Learning In-context Learning for Named Entity Recognition

Jiawei Chen[1,4], Yaojie Lu[1,†], Hongyu Lin[1] , Jie Lou[3], Wei Jia[3], Dai Dai[3]
Hua Wu[3], Boxi Cao[1,4], Xianpei Han[1,2,†], Le Sun[1,2]

[1]Chinese Information Processing Laboratory [2]State Key Laboratory of Computer Science

Institute of Software, Chinese Academy of Sciences

[3]Baidu Inc., Beijing, China

[4]University of Chinese Academy of Sciences

Outline

- Introduction
- Meta-function pre-training
- Experiments
- Conclusions

Outline

- Introduction
- Meta-function pre-training
- Experiments
- Conclusions

Named entity recognition

- Named entity recognition (NER) aims to detect and classify named entities in text.

Named entity recognition

- Named entity recognition (NER) aims to detect and classify named entities in text.

- **Diversity** of entity types
- **Lack** of high-quality annotations

Few-shot learning

- **Fine-tuning-based methods**
- **Metric-based methods**

Drawbacks of few-shot NER methods

- Fine-tuning-based methods
  - Re-training (expensive for large-scale models)
  - Cannot identify novel types on-the-fly
- Metric-based methods
  - Limited to the architectures
  - Sensitive to the domain shift

Drawbacks of few-shot NER methods

- Fine-tuning-based methods
  - Re-training (expensive for large-scale models)
  - Cannot identify novel types on-the-fly
- Metric-based methods
  - Limited to the architectures
  - Sensitive to the domain shift

➡ **In-context learning**

In-context learning

- The model is given a few demonstrations (and instruction) of the task at inference time as conditioning but no weights are updated.

Input

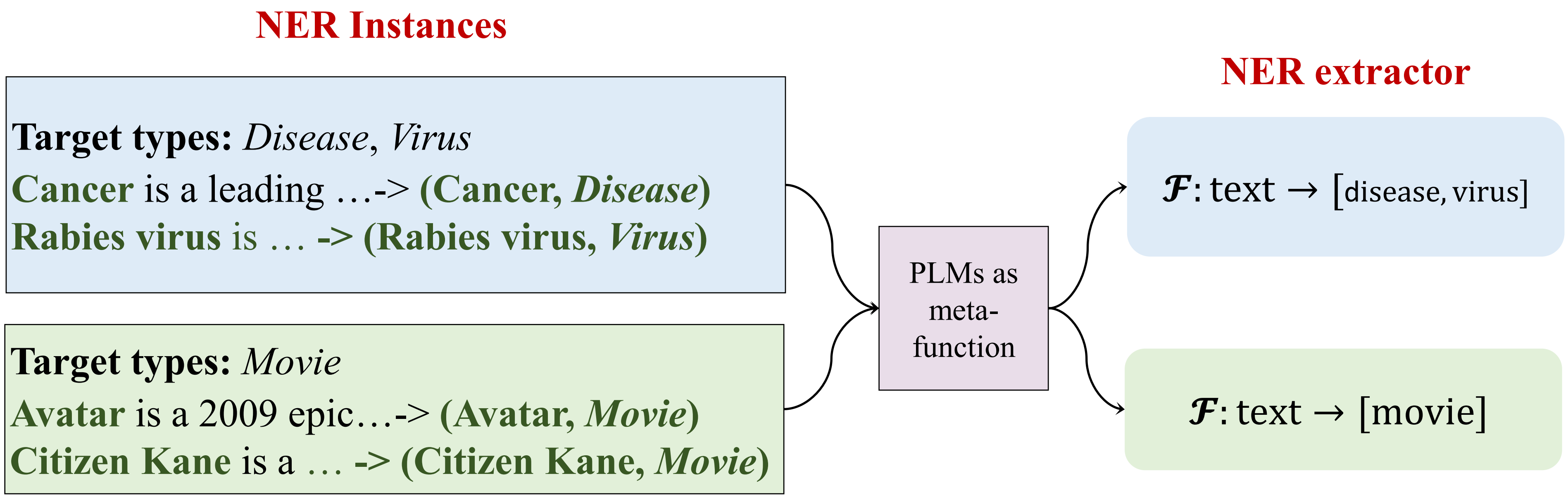
Instruction	**Target types:** *disease; virus*
Demonstrations	**Text:** *Cancer is a leading cause of death worldwide.*
	Entities: *Cancer is disease.*
	Text: *Rabies virus is estimated to cause around 55,000 deaths per year.*
	Entities: *Rabies virus is virus.*
Text	**Text:** *SARS-CoV-2 is a strain of coronavirus that causes COVID-19.*

Output

Entities: *SARS-CoV-2 is virus. COVID-19 is disease.*

Meta-function view

- We model pre-trained language models as a meta-function for NER.
  - **Meta-function**: $\lambda_{instruction,demonstraions,text}.\, M$
  - The new extractor can be implicitly constructed by instruction and demonstrations $(\lambda. M)(insturction, demonstraions) \;\; \rightarrow \;\; \{\mathcal{F}: text \rightarrow entities\}$



Meta-function pre-training

- Inject in-context NER ability into PLMs

In-context learning input

Target types: *disease, virus*
Text: **Cancer** is a leading …
Entities: **Cancer** is *disease.*
Text: **Rabies virus** is estimated …
Entities: **Rabies virus** is *virus*.
Text: SARS-CoV-2 is a strain of coronavirus that causes COVID-19.

→ PLMs as meta-function → $\mathcal{F}$: text $\rightarrow$ [disease, virus]

Meta-function pre-training

- Inject in-context NER ability into PLMs

In-context learning input

Target types: *disease, virus*
Text: **Cancer** is a leading …
Entities: **Cancer** is *disease.*
Text: **Rabies virus** is estimated …
Entities: **Rabies virus** is *virus*.
Text: SARS-CoV-2 is a strain of coronavirus that causes COVID-19.

In-context learning

PLMs as meta-function

$\mathcal{F}$: text → [disease, virus]

Implicitly constructed extractor

Meta-function pre-training

- Inject in-context NER ability into PLMs

In-context learning input

Target types: *disease, virus*
Text: **Cancer** is a leading …
Entities: **Cancer** is *disease.*
Text: **Rabies virus** is estimated …
Entities: **Rabies virus** is *virus*.
Text: SARS-CoV-2 is a strain of coronavirus that causes COVID-19.

In-context learning

PLMs as meta-function

$\mathcal{F}$: text → [disease, virus]

Implicitly constructed extractor

NER Instances

Target types: *Disease*, *Virus*
Cancer is a leading …-> **(Cancer, *Disease*)**
Rabies virus is … **-> (Rabies virus, *Virus*)**

Fine-tuning

Fine-tuned surrogate golden extractor

Meta-function pre-training

- Inject in-context NER ability into PLMs

In-context learning input

Target types: *disease, virus*
Text: **Cancer** is a leading …
Entities: **Cancer** is *disease.*
Text: **Rabies virus** is estimated …
Entities: **Rabies virus** is *virus*.
Text: SARS-CoV-2 is a strain of coronavirus that causes COVID-19.

In-context learning

PLMs as meta-function

$\mathcal{F}$: text → [disease, virus]

Implicitly constructed extractor

NER Instances

Target types: *Disease*, *Virus*
Cancer is a leading …-> **(Cancer, *Disease*)**
Rabies virus is … **-> (Rabies virus, *Virus*)**

Fine-tuning

Fine-tuned surrogate golden extractor

Explicitly fine-tuned extractor

Meta-function pre-training

- Inject in-context NER ability into PLMs

In-context learning input

Target types: *disease, virus*
Text: **Cancer** is a leading …
Entities: **Cancer** is *disease.*
Text: **Rabies virus** is estimated …
Entities: **Rabies virus** is *virus*.
Text: SARS-CoV-2 is a strain of coronavirus that causes COVID-19.

In-context learning

PLMs as meta-function

$\mathcal{F}$: text → [disease, virus]

Implicitly constructed extractor

Close to

NER Instances

Target types: *Disease*, *Virus*
Cancer is a leading …-> **(Cancer, *Disease*)**
Rabies virus is … **-> (Rabies virus, *Virus*)**

Fine-tuning

Fine-tuned surrogate golden extractor

Explicitly fine-tuned extractor

Outline

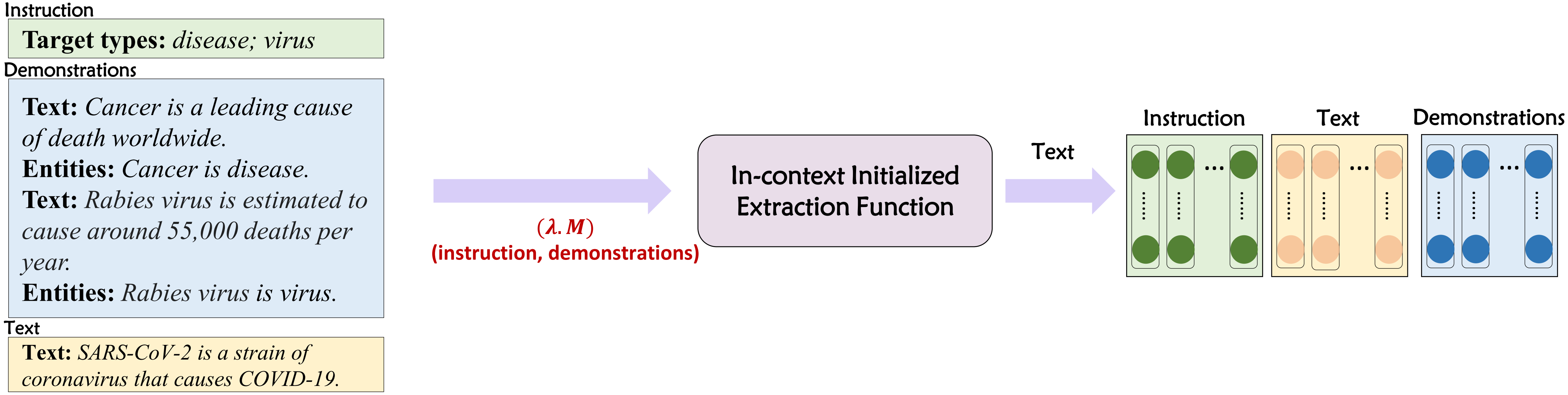
- Introduction
- Meta-function pre-training
- Experiments
- Conclusions

Meta-function pre-training

- Optimizing PLMs via a meta-function loss
  - Implicitly (instruction, demonstration)-constructed extractor will be as close as an explicitly fine-tuned surrogate golden extractor.

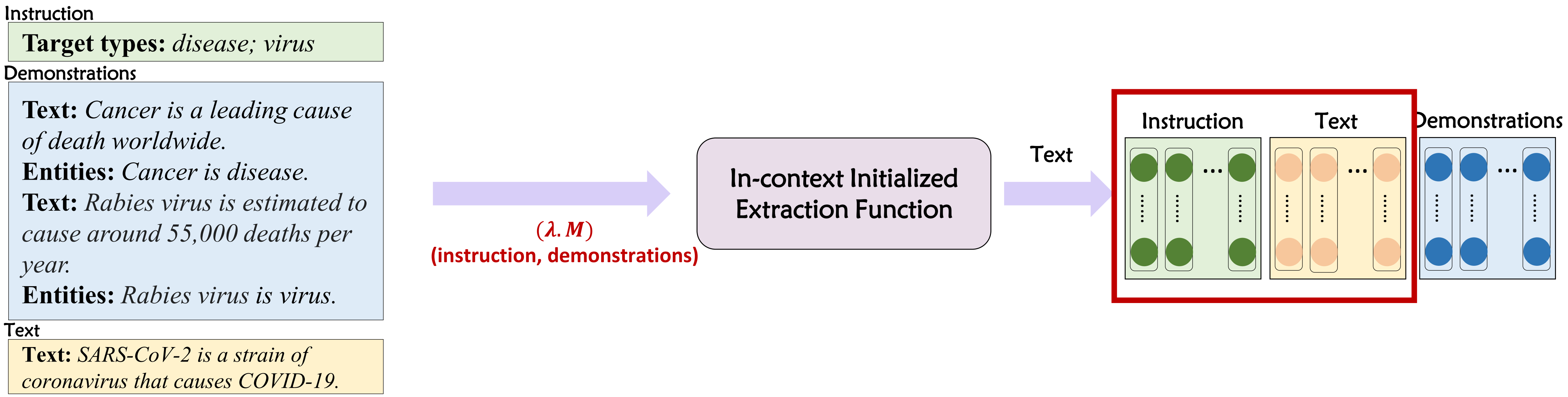
Meta-function pre-training

- Optimizing PLMs via a meta-function loss
  - Implicitly (instruction, demonstration)-constructed extractor will be as close as an explicitly fine-tuned surrogate golden extractor.



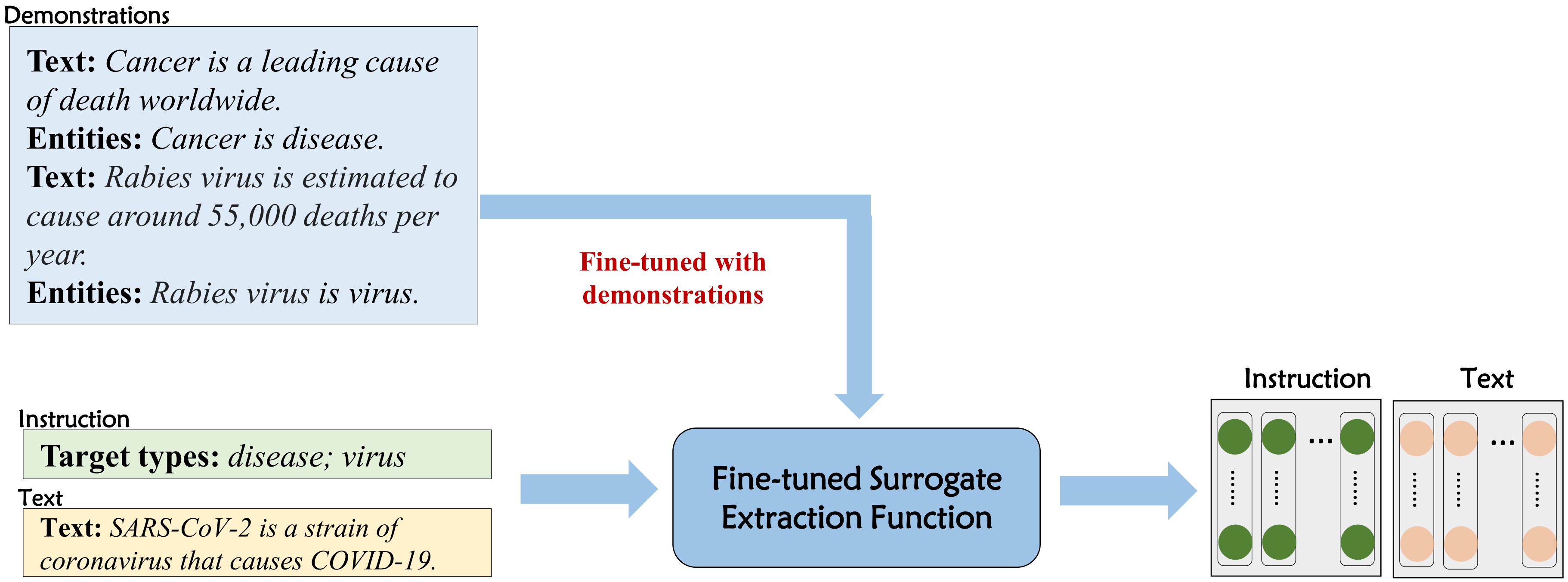
Meta-function pre-training

- Optimizing PLMs via a meta-function loss
  - Implicitly (instruction, demonstration)-constructed extractor will be as close as an explicitly fine-tuned surrogate golden extractor.



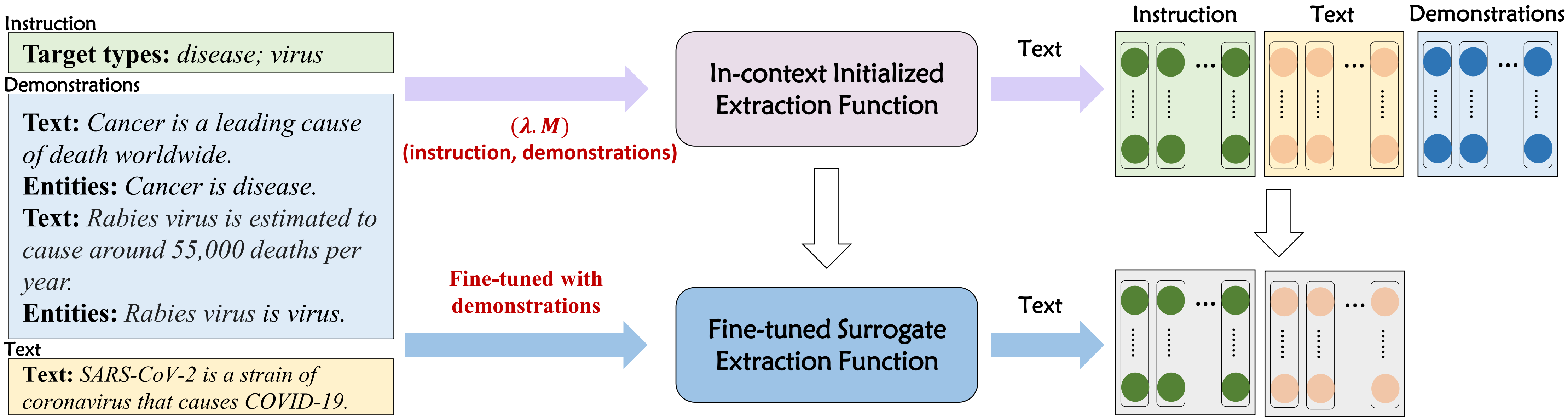
Meta-function pre-training

- Optimizing PLMs via a meta-function loss
  - Implicitly (instruction, demonstration)-constructed extractor will be as close as an explicitly fine-tuned surrogate golden extractor.



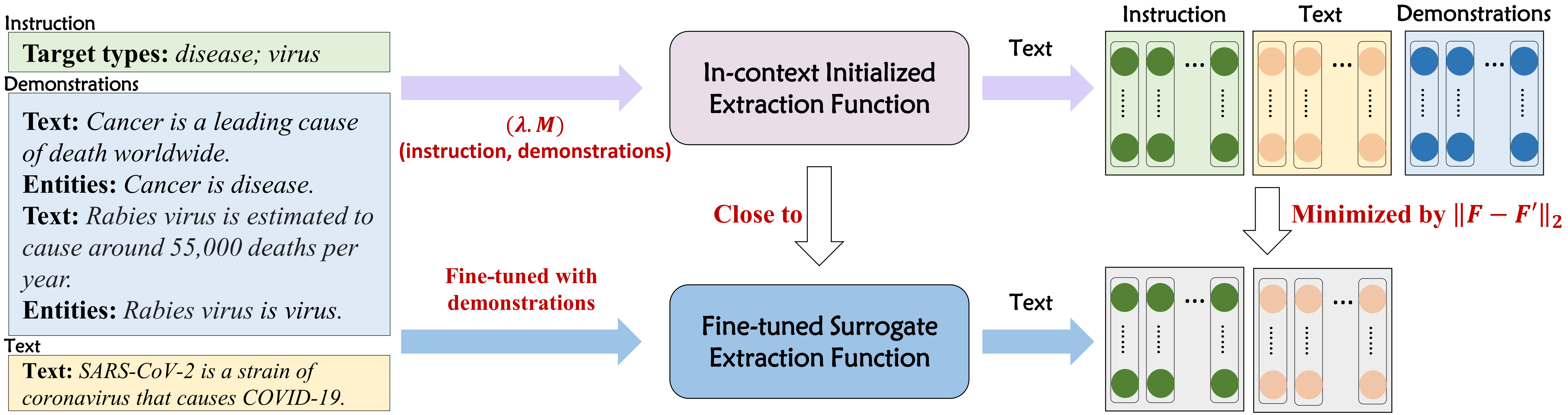
Meta-function pre-training

- Optimizing PLMs via a meta-function loss
  - Implicitly (instruction, demonstration)-constructed extractor will be as close as an explicitly fine-tuned surrogate golden extractor.



Meta-function pre-training

- Optimizing PLMs via a meta-function loss
  - Implicitly (instruction, demonstration)-constructed extractor will be as close as an explicitly fine-tuned surrogate golden extractor.



$$\mathcal{L}_{meta-function} = Average(d(F_{in-context}, F'_{fine-tuned}))$$

Overall pre-training

- Optimizing PLMs via a meta-function loss
- Optimizing PLMs via an extraction loss
  - the sequence-to-sequence entity extractor directly models the generation probability token by token in an auto-regressive way

$$\boldsymbol{\mathcal{L}_{extraction}} = -\log \prod P(y_i | y_{<i}, X, \theta))$$

$$\boldsymbol{\mathcal{L}_{meta-function} = Average(d(F_{in-context}, F'_{fine-tuned}))}$$

Overall pre-training

- Optimizing PLMs via a meta-function loss
- Optimizing PLMs via an extraction loss
  - the sequence-to-sequence entity extractor directly models the generation probability token by token in an auto-regressive way

$$\mathcal{L}_{extraction} = -\log \prod P(y_i | y_{<i}, X, \theta))$$

$$\Downarrow$$

$$\mathcal{L} = \mathcal{L}_{extraction} + \alpha \mathcal{L}_{meta-function}$$

$$\Uparrow$$

$$\mathcal{L}_{meta-function} = Average(d(F_{in-context}, F'_{fine-tuned}))$$

Pre-trained data

- In-context task sampling: sample instance from NER dataset
  - Sample N target entity types and demonstrations
  - Sample text (both positive instance and negative instance)

Pre-trained data

- In-context task sampling: sample instance from NER dataset
  - Sample N target entity types and demonstrations
  - Sample text (both positive instance and negative instance)
- Type anonymization: avoid overfitting to entity type names
  - randomly substituting names with a set of type indicators like <type1>, <type2>, …

Pre-trained data

- In-context task sampling: sample instance from NER dataset
  - Sample N target entity types and demonstrations
  - Sample text (both positive instance and negative instance)
- Type anonymization: avoid overfitting to entity type names
  - randomly substituting names with a set of type indicators like <type1>, <type2>, …
- Entity extraction task
  - we also conduct traditional NER settings (without demonstrations) to improve the ability to extract entities from text when pre-training

Pseudo Extraction Language Modeling Task

- We design a pseudo extraction LM task since NER corpus is usually far smaller than the text corpus.

I think this movie is cool and I really like it very much

Pseudo Extraction Language Modeling Task

- We design a pseudo extraction LM task since NER corpus is usually far smaller than the text corpus.

I think **this movie** is cool and I really **like** it very much

Type anonymization

<type2>: **this movie** <type14>: **like**

Pseudo Extraction Language Modeling Task

- We design a pseudo extraction LM task since NER corpus is usually far smaller than the text corpus.

I think **this movie** is cool and I **like** it very much.

<type2>: **this movie**   <type14>: **like**

Instruction and demonstrations

Target type: <type2>, <type14>
Text: I think **[MASK1]** is cool and I **[MASK2]** it[MASK3].
Entities: [MASK1] is <type2>. [MASK2] is <type14>

Pseudo Extraction Language Modeling Task

- We design a pseudo extraction LM task since NER corpus is usually far smaller than the text corpus.

I think **this movie** is cool and I **like** it very much.

<type2>: **this movie** <type14>: **like**

Input

Target type: <type2>, <type14>
Text: I think **[MASK1]** is cool and I **[MASK2]** it[MASK3].
Entities: [MASK1] is <type2>. [MASK2] is <type14>
Text: I do not like it.

Output

Entities: like is <type14>.

Outline

- Introduction
- Meta-function pre-training
- Experiments
- Conclusions

Baselines

- The baselines include models with different scales and architectures.

- We conduct in-context learning and metric-based few-shot methods.

Models	#Param
T5v1.1-large	770M
GPT2-xl	1.5B
T5-xl	3B
GPT-J-6B	6B
T5-xxl	11B
OPT-13B	13B
GPT-Neox-20B	20B
OPT-30B	30B
OPT-66B	66B
ProtoNet	345M
NNShot	345M
StructShot	345M
CONTAINER	345M

Main result

Models	#Param	CoNLL03		WNUT17		NCBI-disease		SEC-filings		AVE
		1-shot	5-shot	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot	
Pre-trained Language Models										
T5v1.1-large	770M	38.61	44.90	25.52	26.32	26.02	37.63	41.89	53.44	36.79
GPT2-xl	1.5B	33.69	39.55	22.63	24.86	25.54	33.25	42.83	**57.05**	34.93
T5-xl	3B	38.99	45.74	26.39	26.31	23.10	36.78	30.58	42.22	33.76
GPT-J-6B	6B	46.14	50.10	31.41	30.93	35.82	40.98	40.12	39.61	39.39
T5-xxl	11B	40.97	46.14	24.76	25.27	12.19	26.34	32.65	42.44	31.35
OPT-13B	13B	46.65	51.71	27.74	28.36	23.73	34.00	41.60	43.10	37.11
GPT-Neox-20B	20B	52.68	58.12	**36.29**	35.68	35.42	42.85	45.07	45.17	43.91
OPT-30B	30B	42.86	44.77	25.85	27.44	22.31	32.76	40.83	46.52	35.42
OPT-66B	66B	43.83	53.89	30.77	32.00	25.87	34.58	39.15	47.01	38.39
Pre-trained NER Models										
ProtoNet	345M	30.04	60.26	9.74	23.03	24.73	42.32	16.79	23.67	28.82
NNShot	345M	41.92	58.39	15.76	21.78	31.59	33.14	30.19	37.86	33.83
StructShot	345M	42.34	58.44	15.78	22.05	19.87	31.48	30.40	38.44	32.35
CONTAINER	345M	45.43	61.69	15.64	20.37	23.24	27.02	34.07	40.44	33.49
MetaNER-base	220M	53.94	62.59	25.55	30.41	35.00	37.24	46.88	51.39	42.88
MetaNER	770M	**57.40**	**63.45**	31.59	**36.52**	**40.01**	**44.92**	**52.07**	54.87	**47.60**

Meta-function pre-training can effectively inject in-context learning ability into PLMs.

Ablation studies

	CoNLL03			NCBI-disease		
	P	R	F1	P	R	F1
MetaNER	73.59	57.19	**64.34**	**54.96**	**36.85**	**43.79**
w/o MF	68.97	57.62	62.77	38.27	35.26	36.28
w/o LM	70.86	**57.99**	63.77	37.54	34.82	35.67
w/o anonymization	**74.75**	52.86	61.93	47.47	35.30	40.48

MF: meta-function pre-training
LM: pseudo extraction LM task

Ablation studies

	CoNLL03			NCBI-disease		
	P	R	F1	P	R	F1
MetaNER	73.59	57.19	**64.34**	**54.96**	**36.85**	**43.79**
w/o MF	68.97	57.62	62.77	38.27	35.26	36.28
w/o LM	70.86	**57.99**	63.77	37.54	34.82	35.67
w/o anonymization	**74.75**	52.86	61.93	47.47	35.30	40.48

Meta-function pre-training is critical for in-context learning ability

Ablation studies

	CoNLL03			NCBI-disease		
	P	R	F1	P	R	F1
MetaNER	73.59	57.19	**64.34**	**54.96**	**36.85**	**43.79**
w/o MF	68.97	57.62	62.77	38.27	35.26	36.28
w/o LM	70.86	**57.99**	63.77	37.54	34.82	35.67
w/o anonymization	**74.75**	52.86	61.93	47.47	35.30	40.48

The pseudo extraction LM task significantly benefits in-context NER

Ablation studies

	CoNLL03			NCBI-disease		
	P	R	F1	P	R	F1
MetaNER	73.59	57.19	**64.34**	**54.96**	**36.85**	**43.79**
w/o MF	68.97	57.62	62.77	38.27	35.26	36.28
w/o LM	70.86	**57.99**	63.77	37.54	34.82	35.67
w/o anonymization	**74.75**	52.86	61.93	47.47	35.30	40.48

Type name anonymization prevents in-context NER model from type name overfitting, and therefore enhances the in-context learning ability

Outline

- Introduction
- Meta-function pre-training
- Experiments
- Conclusions

Conclusions

- We model PLMs as a meta-function for in-context NER.
- We propose the meta-function pre-training to inject in-context NER ability into PLMs.
- Experimental results show that our method is effective for in-context NER.

Thanks!

Source Code: https://github.com/chen700564/metaner-icl