g., x = "I love this movie.")*

We create a ***template x'***: *[X]* ***Overall, it was a*** *[Z]* ***movie.***

filling prompts - f()

Prompting

target: Y = {++, +, ~, -, --}

intermediate **answer space:** $\mathcal{Z}$ = {“excellent”, “good”, “OK”, “bad”, “horrible”}

We take input ***x*** *(e.g., x = **"I love this movie."**)*

We create a ***template x'**: [X] Overall, it was a [Z] movie.*

filling prompts - f()

$$\hat{\boldsymbol{z}} = \underset{\boldsymbol{z} \in \mathcal{Z}}{\text{search}}\, P(f_{\text{fill}}(\boldsymbol{x}', \boldsymbol{z}); \theta).$$

{f(x', z)} **filled prompts**

Prompting

target: Y = {++, +, ~, -, --}

intermediate **answer space:** $\mathcal{Z}$ = {“excellent”, “good”, “OK”, “bad”, “horrible”}

We take input ***x*** *(e.g., x = **"I love this movie."**)*

We create a ***template x'**: [X] Overall, it was a [Z] movie.*

filling prompts - f()

$$\hat{\boldsymbol{z}} = \underset{\boldsymbol{z} \in \mathcal{Z}}{\text{search}}\, P(f_{\text{fill}}(\boldsymbol{x}', \boldsymbol{z}); \theta).$$

{f(x', z)} **filled prompts**

an **argmax** search or **sampling**

answered prompt, e.g., 'excellent' -> ++

Prompting - another examples

- **NER:** Input example: **[X1]**: Mike went to Paris.
  [X2]: Paris

  Template: **[X1][X2]** is a [Z] entity.

  Z = {'organization', 'location', ...}

- **Translation:**

  Input example: **[X1]**: Te quiero.

  Template: Spanish: **[X1]** English: [Z]

  Z ranges from the entirety of the language

Prompting - another examples

- **Text classification:**

Input example: **[X1]**: Jane visited New York...

Z = { 'travels', 'health', 'science' }

Template: Which of these choices best describes the following document? ”travels”, ”health”, ”science”. **[X1]**[Z]

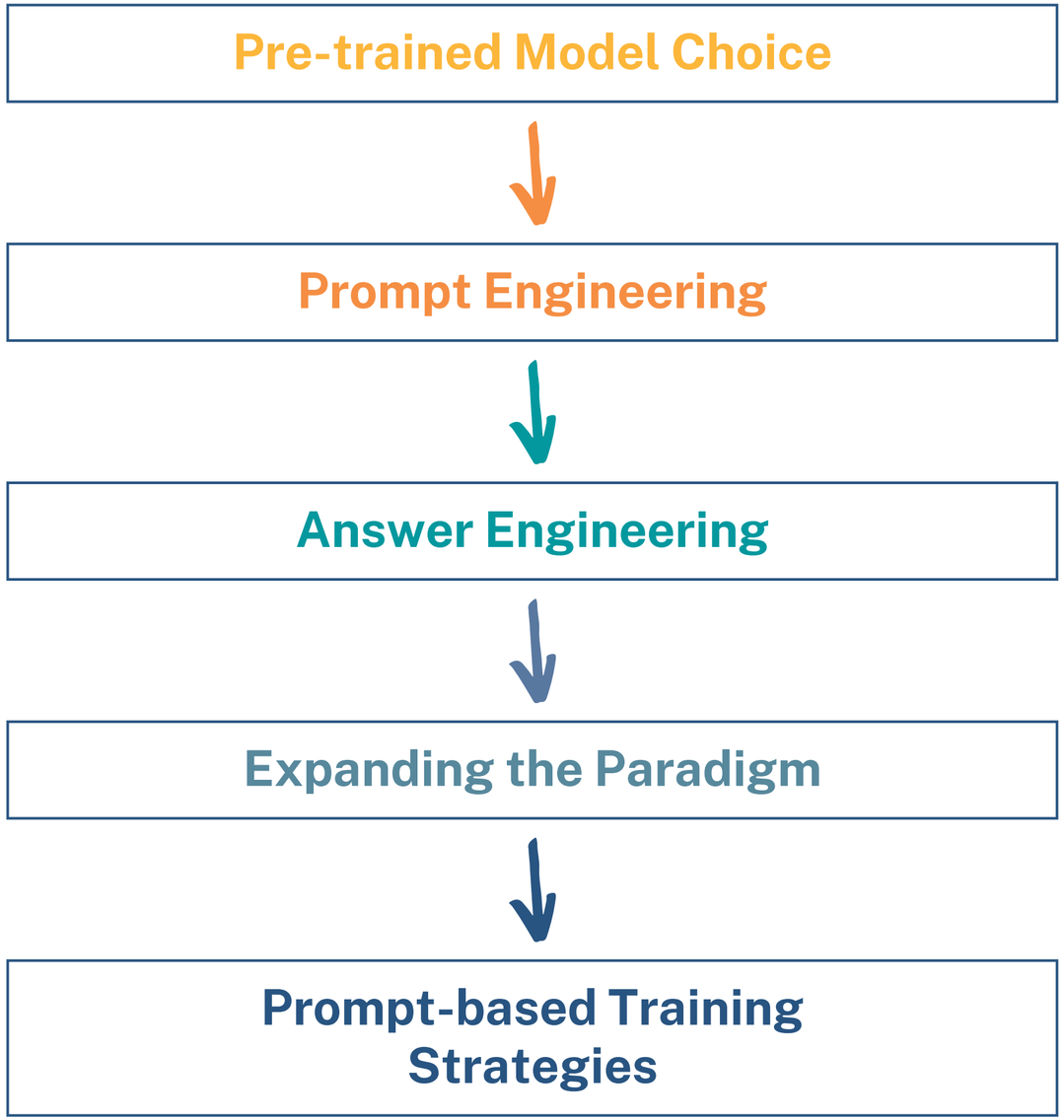
- **Text summarization:**

Input example: **[X1]**: The police found...

Z ranges from the entirety of the language

Template: Text: **[X1]** Summary: [Z]

Design considerations for Prompting



Prompt engineering

is the process of **creating a prompting function (template)** that results in the **most effective performance** on the downstream task.

Prompt shape - cloze prompts

[X1] = Best pizza ever!

[X2] = I want to eat it again!

[Z] = good

Template: **[X1]**. The pizza was so **[Z]**, **[X2]**

Best pizza ever! The pizza was so good. I want to eat it again!

Suitable for: **masked learning models** (MLM)

Prompt shape - prefix prompts

[X] = Best pizza ever! [Z] = Najlepsza pizza!

Template: English: **[X]** Polish: **[Z]**

English: Best pizza ever! Polish: Najlepsza pizza!

Suitable for: tasks regarding **text generation** using auto-regressive LM

Prompts creation

Manual Template Engineering

Most intuitive

Time consuming

Experience dependent

Automated Template Learning

two types: **continous** and **discrete** prompts

---

for though datasets able to find **better templates** than manually created

---

can create custom template for **each input**

Automated discrete prompts

- **Prompt Mining**

scraped from a large text corpus (e.g., Wikipedia)

Example template: [X] [middle words] [Z].

- **Prompt Paraphrasing**

Original template: [X] is created by [Z].

Paraphrased template: [Z] designed [X].

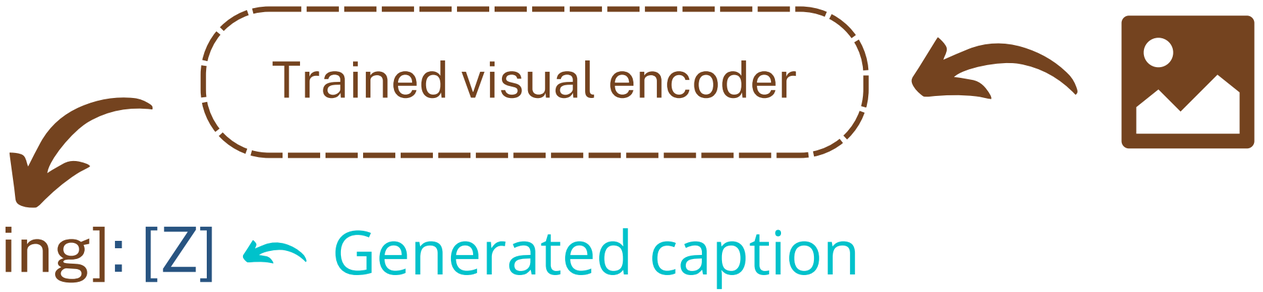
paraphrasing:

- **translating** and then translating back
- a trained **neural prompt rewriter**
- using **synonyms**
- …

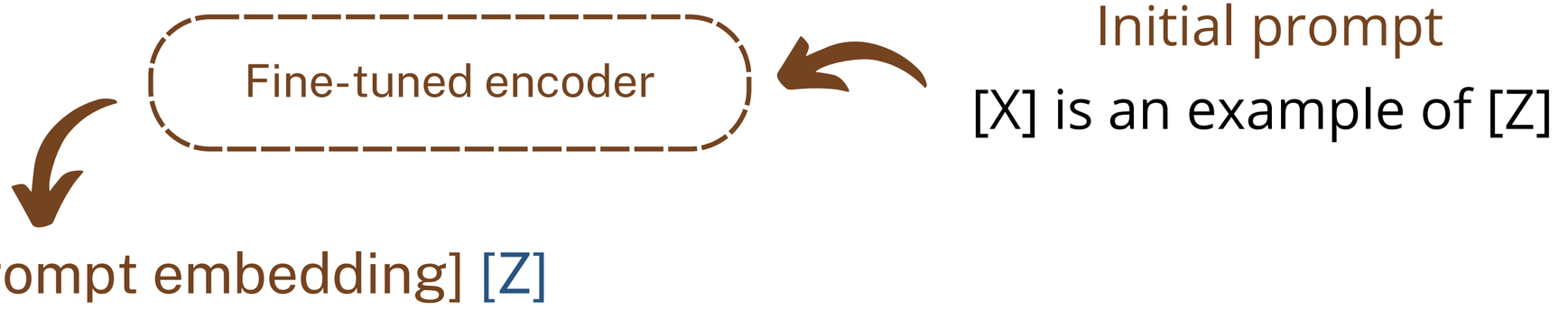
Automated continous prompts

prompting performed **directly** in **the embedding space** of the model

- **Prefix Tuning**



- **Tuning Initialized with Discrete Prompts**



Answer engineering

aims to search for an **answer space Z**
and a **map** to the **original output Y**
that results in an **effective predictive model**

Answer shape

The shape of an answer characterizes its **granularity.**

Tokens

iPhone X is produced by Apple.

Text span

New is the opposite of old, ancient, current.

Sentence/document

French: Je vous aime.
English: I like you.

Answer Space Design Methods

How to design **answer space** $\mathcal{Z}$ and the **mapping** $\mathcal{M}$ from answers to the **requested output** (e.g., class label).

Manual design

Unconstrained space

$\mathcal{Z}$ = {
the space of all tokens
}

$\mathcal{M}$ = Identity(z)

[Z] = "I like..."

"I like..." ↙

Constrained space

$\mathcal{Z}$ = {
"bad",
"good",
"wonderful",
"great"
}

$\mathcal{M}$ = {
0: ["bad"],
1: ["good", "wonderful", "great"]
}

[Z] = "bad" ↷ **0**

Automatic search

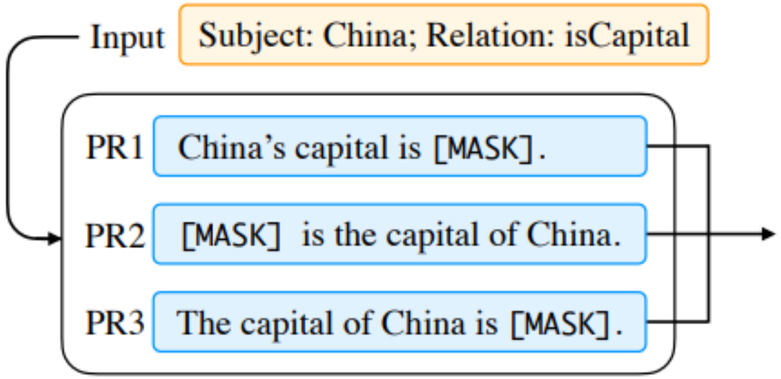
Discrete Answer Search

- Answer paraphrasing
- Prune-then-search
- Label decomposition

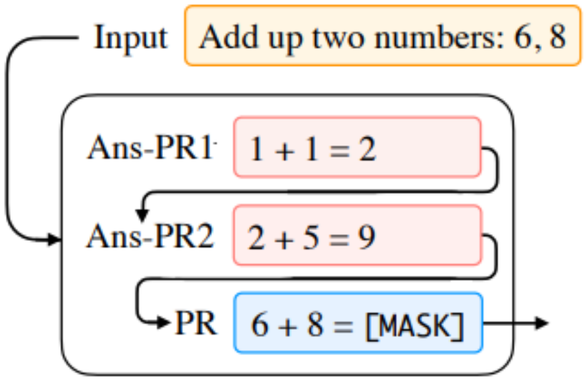
Continuous Answer Search

- Soft answers that can be optimized through gradient descent

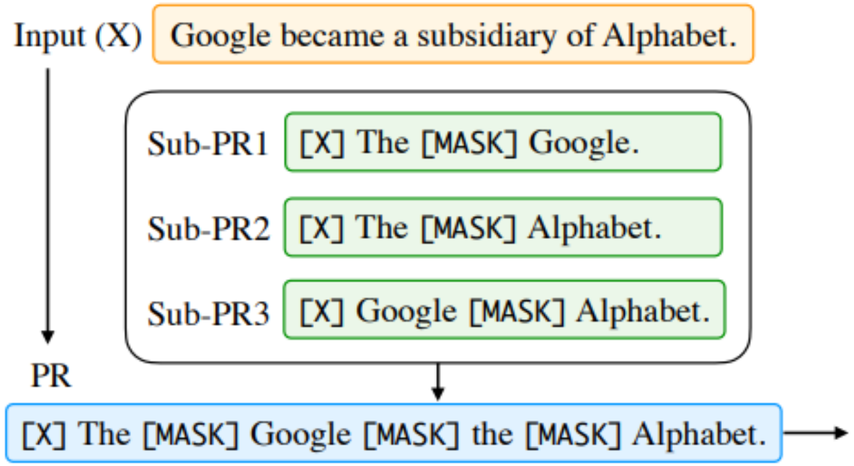
Multi-Prompt Learning



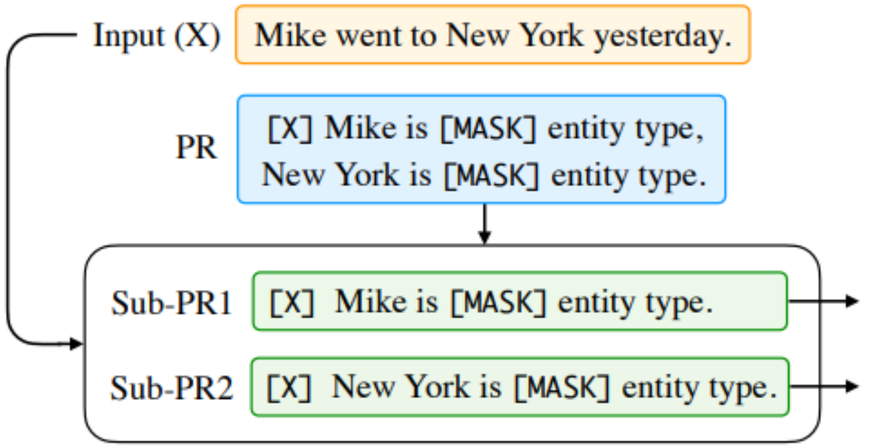
(a) Prompt Ensembling.



(b) Prompt Augmentation.



(c) Prompt Composition.



(d) Prompt Decomposition.

Training Strategies for Prompting Methods

1 Tuning-free Prompting

+ no parameter update process
+ no labels
+ zero-shot settings

-- heavy (prompt/answer) engineering necessary

2 Fixed-LM Prompt Tuning

+ few-shot settings
+ LM parameters remain fixed

-- labels needed
-- tuning prompts/answers

3 Fixed-prompt LM Tuning

+ prompting may add extra value

-- labels needed
-- fine-tuning of LMs

4 Prompt+LM Tuning

+ most expressive method, likely suitable for high-data settings

-- labels needed
-- very costly
-- may overfit on small datasets

To sum up:

Prompt-based learning...

1. aims to **maximize** the use of pretrained LMs.
2. focuses on **adjusting the input** of the task, not the LM.
3. supports effective **zero/one/few-shot** learning settings.
4. is **not** always a **replacement** for **fine-tuning** but rather a yet-another tool.

Questions?

Thank you for your attention!

LinkedIn

pgolik

mjastrzebiowski

References:

[1] 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 Comput. Surv. 55, 9, Article 195 (September 2023), 35 pages. https://doi.org/10.1145/3560815

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. **BERT: Pre-training of deep bidirectional transformers for language understanding.** In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics

[3] Adi Haviv, Jonathan Berant, and Amir Globerson. 2021. **BERTese: Learning to Speak to BERT.** In Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, pages 3618–3623, Online. Association for Computational Linguistics.

[4] Timo Schick and Hinrich Schutze. 2021a. **Exploiting cloze questions for few shot text classification and natural language inference.**

[5]] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. **Unified language model pre-training for natural language understanding and generation.** In Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, pages 13042–13054.