Few-shot Event Detection: An Empirical Study and a Unified View

Yubo Ma[1], Zehao Wang[2], Yixin Cao[3†], Aixin Sun[1†]

[1] S-Lab, Nanyang Technological University
[2] KU Leuven [3] Singapore Management University
yubo001@e.ntu.edu.sg

Abstract

Few-shot event detection (ED) has been widely studied, while this brings noticeable discrepancies, e.g., various motivations, tasks, and experimental settings, that hinder the understanding of models for future progress. This paper presents a thorough empirical study, a unified view of ED models, and a better *unified baseline*. For fair evaluation, we compare 12 representative methods on three datasets, which are roughly grouped into prompt-based and prototype-based models for detailed analysis. Experiments consistently demonstrate that prompt-based methods, including ChatGPT, still significantly trail prototype-based methods in terms of overall performance. To investigate their superior performance, we break down their design elements along several dimensions and build a unified framework on prototype-based methods. Under such unified view, each prototype-method can be viewed a combination of different modules from these design elements. We further combine all advantageous modules and propose a simple yet effective *baseline*, which outperforms existing methods by a large margin (e.g., 2.7% $F1$ gains under *low-resource* setting). [1]

1 Introduction

Event Detection (ED) is the task of identifying event triggers and types in texts. For example, given *"Cash-strapped Vivendi wants to* sell *Universal Studios"*, it is to classify the word *"sell"* into a *TransferOwnership* event. ED is a fundamental step in various tasks such as successive event-centric information extraction (Huang et al., 2022; Ma et al., 2022b; Chen et al., 2022), knowledge systems (Li et al., 2020; Wen et al., 2021), story generation (Li et al., 2022a), etc. However, the annotation of event instances is costly and labor-consuming, which motivates the research on improving ED with limited labeled samples, i.e., the few-shot ED task.

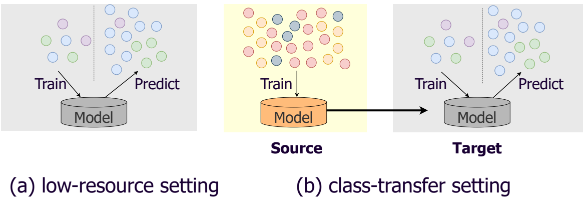


Figure 1: Task settings to access *Generalization* (a) and *Transferability* (b). Colors denote event types.

Extensive studies have been carried out on few-shot ED. Nevertheless, there are noticeable discrepancies among existing methods from three aspects. (1) *Motivation* (Figure 1): Some methods focus on model's *generalization* ability that learns to classify with only a few samples (Li et al., 2022b). Some other methods improve the *transferability*, by introducing additional data, that adapts a well-trained model on the preexisting schema to a new schema using a few samples (Lu et al., 2021). There are also methods considering both abilities (Liu et al., 2020; Hsu et al., 2022). (2) *Task setting*: Even focusing on the same ability, methods might adopt different task settings for training and evaluation. For example, there are at least three settings for transferability: *episode learning* (EL, Deng et al. 2020; Cong et al. 2021), *class-transfer* (CT, Hsu et al. 2022) and *task-transfer* (TT, Lyu et al. 2021; Lu et al. 2022). (3) *Experimental Setting*: Even focusing on the same task setting, their experiments may vary in different sample sources (e.g., a subset of datasets, annotation guidelines, or external corpus) and sample numbers (shot-number or sample-ratio). Table 1 provides a detailed comparison of representative methods.

In this paper, we argue the importance of a unified setting for a better understanding of few-shot ED. First, based on exhaustive background investigation on ED and similar tasks (e.g., NER), we con-

[†] Corresponding Author.

[1] Our code will be publicly available at https://github.com/mayubo2333/fewshot_ED.

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 11211–11236
July 9-14, 2023 ©2023 Association for Computational Linguistics

Table 1: Noticeable discrepancies among existing few-shot ED methods. Explanations of task settings can be found in Section 2.1, which also refer to different motivations: LR for generalization, EL, CT, and TT for transfer abilities. **Dataset** indicates the datasets on which the training and/or evaluation is conducted. **Sample Number** refers to the number of labeled samples used. **Sample Source** refers to where training samples come from. Guidelines: example sentences from annotation guidelines. Datasets: subsets of full datasets. Corpus: (unlabeled) external corpus.

	Method	**Task setting**				**Experimental setting**		
		LR	EL	CT	TT	Dataset	Sample Number	Sample Source
Prototype-based	Seed-based (Bronstein et al., 2015)			✓		ACE	30	Guidelines
	MSEP (Peng et al., 2016)	✓		✓		ACE	0	Guidelines
	ZSL (Huang et al., 2018)			✓		ACE	0	Datasets
	DMBPN (Deng et al., 2020)		✓			FewEvent	{5,10,15}-shot	Datasets
	OntoED (Deng et al., 2021)	✓		✓		MAVEN / FewEvent	{0,1,5,10,15,20}%	Datasets
	Zhang's (Zhang et al., 2021)	✓				ACE	0	Corpus
	PA-CRF (Cong et al., 2021)		✓			FewEvent	{5,10}-shot	Datasets
	ProAcT (Lai et al., 2021)		✓			ACE / FewEvent / RAMS	{5,10}-shot	Datasets
	CausalED (Chen et al., 2021)		✓			ACE / MAVEN / ERE	5-shot	Datasets
	Yu's (Yu et al., 2022)	✓				ACE	176	Guidelines + Corpus
	ZED (Zhang et al., 2022a)	✓				MAVEN	0	Corpus
	HCL-TAT (Zhang et al., 2022b)		✓			FewEvent	{5,10}-shot	Datasets
	KE-PN (Zhao et al., 2022)		✓			ACE / MAVEN / FewEvent	{1,5}-shot	Datasets
Prompt-based	EERC (Liu et al., 2020)	✓		✓	✓	ACE	{0,1,5,10,20}%	Datasets
	FSQA (Feng et al., 2020)	✓			✓	ACE	{0,1,3,5,7,9}-shot	Datasets
	EDTE (Lyu et al., 2021)				✓	ACE / ERE	0	-
	Text2Event (Lu et al., 2021)			✓		ACE / ERE	{1,5,25}%	Datasets
	UIE (Lu et al., 2022)	✓		✓		ACE / CASIE	{1,5,10}-shot/%	Datasets
	DEGREE (Hsu et al., 2022)	✓		✓		ACE / ERE	{0,1,5,10}-shot	Datasets
	PILED (Li et al., 2022b)	✓	✓			ACE / MAVEN / FewEvent	{5,10}-shot	Datasets

duct **an empirical study of twelve SOTA methods under two practical settings**: *low-resource* setting for *generalization* ability and *class-transfer* setting for *transferability*. We roughly classify the existing methods into two groups: prototype-based models to learn event-type representations and proximity measurement for prediction and prompt-based models that convert ED into a familiar task of Pre-trained Language Models (PLMs).

The second contribution is **a unified view of prototype-based methods** to investigate their superior performance. Instead of picking up the best-performing method as in conventional empirical studies, we take one step further. We break down the design elements along several dimensions, e.g., the source of prototypes, the aggregation form of prototypes, etc. From this perspective, five prototype-based methods on which we conduct experiment are instances of distinct modules from these elements. And third, through analyzing each effective design element, we propose **a simple yet effective *unified baseline*** that combines all advantageous elements of existing methods. Experiments validate an average 2.7% $F1$ gains under *low-resource* setting and the best performance under *class-transfer* setting. Our analysis also provides many valuable insights for future research.

2 Preliminary

Event detection (ED) is usually formulated as either a span classification task or a sequence labeling task, depending on whether candidate event spans are provided as inputs. We brief the sequence labeling paradigm here because the two paradigms can be easily converted to each other.

Given a dataset $\mathcal{D}$ annotated with schema E (the set of event types) and a sentence $X = [x_1, ..., x_N]^T \in \mathcal{D}$, where x_i is the i-th word and N the length of this sentence, ED aims to assign a label $y_i \in (E \cup \{\texttt{N.A.}\})$ for each x_i in X. Here $\texttt{N.A.}$ refers to either none events or events beyond pre-defined types E. We say that word x_i triggering an event y_i if $y_i \in E$.

2.1 Few-shot ED task settings

We categorize few-shot ED settings to four cases: *low-resource* (LR), *class-transfer* (CT), *episode learning* (EL) and *task-transfer* (TT). Low-resource setting assesses the *generalization* ability of few-shot ED methods, while the other three settings are for *transferability*. We adopt LR and CT in our empirical study towards practical scenarios. More details can be found in Appendix A.1.

Low-resource setting assumes access to a dataset $\mathcal{D} = (\mathcal{D}_{train}, \mathcal{D}_{dev}, \mathcal{D}_{test})$ annotated with a label set E, where $|\mathcal{D}_{dev}| \leq |\mathcal{D}_{train}| \ll |\mathcal{D}_{test}|$. It assesses the generalization ability of models by (1) utilizing only few samples during training, and (2) evaluating on the real and rich test dataset.

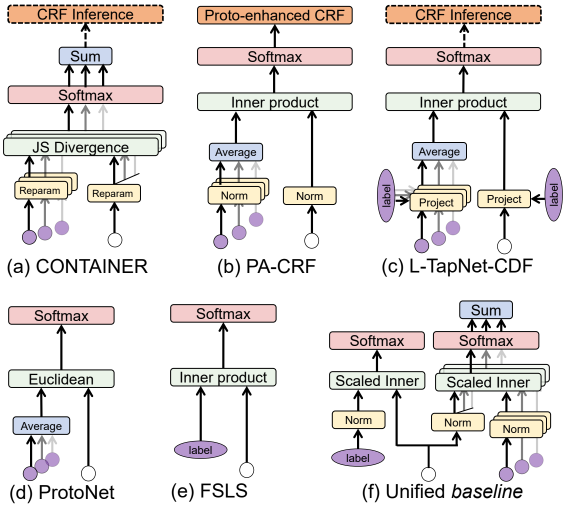
Class-transfer setting assumes access to a source dataset $\mathcal{D}^{(S)}$ with a preexisting schema $E^{(S)}$ and a target dataset $\mathcal{D}^{(T)}$ with a new schema $E^{(T)}$. Note that $D^{(S)}$ and $D^{(T)}$, $E^{(S)}$ and $E^{(T)}$ contain disjoint sentences and event types, respectively. $\mathcal{D}^{(S)}$ contains abundant samples, while $\mathcal{D}^{(T)}$ is the low-resource setting dataset described above. Models under this setting are expected to be pre-trained on $\mathcal{D}^{(S)}$ then further trained and evaluated on $\mathcal{D}^{(T)}$.

2.2 Category of existing methods

We roughly group existing few-shot ED methods into two classes: prompt-based methods and prototype-based methods. More details are introduced in Appendix A.2.

Prompt-based methods leverage the rich language knowledge in PLMs by converting downstream tasks to the task with which PLMs are more familiar. Such format conversion narrows the gap between pre-training and downstream tasks and benefits knowledge induction in PLMs with limited annotations. Specifically, few-shot ED can be converted to machine reading comprehension (MRC, Du and Cardie 2020; Liu et al. 2020; Feng et al. 2020), natural language inference (NLI, Lyu et al. 2021), conditional generation (CG, Paolini et al. 2021; Lu et al. 2021, 2022; Hsu et al. 2022), and the cloze task (Li et al., 2022b). We give examples of these prompts in Table 6.

Prototype-based methods predict an event type for each word/span mention by measuring its representation proximity to *prototypes*. Here we define prototypes in a *generalized* format — it is an embedding that represents some event type. For example, Prototypical Network (ProtoNet, Snell et al. 2017) and its variants (Lai et al., 2020a,b; Deng et al., 2020, 2021; Cong et al., 2021; Lai et al., 2021) construct prototypes via a subset of sample mentions. In addition to event mentions, a line of work leverage related knowledge to learn or enhance prototypes' representation, including AMR graphs (Huang et al., 2018), event-event relations (Deng et al., 2021), definitions (Shen et al., 2021) and FrameNet (Zhao et al., 2022). Zhang et al. (2022b) recently introduce contrastive learning (Hadsell et al., 2006) in few-shot ED task. Such method also determines the event by measuring the distances with other samples and aggregates these distances to evaluate an overall distance to each event type. Therefore we view it as a *generalized* format of prototype-based methods as well.



(a) CONTAINER (b) PA-CRF (c) L-TapNet-CDF

(d) ProtoNet (e) FSLS (f) Unified *baseline*

Figure 2: The architectures of five existing prototype-based methods and the unified baseline. Given event mention x and event type y, each sub-figure depicts how to compute the logits$(y|x)$. White circles: representation of predicted event h_x. Purple circles: representation of prototypes h_{c_y} ($c_y \in \mathcal{C}_y$). Yellow modules: transfer functions. Green modules: distance functions. Blue modules: aggregation form. Orange modules: CRF modules. Dashed lines in (a) and (c) represent that their CRFs are only used during inference.

For comprehensiveness, we also include competitive methods from similar tasks, *i.e.,* Named Entity Recognition and Slot Tagging, which are highly adaptable to ED. Such expansion enriches the categorization and enables us to build a unified view in Section 3. For instance, some methods (Hou et al., 2020; Ma et al., 2022a) leverage label semantics to enhance or directly construct the prototypes. Others (Das et al., 2022) leverage contrastive learning for better prototype representations.

3 A Prototype-based Unified View

Due to the superior performance (Sections 5 and 6), we zoom into prototype-based methods to provide a unified view towards a better understanding. We observe that they share lots of similar components. As shown in Table 2 and Figure 2, we decompose prototype-based methods into 5 design elements: prototype source, transfer function, distance function, aggregation form, and CRF module. This unified view enables us to compare choices in each design element directly. By aggregating the

Table 2: Decomposing five prototype-based methods and *unified baseline* along design elements. "Both" in column 1 means both event mentions and label names for y are prototype sources. JSD: Jensen–Shannon divergence. $\mathcal{M}$: Projection matrix in TapNet. $\mathcal{N}(\mu(h), \Sigma(h))$: Gaussian distribution with mean $\mu(h)$ and covariance matrix $\Sigma(h)$.

Method	Prototype $\mathcal{C}_y$	Aggregation	Distance $d(u, v)$	Transfer $f(h)$	CRF Module
ProtoNet (Snell et al., 2017)	Event mentions	feature	$\|\|u-v\|\|_2$	h	–
L-TapNet-CDT (Hou et al., 2020)	Both	feature	$-u^T v/\tau$	$\mathcal{M}\frac{h}{\|\|h\|\|}$	CRF-Inference
PA-CRF (Cong et al., 2021)	Event mentions	feature	$-u^T v$	$\frac{h}{\|\|h\|\|}$	CRF-PA
CONTAINER (Das et al., 2022)	Event mentions	score	JSD$(u\|\|v)$	$\mathcal{N}(\mu(h), \Sigma(h))$	CRF-Inference
FSLS (Ma et al., 2022a)	Label name	–	$-u^T v$	h	–
Unified Baseline (Ours)	Both	score + loss	$-u^T v/\tau$	$\frac{h}{\|\|h\|\|}$	–

effective choices, we end with a *Unified Baseline*.

Formally, given an event mention x, prototype-based methods predict the likelihood $p(y|x)$ from logits$(y|x)$ for each $y \in (E \cup \{\texttt{N.A.}\})$

$$p(y|x) = \text{Softmax}_{y\sim(E\cup\{\texttt{N.A.}\})}\text{logits}(y|x)$$

The general framework is as follows. Denote the PLM's output representation of event mention x and data c_y in prototype source $\mathcal{C}_y$ as h_x and h_{c_y} respectively, where $h \in R^m$ and m is the dimension of PLM's hidden space. The first step is to convert h_x and h_{c_y} to appropriate representations via a transfer function $f(\cdot)$. Then the methods maintain either a single or multiple prototypes c_y's for each event type, determined by the adopted aggregation form. Third, the distance between $f(h_x)$ and $f(h_{c_y})$ (single prototype) or $f(h_{c_y})$'s (multiple prototypes) is computed via a distance function $d(\cdot, \cdot)$ to learn the proximity scores, *i.e.,* logits$(y|x)$. Finally, an optional CRF module is used to adjust logits$(y|x)$ for x in the same sentence to model their label dependencies. For inference, we adopt nearest neighbor classification by assigning the sample with nearest event type in $\cup_{y\in(E\cup\{\texttt{N.A.}\})}\mathcal{C}_y$, *i.e.,*

$$\hat{y}_x = \underset{y\in(E\cup\{\texttt{N.A.}\})}{\text{argmin}} \min_{c_y\in\mathcal{C}_y} d(f(h_x), f(h_{c_y}))$$

Next, we detail the five design elements:

Prototype source $\mathcal{C}_y$ (purple circles in Figure 2, same below) indicates a set about the source of data / information for constructing the prototypes. There are mainly two types of sources:

(1) *event mentions* (purple circle without words): ProtoNet and its variants in Figure 2(b),(c),(d) additionally split a support set $\mathcal{S}_y$ from training data as prototype source, while contrastive learning methods in Figure 2(a) view every annotated mention as the source (except the query one).

(2) *Label semantics* (purple ellipses with words): Sometimes, the label name l_y is utilized as the source to enhance or directly construct the prototypes. For example, FSLS in Figure 2(e) views the text representation of type names as prototypes, while L-TapNet-CDT in Figure 2(c) utilizes both the above kinds of prototype sources.

Transfer function $f : R^m \to R^n$ (yellow modules) transfers PLM outputs into the distance space for prototype proximity measurement. Widely used transfer functions include normalization in Figure 2(b), down-projection in Figure 2(c), reparameterization in Figure 2(a), or an identity function.

Distance function $d : R^n \times R^n \to R_+$ (green modules) measures the distance of two transferred representations within the same embedded space. Common distance functions are euclidean distance in Figure 2(d) and negative cosine similarity in Figure 2(b),(c),(e).

Aggregation form (blue modules) describes how to compute logits$(y|x)$ based on a single or multiple prototype sources. Aggregation may happen at three levels.

(1) *feature-level*: ProtoNet and its variants in Figure 2(b),(c),(d) aims to construct a *single* prototype $h_{\bar{c}_y}$ for each event type y by merging various features, which ease the calculation logits$(y|x) = -d(f(h_x), f(h_{\bar{c}_y}))$.

(2) *score-level*: CONTAINER in Figure 2(a) views each data as a prototype (they have *multiple* prototypes for each type y) and computes the distance $d(f(h_x), f(h_{c_y}))$ for each $c_y \in \mathcal{C}_y$. These distances are then merged to obtain logits$(y|x)$.

(3) *loss-level*: Such form has multiple parallel branches b for each mention x. Each branch has its own logits$^{(b)}(y|x)$ and is optimized with different loss components during training. Thus it could be viewed as a multi-task learning format. See *unified baseline* in Figure 2(f).

CRF module (orange modules) adjusts predictions within the same sentence by explicitly considering the label dependencies between sequential inputs. The vanilla CRF (Lafferty et al., 2001) and its variants in Figure 2(a),(b),(c) post additional constraints into few-shot learning.

4 Experimental setup

4.1 Few-shot datasets and Evaluation

Dataset source. We utilize ACE05 (Doddington et al., 2004), MAVEN (Wang et al., 2020) and ERE (Song et al., 2015) to construct few-shot ED datasets in this empirical study. Detailed statistics about these three datasets are in Appendix B.1.

Low-resource setting. We adopt K-shot sampling strategy to construct few-shot datasets for the low-resource setting, i.e., sampling K_{train} and K_{dev} samples per event type to construct the train and dev sets, respectively.[2] We set three (K_{train}, K_{dev}) in our evaluation: (2, 1), (5, 2) and (10, 2). We follow Yang and Katiyar (2020) taking a greedy sampling algorithm to approximately select K samples for each event type. See Appendix B.2 for details and the statistics of the sampled few-shot datasets. We inherit the original test set as $\mathcal{D}_{test}$.

Class-transfer setting. The few-shot datasets are curated in two sub-steps: (1) Dividing both event types and sentences in the original dataset into two disjoint parts, named *source dataset* and *target dataset pool*, respectively. (2) Sampling few-shot samples from the target dataset pool to construct target dataset. The same sampling algorithm as in *low-resource* setting is used. Then we have the source dataset and the sampled target dataset. See Appendix B.2 for details and the statistics of the sampled few-shot datasets.

Evaluation Metric We use micro-$F1$ score as the evaluation metric. To reduce the random fluctuation, the reported values of each setting are the averaged score and sample standard deviation, of results w.r.t 10 sampled few-shot datasets.

4.2 Evaluated methods

We evaluate 12 representative methods, including vanilla fine-tuning, in-context learning, 5 prompt-based and 5 prototype-based methods. These methods are detailed in Appendix B.3.

Fine-tuning To validate the effectiveness of few-shot methods, we fine-tune a supervised classifier for comparison as a trivial baseline.

In-context learning To validate few-shot ED tasks still not well-solved in the era of Large Language Models (LLMs), we design such baseline instructing LLMs to detect event triggers by the means of in-context learning (ICL).

Prompt-based (1) *EEQA* (QA-based, Du and Cardie 2020), (2) *EETE* (NLI-based, Lyu et al. 2021), (3) *PTE* (cloze task, Schick and Schütze 2021), (4) *UIE* (generation, Lu et al. 2022) and (5) *DEGREE* (generation, Hsu et al. 2022).

Prototype-based (1) *ProtoNet* (Snell et al., 2017), (2) *L-TapNet-CDT* (Hou et al., 2020), (3) *PA-CRF* (Cong et al., 2021), (4) *CONTAINER* (Das et al., 2022) and (5) *FSLS* (Ma et al., 2022a). See Table 2 and Figure 2 for more details.

4.3 Implementation details

We unify PLMs in each method as much as possible for a fair comparison in our empirical study. Specifically, we use `RoBERTa-base` (Liu et al., 2019) for all prototype-based methods and three non-generation prompt-based methods. However, we keep the method's original PLM for two prompt-based methods with generation prompt, UIE (`T5-base`, Raffel et al. 2020) and DEGREE (`BART-large`, Lewis et al. 2020). We observe their performance collapses with smaller PLMs. Regarding ICL method, we use ChatGPT (`gpt-3.5-turbo-0301`) as the language model. See more details in Appendix B.4.

5 Results: Low-resource Learning

5.1 Overall comparison

We first overview the results of the 12 methods under the low-resource setting in Table 3.

Fine-tuning. Despite its simpleness, fine-tuning achieves acceptable performance. In particular, it is even comparable to the strongest existing methods on MAVEN dataset, only being 1.1% and 0.5% less under 5-shot and 10-shot settings. One possible reason that fine-tuning is good on MAVEN is that MAVEN has 168 event types, much larger than others. When the absolute number of samples is relatively large, PLMs might capture implicit interactions among different event types, even though the samples per event type are limited. When the

[2] Recent systematic research on few-shot NLP tasks (Perez et al., 2021) is of opposition to introducing an additional dev set for few-shot learning. We agree with their opinion but choose to keep a **very small** dev set mainly for feasibility consideration. Given the number of experiments in our empirical study, it is infeasible to conduct cross-validation on every single train set for hyperparameter search.

Table 3: Overall results of *fine-tuning* method, 10 existing few-shot ED methods, and the *unified baseline* under low-resource setting. The best results are in bold face and the second best are underlined. The results are averaged over 10 repeated experiments, and sample standard deviations are in the round bracket. The standard deviations are derived from different **sampling few-shot datasets** instead of **random seeds**. Thus high standard deviation values do not mean that no significant difference among these methods.

	Method	**ACE05**			**MAVEN**			**ERE**		
		2-shot	5-shot	10-shot	2-shot	5-shot	10-shot	2-shot	5-shot	10-shot
	Fine-tuning	33.3(4.4)	42.5(4.6)	48.2(1.5)	40.8(4.7)	52.1(0.7)	55.7(0.2)	32.9(2.1)	39.8(2.9)	43.6(1.7)
	In-context Learning	38.9(3.0)	34.3(1.2)	36.7(0.8)	22.1(1.0)	22.7(0.3)	23.9(0.7)	24.2(3.3)	26.0(0.7)	25.5(1.7)
Prompt-based	EEQA	24.1(12.2)	43.1(2.7)	48.3 (2.4)	33.4(9.2)	48.1(0.9)	52.5(0.5)	13.7(8.6)	34.4(1.7)	39.8(2.4)
	EETE	15.7(0.6)	19.1(0.3)	21.4(0.2)	28.9(4.3)	30.6(1.3)	32.5(1.1)	10.6(2.3)	12.8(2.2)	13.7(2.8)
	PTE	38.4(4.2)	42.6(7.2)	49.8(1.9)	41.3(1.4)	46.0(0.6)	49.5(0.6)	33.4(2.8)	36.9(1.3)	37.0(1.8)
	UIE	29.3(2.9)	38.3(4.2)	43.4(3.5)	33.7(1.4)	44.4(0.3)	50.5(0.5)	19.7(1.5)	30.8(1.9)	34.1(1.6)
	DEGREE	40.0(2.9)	45.5(3.2)	48.5(2.1)	43.3(1.0)	43.4(5.9)	45.5(4.3)	31.3(3.1)	36.0(4.6)	40.7(2.2)
Prototype-bsd	ProtoNet	38.3(5.0)	47.2(3.9)	52.3(2.4)	44.5(2.2)	51.7(0.6)	55.4(0.2)	31.6(2.7)	39.7(2.4)	44.3(2.3)
	PA-CRF	34.9(7.2)	48.1(3.9)	51.7(2.6)	44.8(2.2)	51.8(1.0)	55.3(0.4)	30.6(2.8)	38.0(3.9)	40.4(2.0)
	L-TapNet-CDT	43.2(3.8)	49.8(2.9)	53.5(3.4)	48.6(1.2)	53.2(0.4)	56.1(0.9)	35.6(2.6)	42.7(1.7)	45.1(3.2)
	CONTAINER	40.1(3.8)	47.7(3.3)	50.1(1.8)	44.2(1.4)	50.8(0.9)	52.9(0.3)	34.4(3.6)	39.3(1.9)	44.5(2.3)
	FSLS	39.2(3.4)	47.5(3.2)	51.9(1.7)	46.7(1.2)	51.5(0.5)	56.2(0.2)	34.5(3.1)	39.8(2.5)	44.0(2.0)
	Unified Baseline	**46.0**(4.6)	**54.4**(2.6)	**56.7**(1.5)	**49.5**(1.7)	**54.7**(0.8)	**57.8**(1.2)	**38.8**(2.4)	**45.5**(2.8)	**48.4**(2.6)

sample number is scarce, however, fine-tuning is much poorer than existing competitive methods (see ACE05). Thus, we validate the necessity and progress of existing few-shot methods.

In-context learning. We find the performance of ICL-based methods lags far behind that of tuning-required methods, though the backbone of ICL approach (ChatGPT) is much larger than other PLMs (<1B). A series of recent work (Ma et al., 2023; Gao et al., 2023; Zhan et al., 2023) observe the similar results as ours [3]. Thus we validate few-shot ED tasks could not be solved smoothly by cutting-edge LLMs and deserves further exploration.

Prompt-based methods. Prompt-based methods deliver much poorer results than expected, even compared to fine-tuning, especially when the sample number is extremely scarce. It shows designing effective prompts for ED tasks with very limited annotations is still challenging or even impossible. We speculate it is due to the natural gap between ED tasks and pre-training tasks in PLMs.

Among prompt-based methods, PTE and DEGREE achieve relatively robust performance under all settings. DEGREE is advantageous when the sample size is small, but it cannot well handle a dataset with many event types like MAVEN. When sample sizes are relatively large, EEQA shows competitive performance as well.

5.2 Prototype-based methods

Since prototype-based methods have overall better results, we zoom into the design elements to search for effective choices based on the unified view.

Transfer function, Distance function, and CRF. We compare combinations of transfer and distance functions and four variants of CRF modules in Appendices C.1 and C.2. We make two findings: (1) A scaled coefficient in the distance function achieves better performance with the normalization transfer function. (2) There is no significant difference between models with or without CRF modules. Based on these findings, we observe a significant improvement in five existing methods by simply substituting their d and f for more appropriate choices, see Figure 3 and Appendix C.1. We would use these new transfer and distance functions in further analysis and discussion.

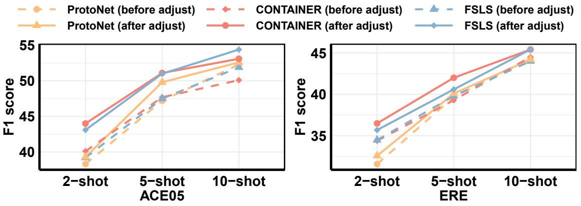


Figure 3: Results of existing methods *before* (dashed lines) and *after* (solid lines) adjustment that substitute their transfer and distance functions to appropriate ones. See full results in Table 8.

Prototype Source. We explore whether label semantic and event mentions are complementary pro-

[3] We refer readers to Ma et al. (2023) for a more detailed discussion on why ICL approaches stumble across few-shot ED tasks.

totype sources, i.e., whether utilizing both achieves better performance than either one. We choose ProtoNet and FSLS as base models which contain only a single kind of prototype source (mentions or labels). Then we combine the two models using three aggregating forms mentioned in Section 3 and show their results in Figure 4. Observe that: (1) leveraging label semantics and mentions as prototype sources simultaneously improve the performance under almost all settings, and (2) merging the two kinds of sources at loss level is the best choice among three aggregation alternatives.

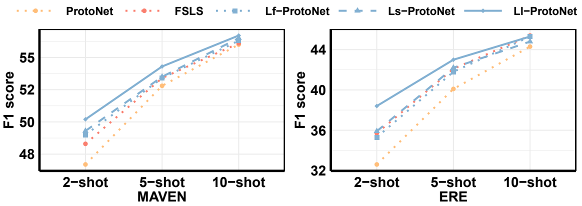


Figure 4: Results of three approaches aggregating label semantics and event mentions on MAVEN and ERE few-shot datasets. **Lf**: feature-level. **Ls**: score-level. **Ll**: loss-level. See full results in Table 9.

Contrastive or Prototypical Learning. Next, we investigate the effectiveness of contrastive learning (CL, see CONTAINER) and prototypical learning (PL, see ProtoNet and its variants) for event mentions. We compare three label-enhanced (since we have validated the benefits of label semantics) methods aggregating event mentions with different approaches. (1) *Ll-ProtoNet*: the strongest method utilizing PL in last part. (2) *Ll-CONTAINER*: the method utilizing in-batch CL as CONTAINER does. (3) *Ll-MoCo*: the method utilizing CL with MoCo setting (He et al., 2020). The in-batch CL and MoCo CL are detailed in Appendix C.4.

Figure 5 suggests CL-based methods outperform Ll-ProtoNet. There are two possible reasons: (1) CL has higher sample efficiency since every two samples interact during training. PL, however, further splits samples into support and query set during training; samples within the same set are not interacted with each other. (2) CL adopts score-level aggregation while PL adopts feature-level aggregation. We find the former also slightly outperforms the latter in Figure 4. We also observe that MoCo CL usually has a better performance than in-batch CL when there exists complicated event types (see MAVEN), or when the sample number is relatively large (see ACE 10-shot). We provide a more detailed explanation in Appendix C.4.

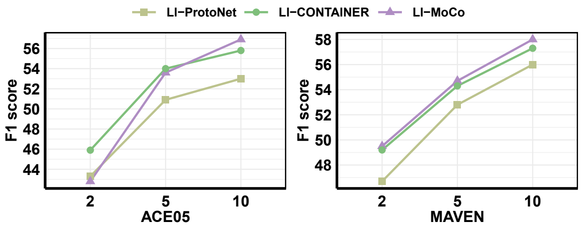


Figure 5: Results of (label-enhanced) PL and CL methods on ACE05 and MAVEN few-shot datasets. See full results on three datasets in Table 10.

5.3 The unified baseline

Here is a summary of the findings: (1) Scaled euclidean or cosine similarity as distance measure with normalized transfer benefits existing methods. (2) CRF modules show no improvement in performance. (3) Label semantic and event mentions are complementary prototype sources, and aggregating them at loss-level is the best choice. (4) As for the branch of event mentions, CL is more advantageous than PL for few-shot ED tasks. (5) MoCo CL performs better when there are a good number of sentences, otherwise in-batch CL is better.

Based on these findings, we develop a simple but effective *unified baseline* as follows. We utilize both label semantic and event mentions as prototype sources and aggregate two types of sources at loss-level. Specifically, we assign two branches with their own losses for label semantic and event mentions respectively. Both two branches adopt scaled cosine similarity $d_\tau(u, v) = -\frac{u^T v}{\tau}$ as distance measure and normalization $f(h) = h/\|h\|_2$ as transfer function. We do not add CRF modules.

For label semantic branch, we follow FSLS and set the embeddings of event name as prototypes. Here h_x and h_{e_y} represent the PLM representation of event mention x and label name e_y, respectively.

$$e_y = \text{Event\_name}(y)$$
$$\text{logits}^{(l)}(y|x) = -d_\tau(f(h_x), f(h_{e_y}))$$

For event mention branch, we adopt CL which aggregates prototype sources (event mentions) at score-level. If the total sentence number in train set is smaller than 128, we take in-batch CL (CONTAINER) strategy as below:

$$\text{logits}^{(m)}(y|x) = \sum_{x' \in \mathcal{S}_y(x)} \frac{-d(f(h_x), f(h_{x'}))}{|\mathcal{S}_y(x)|}$$

$\mathcal{S}_y(x) = \{x' | (x', y') \in D, y' = y, x' \neq x\}$ is the set of all other mentions with the same label.

If the total sentence number in train set is larger than 128, we instead take MoCo CL maintaining a queue for $\mathcal{S}_y(x)$ and a momentum encoder.

We then calculate the losses of these two branches and merge them for joint optimization:

$$p^{(l/m)}(y|x) = \text{Softmax}_y[\text{logits}^{(l/m)}(y|x)]$$

$$L^{(l/m)}(y|x) = -\sum_{(x,y)} y\log(p^{(l/m)}(y|x))$$

$$L = L^{(l)} + L^{(m)}$$

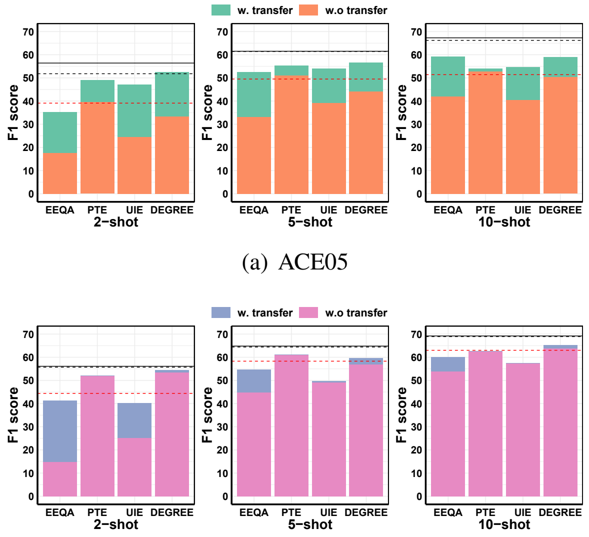
The diagram of the *unified baseline* is illustrated in Figure 2(f) and its performance is shown in Table 3. Clearly, *unified baseline* outperforms all existing methods significantly, 2.7% $F1$ gains on average, under all low-resource settings.

6 Results: Class-transfer Learning

In this section, we evaluate existing methods and the *unified baseline* under class-transfer setting. Here we do not consider in-context learning because previous expetiments show it still lags far from both prompt- and prototype-based methods.

6.1 Prompt-based methods

We first focus on 4 existing prompt-based methods and explore whether they could smoothly transfer event knowledge from a preexisting (source) schema to a new (target) schema. We show results in Figure 6 and Appendix D.1. The findings are summarized as follows. (1) The transfer of knowledge from source event types to target event types facilitates the model prediction under most scenarios. It verifies that an appropriate prompt usually benefits inducing the knowledge learned in PLMs. (2) However, such improvement gradually fades with the increase of sample number from either source or target schema. For example, the 5-shot v.s 10-shot performance for PTE and UIE are highly comparable. We speculate these prompts act more like a catalyst: they mainly teach model how to induce knowledge from PLMs themselves rather than learn new knowledge from samples. Thus the performance is at a standstill once the sample number exceeds some threshold. (3) Overall, the performance of prompt-based methods remains inferior to prototype-based methods in class-transfer setting (see black lines in Figure 6). Since similar results are observed in low-resource settings as well, we conclude that prototype-based methods are better few-shot ED task solver.



(a) ACE05

(b) MAVEN

Figure 6: Class-transfer results of prompt-based methods. We plot *fine-tuning* (red dash lines), best and second best prototype-based methods (black solid/dash lines) for comparison. See full results in Table 11.

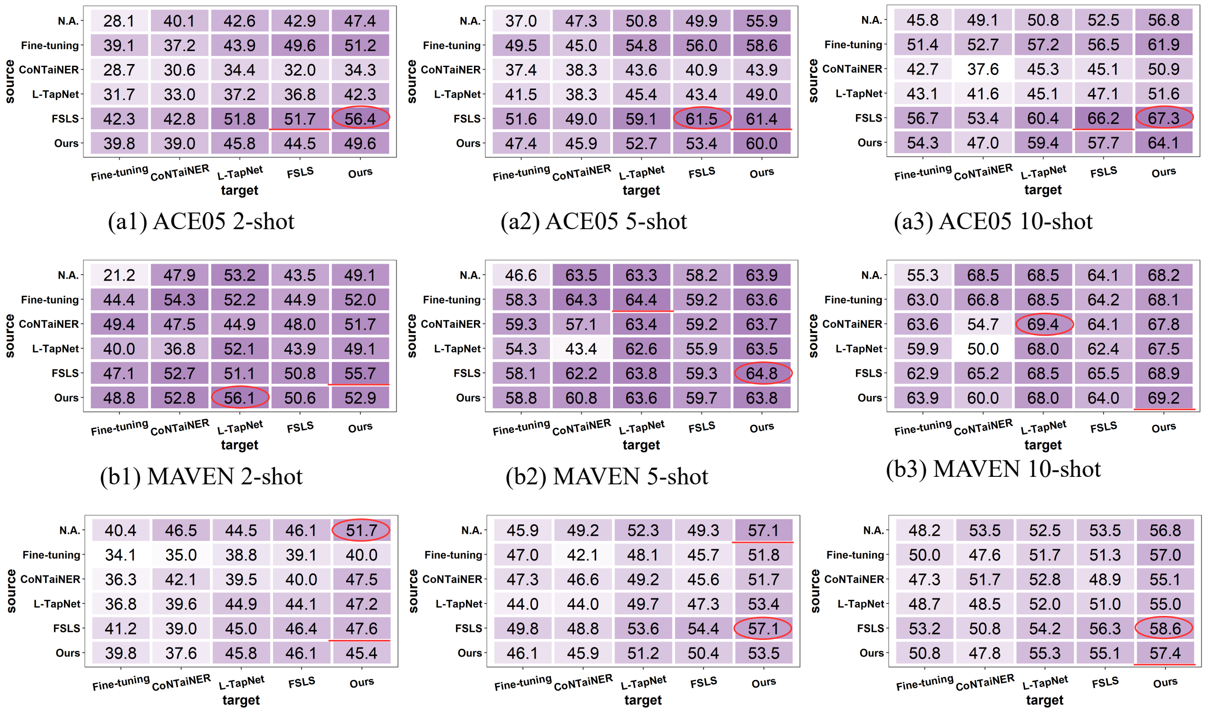
6.2 Prototype-based methods

We further explore the transfer ability of existing prototype-based methods and *unified baseline*[4]. Thanks to the unified view, we conduct a more thorough experiment that enumerates all possible combinations of models used in the source and target domain, to assess if the generalization ability affects transferability. That is, the parameters in PLMs will be shared from source to target model. We show results in Figure 7 and Appendix D.2.

1. Is transfer learning effective for prototype-based methods? It depends on the dataset (compare the first row with other rows in each column). For ACE05 and MAVEN datasets, the overall answer is yes. Contrary to our expectation, transfer learning affects most target models on ERE dataset negatively, especially for 2- and 5-shot settings.

2. Do prototype-based methods perform better than simple fine-tuning? It depends on whether *fine-tuning* the source or target model. When *fine-tuning* a source model (row 2), it sometimes achieves comparable even better performance than the prototype-based methods (last 4 rows). When *fine-tuning* a target model (column 1), however, the performance drops significantly. Thus, we speculate that powerful prototype-based methods are more necessary in target domain than source domain.

[4] Transfer and distance functions in all methods are substituted to appropriate ones and CRF modules are removed.



(a1) ACE05 2-shot (a2) ACE05 5-shot (a3) ACE05 10-shot

(b1) MAVEN 2-shot (b2) MAVEN 5-shot (b3) MAVEN 10-shot

(c1) ERE 2-shot (c2) ERE 5-shot (c3) ERE 10-shot

Figure 7: Class-transfer results of *fine-tuning* methods and four prototype-based methods on three datasets. For each matrix, row and column represent the source and target models, respectively. For example, the value in top-left corners of every matrix means the performance when directly finetuning a model in target dataset (source: N.A. / target: Fine-tuning). Each value is the results averaged over 10 repeated experiments. See full results in Table 12.

3. Is the choice of prototype-based methods important? Yes. When we select inappropriate prototype-based methods, they could achieve worse performance than simple fine-tuning and sometimes even worse than models without class transfer. For example, CONTAINER and L-TapNet are inappropriate source model for ACE05 dataset.

4. Do the same source and target models benefit the event-related knowledge transfer? No. The figures show the best model combinations often deviate from the diagonals. It indicates that different source and target models sometimes achieve better results.

5. Is there a source-target combination performing well on all settings? Strictly speaking, the answer is No. Nevertheless, we find that adopting FSLS as the source model and our *unified baseline* as the target model is more likely to achieve competitive (best or second best) performance among all alternatives. It indicates that (1) the quality of different combinations show kinds of **tendency** though no consistent conclusion could be drawn. (2) a model with moderate inductive bias (like FSLS) might be better for the source dataset with abundant samples. Then our *unified baseline* could play a role during the target stage with limited samples.

7 Conclusion

We have conducted a comprehensive empirical study comparing 12 representative methods under unified *low-resource* and *class-transfer* settings. For systematic analysis, we proposed a unified framework of promising prototype-based methods. Based on it, we presented a simple and effective *baseline* that outperforms all existing methods significantly under *low-resource* setting, and is an ideal choice as the target model under *class-transfer* setting. In the future, we aim to explore how to leverage unlabeled corpus for few-shot ED tasks, such as data augmentation, weakly-supervised learning, and self-training.

Acknowlegement

This study is supported under the RIE2020 Industry Alignment Fund – Industry Collaboration Projects (IAF-ICP) Funding Initiative, the Singapore Ministry of Education (MOE) Academic Research Fund (AcRF) Tier 1 grant, as well as cash and in-kind contribution from the industry partner(s).

Limitations

We compare 12 representative methods, present a *unified view* on existing prototype-based methods, and propose a competitive *unified baseline* by combining the advantageous modules of these methods. We test all methods, including the unified baseline, on three commonly-used English datasets using various experimental settings and achieve consistent results. However we acknowledge the potential disproportionality of our experiments in terms of language, domain, schema type and data scarcity extent. Therefore, for future work, we aim to conduct our empirical studies on more diverse event-detection (ED) datasets.

We are fortunate to witness the rapid development of Large Language Models (LLMs Brown et al. 2020b; Ouyang et al. 2022; Chung et al. 2022) in recent times. In our work, we set in-context learning as a baseline and evaluate the performance of LLMs on few-shot ED tasks. We find current LLMs still face challenges in dealing with Information Extraction (IE) tasks that require structured outputs (Qin et al., 2023; Josifoski et al., 2023). However, we acknowledge the ICL approach adopted here is relatively simple. We do not work hard to find the optimal prompt format, demonstration selection strategy, etc., to reach the upper bounds of LLMs' performance. We view how to leverage the power of LLMs on ED tasks as an open problem and leave it for future work.

In this work, we focus more on the model aspect of few-shot ED tasks rather than data aspect. In other words, we assume having and only having access to a small set of labeled instances. In the future, we plan to explore how to utilize annotation guidelines, unlabeled corpus and external structured knowledge to improve few-shot ED tasks.

References

Ofer Bronstein, Ido Dagan, Qi Li, Heng Ji, and Anette Frank. 2015. Seed-based event trigger labeling: How far can event descriptions get us? In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 372–376, Beijing, China. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020a. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*.

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, and Dario Amodei. 2020b. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA. Curran Associates Inc.

Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. 2021. Honey or poison? solving the trigger curse in few-shot event detection via causal intervention. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8078–8088, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Meiqi Chen, Yixin Cao, Kunquan Deng, Mukai Li, Kun Wang, Jing Shao, and Yan Zhang. 2022. ERGO: Event relational graph transformer for document-level event causality identification. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 2118–2128, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. ICML'20.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models.

Xin Cong, Shiyao Cui, Bowen Yu, Tingwen Liu, Wang Yubin, and Bin Wang. 2021. Few-Shot Event Detection with Prototypical Amortized Conditional Random Field. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 28–40, Online. Association for Computational Linguistics.

Sarkar Snigdha Sarathi Das, Arzoo Katiyar, Rebecca Passonneau, and Rui Zhang. 2022. CONTaiNER: Few-shot named entity recognition via contrastive learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6338–6353, Dublin, Ireland. Association for Computational Linguistics.

Shumin Deng, Ningyu Zhang, Jiaojian Kang, Yichi Zhang, Wei Zhang, and Huajun Chen. 2020. Meta-learning with dynamic-memory-based prototypical network for few-shot event detection. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. ACM.

Shumin Deng, Ningyu Zhang, Luoqiu Li, Chen Hui, Tou Huaixiao, Mosha Chen, Fei Huang, and Huajun Chen. 2021. OntoED: Low-resource event detection with ontology embedding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2828–2839, Online. Association for Computational Linguistics.

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.

Ning Ding, Shengding Hu, Weilin Zhao, Yulin Chen, Zhiyuan Liu, Haitao Zheng, and Maosong Sun. 2022. OpenPrompt: An open-source framework for prompt-learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 105–113, Dublin, Ireland. Association for Computational Linguistics.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The automatic content extraction (ACE) program – tasks, data, and evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal. European Language Resources Association (ELRA).

Xinya Du and Claire Cardie. 2020. Event extraction by answering (almost) natural questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 671–683, Online. Association for Computational Linguistics.

Rui Feng, Jie Yuan, and Chao Zhang. 2020. Probing and fine-tuning reading comprehension models for few-shot event extraction. *CoRR*, abs/2010.11325.

Alexander Fritzler, Varvara Logacheva, and Maksim Kretov. 2019. Few-shot classification in named entity recognition task. New York, NY, USA. Association for Computing Machinery.

Jun Gao, Huan Zhao, Changlong Yu, and Ruifeng Xu. 2023. Exploring the feasibility of chatgpt for event extraction.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

R. Hadsell, S. Chopra, and Y. LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9726–9735.

Yutai Hou, Wanxiang Che, Yongkui Lai, Zhihan Zhou, Yijia Liu, Han Liu, and Ting Liu. 2020. Few-shot slot tagging with collapsed dependency transfer and label-enhanced task-adaptive projection network. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1381–1393, Online. Association for Computational Linguistics.

I-Hung Hsu, Kuan-Hao Huang, Elizabeth Boschee, Scott Miller, Prem Natarajan, Kai-Wei Chang, and Nanyun Peng. 2022. DEGREE: A data-efficient generation-based event extraction model. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1890–1908, Seattle, United States. Association for Computational Linguistics.

Jiaxin Huang, Chunyuan Li, Krishan Subudhi, Damien Jose, Shobana Balakrishnan, Weizhu Chen, Baolin Peng, Jianfeng Gao, and Jiawei Han. 2021. Few-shot named entity recognition: An empirical baseline study. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10408–10423, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Kuan-Hao Huang, I-Hung Hsu, Prem Natarajan, Kai-Wei Chang, and Nanyun Peng. 2022. Multilingual generative language models for zero-shot cross-lingual event argument extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4633–4646, Dublin, Ireland. Association for Computational Linguistics.

Lifu Huang, Heng Ji, Kyunghyun Cho, Ido Dagan, Sebastian Riedel, and Clare Voss. 2018. Zero-shot transfer learning for event extraction. In *Proceedings of the 56th Annual Meeting of the Association for*

Computational Linguistics (Volume 1: Long Papers), pages 2160–2170, Melbourne, Australia. Association for Computational Linguistics.

Martin Josifoski, Marija Sakota, Maxime Peyrard, and Robert West. 2023. Exploiting asymmetry for synthetic training data generation: Synthie and the case of information extraction.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*.

Viet Lai, Franck Dernoncourt, and Thien Huu Nguyen. 2021. Learning prototype representations across few-shot tasks for event detection. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5270–5277, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Viet Dac Lai, Franck Dernoncourt, and Thien Huu Nguyen. 2020a. Exploiting the matching information in the support set for few shot event classification. In *Advances in Knowledge Discovery and Data Mining: 24th Pacific-Asia Conference, PAKDD 2020*.

Viet Dac Lai, Thien Huu Nguyen, and Franck Dernoncourt. 2020b. Extensively matching for few-shot learning event detection. In *Proceedings of the First Joint Workshop on Narrative Understanding, Storylines, and Events*, pages 38–45, Online. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Manling Li, Alireza Zareian, Ying Lin, Xiaoman Pan, Spencer Whitehead, Brian Chen, Bo Wu, Heng Ji, Shih-Fu Chang, Clare Voss, Daniel Napierski, and Marjorie Freedman. 2020. GAIA: A fine-grained multimedia knowledge extraction system. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 77–86, Online. Association for Computational Linguistics.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 73–82, Sofia, Bulgaria. Association for Computational Linguistics.

Qintong Li, Piji Li, Wei Bi, Zhaochun Ren, Yuxuan Lai, and Lingpeng Kong. 2022a. Event transition planning for open-ended text generation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3412–3426, Dublin, Ireland. Association for Computational Linguistics.

Sha Li, Liyuan Liu, Yiqing Xie, Heng Ji, and Jiawei Han. 2022b. Piled: An identify-and-localize framework for few-shot event detection.

Ying Lin, Heng Ji, Fei Huang, and Lingfei Wu. 2020. A joint neural model for information extraction with global features. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7999–8009, Online. Association for Computational Linguistics.

Jian Liu, Yubo Chen, Kang Liu, Wei Bi, and Xiaojiang Liu. 2020. Event extraction as machine reading comprehension. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1641–1651, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Yaojie Lu, Hongyu Lin, Jin Xu, Xianpei Han, Jialong Tang, Annan Li, Le Sun, Meng Liao, and Shaoyi Chen. 2021. Text2Event: Controllable sequence-to-structure generation for end-to-end event extraction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2795–2806, Online. Association for Computational Linguistics.

Yaojie Lu, Qing Liu, Dai Dai, Xinyan Xiao, Hongyu Lin, Xianpei Han, Le Sun, and Hua Wu. 2022. Unified structure generation for universal information extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5755–5772, Dublin, Ireland. Association for Computational Linguistics.

Qing Lyu, Hongming Zhang, Elior Sulem, and Dan Roth. 2021. Zero-shot event extraction via transfer learning: Challenges and insights. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 322–332, Online. Association for Computational Linguistics.

Jie Ma, Miguel Ballesteros, Srikanth Doss, Rishita Anubhai, Sunil Mallya, Yaser Al-Onaizan, and Dan Roth. 2022a. Label semantics for few shot named entity recognition. In *Findings of the Association for*

Computational Linguistics: ACL 2022, pages 1956–1971, Dublin, Ireland. Association for Computational Linguistics.

Yubo Ma, Yixin Cao, YongChing Hong, and Aixin Sun. 2023. Large language model is not a good few-shot information extractor, but a good reranker for hard samples!

Yubo Ma, Zehao Wang, Yixin Cao, Mukai Li, Meiqi Chen, Kun Wang, and Jing Shao. 2022b. Prompt for extraction? PAIE: Prompting argument interaction for event argument extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6759–6774, Dublin, Ireland. Association for Computational Linguistics.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback.

Giovanni Paolini, Ben Athiwaratkun, Jason Krone, Jie Ma, Alessandro Achille, Rishita Anubhai, Cicero Nogueira dos Santos, Bing Xiang, and Stefano Soatto. 2021. Structured prediction as translation between augmented natural languages.

Haoruo Peng, Yangqiu Song, and Dan Roth. 2016. Event detection and co-reference with minimal supervision. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 392–402, Austin, Texas. Association for Computational Linguistics.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. True few-shot learning with language models. In *Advances in Neural Information Processing Systems*, volume 34, pages 11054–11070.

Chengwei Qin, Aston Zhang, Zhuosheng Zhang, Jiaao Chen, Michihiro Yasunaga, and Diyi Yang. 2023. Is chatgpt a general-purpose natural language processing task solver?

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352, Online. Association for Computational Linguistics.

Shirong Shen, Tongtong Wu, Guilin Qi, Yuan-Fang Li, Gholamreza Haffari, and Sheng Bi. 2021. Adaptive knowledge-enhanced Bayesian meta-learning for few-shot event detection. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2417–2429, Online. Association for Computational Linguistics.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 4080–4090, Red Hook, NY, USA.

Zhiyi Song, Ann Bies, Stephanie Strassel, Tom Riese, Justin Mott, Joe Ellis, Jonathan Wright, Seth Kulick, Neville Ryant, and Xiaoyi Ma. 2015. From light to rich ERE: Annotation of entities, relations, and events. In *Proceedings of the The 3rd Workshop on EVENTS: Definition, Detection, Coreference, and Representation*, pages 89–98, Denver, Colorado. Association for Computational Linguistics.

Xiaozhi Wang, Ziqi Wang, Xu Han, Wangyi Jiang, Rong Han, Zhiyuan Liu, Juanzi Li, Peng Li, Yankai Lin, and Jie Zhou. 2020. MAVEN: A Massive General Domain Event Detection Dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1652–1671, Online. Association for Computational Linguistics.

Haoyang Wen, Ying Lin, Tuan Lai, Xiaoman Pan, Sha Li, Xudong Lin, Ben Zhou, Manling Li, Haoyu Wang, Hongming Zhang, Xiaodong Yu, Alexander Dong, Zhenhailong Wang, Yi Fung, Piyush Mishra, Qing Lyu, Dídac Surís, Brian Chen, Susan Windisch Brown, Martha Palmer, Chris Callison-Burch, Carl Vondrick, Jiawei Han, Dan Roth, Shih-Fu Chang, and Heng Ji. 2021. RESIN: A dockerized schema-guided cross-document cross-lingual cross-media information extraction and event tracking system. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstrations*, pages 133–143, Online. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Yi Yang and Arzoo Katiyar. 2020. Simple and effective few-shot named entity recognition with structured nearest neighbor learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6365–6375, Online. Association for Computational Linguistics.

Sung Whan Yoon, Jun Seo, and Jaekyun Moon. 2019. TapNet: Neural network augmented with task-adaptive projection for few-shot learning. In *Proceedings of the 36th International Conference on Machine Learning*.

Pengfei Yu, Zixuan Zhang, Clare Voss, Jonathan May, and Heng Ji. 2022. Building an event extractor with only a few examples. In *Proceedings of the Third Workshop on Deep Learning for Low-Resource Natural Language Processing*, pages 102–109, Hybrid. Association for Computational Linguistics.

Qiusi Zhan, Sha Li, Kathryn Conger, Martha Palmer, Heng Ji, and Jiawei Han. 2023. Glen: General-purpose event detection for thousands of types.

Hongming Zhang, Haoyu Wang, and Dan Roth. 2021. Zero-shot Label-aware Event Trigger and Argument Classification. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1331–1340, Online. Association for Computational Linguistics.

Hongming Zhang, Wenlin Yao, and Dong Yu. 2022a. Efficient zero-shot event extraction with context-definition alignment. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 7169–7179, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Ruihan Zhang, Wei Wei, Xian-Ling Mao, Rui Fang, and Dangyang Chen. 2022b. HCL-TAT: A hybrid contrastive learning method for few-shot event detection with task-adaptive threshold. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1808–1819, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Kailin Zhao, Xiaolong Jin, Long Bai, Jiafeng Guo, and Xueqi Cheng. 2022. Knowledge-enhanced self-supervised prototypical network for few-shot event detection. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6266–6275, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

A Related Work

A.1 Taxonomy of task settings

Various solutions have been proposed to improve the *generalization* and *transfer* abilities of few-shot ED methods. There exists a bottleneck: the models adopt very different tasks and experimental settings. We categorize existing task settings to four cases as shown in Figure 8: *low-resource* (LR), *class transfer* (CL), *episode learning* (EL), and *task transfer* (TT) settings. LR is used to evaluate the *generalization* ability, learning rapidly with only few examples in target domain. The other settings (CL, EL, and TT) evaluate the *transfer* ability, adapting a model trained with a preexisting schema with abundant samples, to a new (target) schema with only few examples. Based on the pros and cons presented here, we adopt the *low-resource* and *class transfer* settings in our empirical study.

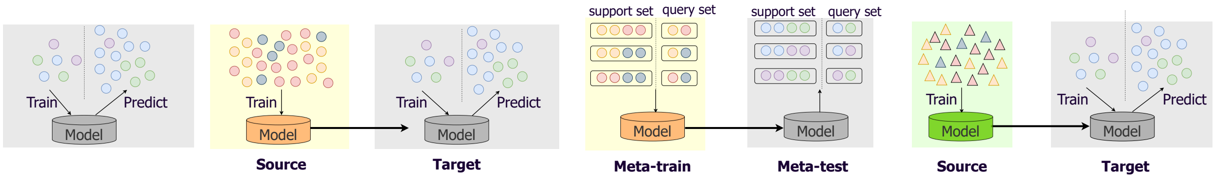
1. Low-resource setting assesses the generalization ability of models by (1) utilizing only few samples during training, (2) evaluating on the real and rich test dataset. Conventionally, the few-shot $|\mathcal{D}_{train}|$ and $|\mathcal{D}_{dev}|$ are downsampled from a full dataset by two main strategies: (1) *K-shot sampling* which picks out K samples for each event type, or (2) *ratio sampling* which picks out partial sentences with a fixed ratio. We view both sampling strategies as reasonable and adopt K-shot sampling in this work.

The surging development of PLMs makes training with only few (or even zero) examples possible, and achieves acceptable performance (Devlin et al., 2019; Raffel et al., 2020; Brown et al., 2020a). Accordingly, a series of prompt-based methods (Du and Cardie, 2020; Liu et al., 2020; Feng et al., 2020; Paolini et al., 2021; Lu et al., 2021; Deng et al., 2021; Hsu et al., 2022; Li et al., 2022b) adopt such setting to train and evaluate their models.

2. Class transfer setting assesses the *transferability* of a model by providing abundant samples in the source (preexisting) schema and scarce samples in target (new) schema. It trains a classifier in source schema and then transfers such classifier to the target schema with only few examples.

Such setting has been applied since an early stage (Bronstein et al., 2015; Peng et al., 2016; Zhang et al., 2021), and is often used together with low-resource setting to additionally evaluate transferability of the models (Paolini et al., 2021; Lu et al., 2021; Hsu et al., 2022).

3. Episode learning setting is a classical few-shot setting. It has two phases, *meta-training* and *meta-testing*, each of which consists of multiple episodes. Each episode is a few-shot problem with its own train (support) and test (query) sets and event-type classes. Since the sets in each episode are sampled uniformly having K different classes and each class having N instances, episode learning is also known



(a) low-resource setting (b) class-transfer setting (c) episode learning setting (d) task-transfer setting

Figure 8: Four few-shot settings summarized from previous work. Different colors represent different event types. Different shapes represent samples with different tasks.

as N**-way-**K**-shot** classification.

Many existing few-shot ED methods adopt this setting (Lai et al., 2020a,b; Deng et al., 2020; Cong et al., 2021; Lai et al., 2021; Chen et al., 2021; Zhang et al., 2022b; Zhao et al., 2022). However, we argue that episode learning assumes an unrealistic scenario. First, during the meta-training stage, a large number of episodes is needed, for example, 20,000 in Cong et al. (2021). Though the label sets of meta-training and meta-testing stages are disjoint, class transfer setting is more reasonable when there are many samples in another schema. Second, tasks with episode learning are evaluated by the performance on samples of the test (query) set in the meta-testing phase. The test sets are sampled uniformly, leading to a significant discrepancy with the true data distribution in many NLP tasks. The absence of sentences without any events further leads to distribution distortion. Further, each episode contains samples with only K different classes, where K is usually much smaller than the event types in the target schema. All these factors may lead to an overestimation on the ability of few-shot learning systems. For above reasons, we do not consider this setting in our experiments.

4. Task transfer setting is very similar to class transfer. The main difference is that it relaxes the constraint in source phase, from the same task with different schema to different tasks.[5] The development of this setting also heavily relies on the success of PLMs. Liu et al. (2020), Feng et al. (2020) and Lyu et al. (2021) leverage model pre-trained with SQuAD 2.0 (QA dataset, Rajpurkar et al. 2018) or MNLI (NLI dataset, Williams et al. 2018) to improve the performance of zero-/few-shot ED models. Paolini et al. (2021) and Lu et al. (2022) recently construct unified generation frameworks on multiple IE tasks. Their experiments also reveal that pre-training on these tasks benefits few-shot ED. Though task transfer setting is reasonable and promising, we do not include this setting out of its extreme diversity and complexity. That is, there are (1) too many candidate tasks as pre-training tasks, and (2) too many optional datasets for each pre-training task. Thus it is almost infeasible to conduct a comprehensive empirical study on task transfer setting.

A.2 Taxonomy of methods

We categorize existing methods to two main classes, **prompt-based** methods and **prototype-based** methods, and list them in Table 1. Here we give a detailed introduction of existing methods. Note that in our empirical study, we also include some methods which are originally developed for similar few-shot tasks but can be easily adapted to ED. We leave a special subsection for them.

Few-shot ED methods. Due to the prohibitively cost for labeling amounts of event mentions, few-shot ED is a long-standing topic in event-related research community. The proposed solutions are mainly in two branches. The first branch, *prototype-based* [6] methods, is a classical approach on few-shot learning. It defines a single or multiple *prototypes* for each event type representing the label-wise properties. It then learns the embedding representation of each sample via shortening the distance from its corresponding prototypes given a distance/similarity metric. Bronstein et al. (2015) and Peng et al. (2016) leverage the seed instances in annotation guideline and mine the lexical/semantic features of trigger words to obtain the prototypes. Zhang et al. (2021) inherit such paradigm and define prototypes as the average contextualized embeddings of the related trig-

[5] Generally speaking, all methods using PLMs belong to this setting in which the source task is exactly the pre-training task of PLMs, masked- or next-word prediction. In this work, we limit the discussion of task transfer to which the source task is another downstream task rather than the general pre-training task in PLMs.

[6] Different from other sections, here we adopt a chronological order and firstly introduce prototype-based methods.

ger words weakly labeled in external corpus. With the help AMR Parsing, Huang et al. (2018) additionally consider the graph structures of preexisting schema as prototypes, and encode AMR graph representation of each event mention as representations. Deng et al. (2020) introduces Dynamic Memory Network (DMN), while Lai et al. (2020a) and Lai et al. (2021) introduce two different auxiliary losses improving intra-/inter-consistency of different episodes to facilitate their prototype representations. Deng et al. (2021) further consider the relations among events to constrain the prototypes and benefit both rare and new events. Cong et al. (2021) amortize CRF module by modeling the transition probabilities of different event types with their prototypes. Chen et al. (2021) leverage causal inference and intervene on context via backdoor adjustment during training to reduce overfitting of trigger words for more robust prototypes. Recently, Zhang et al. (2022a) and Zhang et al. (2022b) introduce contrastive learning into few-shot ED task and their proposed methods actually could be viewed as *generalized* prototype-based methods with *multiple* prototypes rather than one.

The other branch, *prompting methods*, is made possible with the surge of development in PLMs. Given a specific task, prompting methods map the task format to a new format with which the PLMs are more familiar, such as masked word prediction (Schick and Schütze, 2021) and sequence generation (Raffel et al., 2020; Brown et al., 2020a). Such format conversion narrows down the gaps between pre-training tasks and downstream tasks, which is beneficial for inducing learned knowledge from PLMs with limited annotations. As for event detection (and many other IE tasks), however, it is not trivial to design a smooth format conversion. One simple idea is leveraging one single template to prompt both event types and their triggers simultaneously (Paolini et al., 2021; Lu et al., 2021). However, such prompting methods show performance far from satisfactory, especially when they are not enhanced by two-stage pre-training and redundant hinting prefix (Lu et al., 2022). Another natural idea is enumerating all legal spans and querying the PLMs whether each span belongs to any class, or vice versa (Hsu et al., 2022). A major limitation here is the prohibitively time complexity, particularly when there are many event types. Combining the merits of *prompting methods* and conventional *fine-tuning methods* is another solution. Du and Cardie (2020) and Liu et al. (2020) use QA/MRC format to prompt the location of trigger words, while still predicting their event types via an additional linear head. Lyu et al. (2021) first segment one sentence into several clauses and view the predicates of clauses as trigger candidates. Then they leverage NLI format to query the event types of these candidates. Recently, Li et al. (2022b) propose a strategy combining Pattern-Exploiting Training (PET, Schick and Schütze 2021) and CRF module. Initially, they conduct sentence-level event detection determining whether one sentence contains any event types or not. For each identified event type, they further use a linear chain CRF to locate the trigger word.

Few-shot NER/ST methods. There are several models which are originally designed for similar tasks like Named Entity Recognition (NER) and Slot Tagging (ST) but could be applied to ED task.

Similar to ED methods, one classical paradigm in NER is utilizing ProtoNet (Snell et al., 2017) and its variants to learn *one* representative prototypes for each class type with only few examples. Fritzler et al. (2019) firstly combine ProtoNet and CRF module to solve NER tasks. Hou et al. (2020) propose L-TapNet-CDT, which enhances TapNet (Yoon et al., 2019), a variant of ProtoNet, with textual label names and achieves great performance among several ST tasks. Both methods construct prototypes by computing the average embeddings of several sampled examples (support set). Yang and Katiyar (2020) propose a simpler algorithm, leveraging supervised classifier learned in preexisting schema as feature extractor and adopting nearest neighbors classification during inference, and show competitive performance in class transfer setting for few-shot NER task. Das et al. (2022) introduce contrastive learning into few-shot NER task. Ma et al. (2022a) recently developed a simple but effective method on few-shot NER by constructing prototypes only with their labels.

B Datasets and Models

We curate few-shot datasets used in this emprical study from three full and commonly-used datasets: ACE05 (Doddington et al., 2004), MAVEN (Wang et al., 2020) and ERE (Song et al., 2015).

B.1 Full dataset

ACE05 is a joint information extraction dataset, with annotations of entities, relations, and events.

Table 4: Statistics of three full ED datasets.

Dataset		ACE05	MAVEN	ERE
#Event type		33	168	38
#Sents	Train	14,024	32,360	14,736
	Test	728	8,035	1,163
#Mentions	Train	5,349	77,993	6,208
	Test	424	18,904	551

We only use its event annotation for ED task. It contains 599 English documents and 33 event types in total. We split documents in ACE05 following previous work (Li et al., 2013) to construct train and test dataset respectively. MAVEN is a newly-built large-scale ED dataset with 4480 documents and 168 event types. We use the official split for MAVEN dataset. ERE is another joint information extraction dataset having a similar scale as ACE05 (458 documents, 38 event types). We follow the preprocessing procedure in Lin et al. (2020). Table 4 reports detailed statistics of the three datasets.

ED could be viewed as either a span classification or a sequence labeling task. In our work, we adopt span classification paradigm for MAVEN dataset since it provides official spans for candidate triggers (including negative samples). For the other two datasets, we follow sequence labeling paradigm to predict the event type word by word.

B.2 Dataset construction

This section introduces how we construct few-shot datasets from the three full ED datasets.

Low-resource setting. We downsample sentences from original full training dataset to construct $\mathcal{D}_{train}$ and $\mathcal{D}_{dev}$, and inherit the original test set as the unified $\mathcal{D}_{test}$. For $\mathcal{D}_{train}$ and $\mathcal{D}_{dev}$, we adopt K-shot sampling strategy that each event type has (at least) K samples. Since our sampling is at sentence-level and each sentence could have multiple events, the sampling is NP-complete[7] and unlikely to find a practical solution satisfying exactly K samples for each event type. Therefore, we follow Yang and Katiyar (2020) and Ma et al. (2022a) and adopt a greedy sampling algorithm to select sentences, as shown in Alg. 1. Note that the actual sample number of each event type can be larger than K under this sampling strategy. The statistics of the curated datasets are listed in Table 5 (top).

Class-Transfer setting This setting has a more

[7] The *Subset Sum Problem*, a classical NP-complete problem, can be reduced to this sampling problem.

Algorithm 1 Greedy Sampling

```
Require: shot number $K$, original full dataset $\mathcal{D} = \{(\mathbf{X}, \mathbf{Y})\}$ tagged with label set $E$
Sort $E$ based on their frequencies in $\{\mathbf{Y}\}$ as an ascending order
$S \leftarrow \phi$, Counter $\leftarrow$ dict()
for $y \in E$ do
    Counter($y$) $\leftarrow 0$
end for
for $y \in E$ do
    while Counter($y$) $< K$ do
        Sample $(\mathbf{X}, \mathbf{Y}) \in \mathcal{D}$ s.t.$\exists j, y_j = y$
        $\mathcal{D} \leftarrow \mathcal{D} \backslash (\mathbf{X}, \mathbf{Y})$
        Update Counter (not only $y$ but all event types in $\mathbf{Y}$)
    end while
end for
for $s \in \mathcal{S}$ do
    $\mathcal{S} \leftarrow \mathcal{S} \backslash s$ and update Counter
    if $\exists y \in E$, s.t. Counter($y$) $< K$ then
        $\mathcal{S} \leftarrow \mathcal{S} \bigcup s$
    end if
end for
return $\mathcal{S}$
```

Table 5: The statistics of curated datasets for few-shot ED tasks. Top: Low-resource setting. Bottom: Class transfer setting. We set different random seeds and generate 10 few-shot sets for each setting. We report their average statistics.

Low-resource		# Labels	# Sent	# Event	# Avg shot
ACE05	2-shot	33	47.7	76.4	2.32
	5-shot		110.7	172.2	5.22
	10-shot		211.5	317.5	9.62
MAVEN	2-shot	168	152.6	530.1	3.16
	5-shot		359.6	1226.3	7.30
	10-shot		705.1	2329.2	13.86
ERE	2-shot	38	43.6	108.9	2.87
	5-shot		102.5	249.9	6.58
	10-shot		197.1	472.3	12.43
Class-transfer		# Labels	# Sent	# Event	# Avg shot
ACE05	2-shot	23	37.1	50.2	2.18
	5-shot		84.6	113.0	4.91
	10-shot		159.8	209.9	9.13
MAVEN	2-shot	48	84.3	97.4	2.03
	5-shot		211.3	236.6	4.93
	10-shot		417.3	453.6	9.45
ERE	2-shot	28	39.7	66.1	2.36
	5-shot		95.0	153.5	5.48
	10-shot		182.5	291.0	10.39

complicated curation process, and roughly consists of two sub-steps: (1) Dividing both event types and sentences in the original dataset into two disjoint parts named source dataset and target dataset pool. (2) Using the entire source dataset, and selecting few-shot samples from the target pool to construct target set.

For step (1), we follow Huang et al. (2018) and Chen et al. (2021) to pick out the most frequent 10, 120, and 10 event types from ACE05, MAVEN and ERE dataset respectively, as $E^{(S)}$. The remaining types are $E^{(T)}$. Then we take sentences containing any annotations in $E^{(T)}$ to $D^{(T)}_{full}$ for enriching the sampling pool of target dataset as much as possible,

$$D^{(T)}_{full} = \{(\boldsymbol{X}, R(\boldsymbol{Y}; E^{(S)}))|(\boldsymbol{X}, \boldsymbol{Y}) \in D, \exists y_j \in E^{(T)}\}$$

where $R(\boldsymbol{Y}; E^{(S)}$ represents the relabeling operation that substituting any $y_j \in E^{(S)})$ to `N.A.` to avoid information leakage. The remaining sentences are collected as $D^{(S)}$.

$$D^{(S)} = \{(\boldsymbol{X}, R(\boldsymbol{Y}; E^{(T)}))|(\boldsymbol{X}, \boldsymbol{Y}) \notin D^{(T)}_{full}\}$$

For step (2), we adopt the same strategy as low-resource setting to sample K-shot $D^{(T)}_{train}$ and $D^{(T)}_{dev}$ from target sampling pool $D^{(T)}_{full}$. Statistics of curated datasets are summarized in Table 5 (bottom).

B.3 Existing methods

We conduct our empirical study on twelve representative existing methods. Besides vanilla fine-tuning and in-context learning, five of them are prompt-based and the other five are prototype-based.

1. Prompt-based methods leverage the rich knowledge in PLMs by converting specific downstream tasks to the formats that PLMs are more familiar with. We give examples about prompt format of the five prompt-based methods in Table 6.

EEQA/EERC (Du and Cardie, 2020; Liu et al., 2020): a QA/MRC-based method which first extracts the trigger word with a natural language query then classifies its type with an additional classifier.

EDTE (Lyu et al., 2021): a NLI-based method which enumerates all event types and judges whether a clause is entailed by any event. The clause is obtained by SRL processing and the trigger candidate is the predicate of each clause.

PTE (Schick and Schütze, 2021): a cloze-style prompt method which enumerates each word in the sentence and predicts whether it is the trigger of any event type.

UIE (Lu et al., 2022): a generation based method that takes in a sentence and outputs a filled *universal* template, indicating the trigger words and their event types in the sentence.

DEGREE (Hsu et al., 2022): also adopts a generation paradigm but it enumerates all event types by designing *type-specific* template, and outputs related triggers (if have).

2. Prototype-based methods predict an event type for each word or span by measuring the representation proximity between the samples and the *prototypes* for each event type.

Prototypical Network (Snell et al., 2017): a classical prototype-based method originally developed for episode learning. Huang et al. (2021) adapt it to low-resource setting via further splitting the training set into support set $\mathcal{S}_y$ and query set $\mathcal{Q}_y$. The prototype $\bar{c}_y$ of each event type is constructed by averaged PLM representations of samples in $\mathcal{S}_y$.

$$h_{\bar{c}_y} = \frac{1}{\mathcal{S}_y} \sum_{s \in \mathcal{S}_y} h_s$$

For samples x in $\mathcal{Q}_y$ during training, or in the test set during inference, logits$(y|x)$ is defined as the negative euclidean distance between $h(x)$ and $\bar{c}_y$.

$$\text{logits}(y|x) = -||h_x - h_{\bar{c}_y}||_2$$

L-TapNet-CDT (Hou et al., 2020): a ProtoNet-variant method with three main improvements: (1) it introduces TapNet, a variant of ProtoNet. TapNet's main difference from ProtoNet lies in a projection space $\mathcal{M}$ analytically constructed. The distance is computed in the subspace spanned by $\mathcal{M}$.

$$\text{logits}(y|x) = -||\mathcal{M}(h_x - h_{\bar{c}_y})||_2$$

(2) the basis in column space of $\mathcal{M}^{\perp}$ is aligned with label semantic, thus $\mathcal{M}(E)$ is label-enhanced. (3) a collapsed dependency transfer (CDT) module is used solely during inference stage to scale the event-type score.

$$\text{logits}(y|x) \leftarrow \text{logits}(y|x) + \text{TRANS}(y)$$

PA-CRF (Cong et al., 2021): a ProtoNet-variant method with a CRF module as well. Different from CDT, however, the transition scores are approximated between event types based on the their prototypes and learned during training.

Table 6: Prompt examples for different methods based on a sentence example `X`: *The current government was formed in October 2000*, in which the word *formed* triggering an *Start-Org* event. The underline part in UIE prompt is their designed Structured Schema Instructor (SSI), and the *DESCRIPTION*(y) in DEGREE prompt is a description about event type $y \in E$ written in natural languages. We refer readers for their original paper in details.

Method	Prompt Input	Output
EEQA (Du and Cardie, 2020)	`X`. What is the trigger in the event?	formed.
EDTE (Lyu et al., 2021)	Premise: `X`. Hypothesis: This text is about a Start-Org event. ... Premise: `X`. Hypothesis: This text is about an Attack event.	Yes. ... No.
PTE (Schick and Schütze, 2021)	`X`. The word *formed* triggers a/an `[MASK]` event. ... `X`. The word *current* triggers a/an `[MASK]` event.	Start-Org ... N.A.
UIE (Lu et al., 2022)	<spot> Start-org <spot> Attack <spot> ... <spot>. `X`.	(Start-Org: formed)
DEGREE (Hsu et al., 2022)	`X`. *DESCRIPTION*(Start-Org). Event trigger is `[MASK]`. ... `X`. *DESCRIPTION*(Attack). Event trigger is `[MASK]`.	Event trigger is formed ... Event trigger is N.A.

FSLS (Ma et al., 2022a): a recently proposed few-shot NER method that generalizes well to ED task. The prototype of each event type is not constructed from support set $\mathcal{S}_y$ but from the label semantic, i.e. the PLM representation of the label name.

$$e_y = \text{Event\_name}(y)$$
$$\text{logits}(y|x) = h_x^T h_{e_y}$$

CONTAINER (Das et al., 2022): a contrastive learning approach. We view it as a *generalized* Prototype-based method since both of their motivations are to pull together the representations of samples with same event types. Different from ProtoNet, there is no explicit division between support set and query set during training process. Instead each sample acts as query and other samples as support samples. For example, given sample x with event type e, its *special* supported set can be viewed as:

$$\mathcal{S}_y(x) = \{x'|(x', y') \in D, y' = y, x' \neq x\}$$

Then its score related to e is calculated as the average distance with samples in $\mathcal{S}_y(x)$.

$$\text{logits}(y|x) = \sum_{x' \in \mathcal{S}_y(x)} \frac{-d(f(h_x), f(h_{x'}))}{|\mathcal{S}_y(x)|}$$

B.4 Implementation Details

For all methods, we initialize their pre-trained weights and further train them using Huggingface library.[8] Each experiment is run on single NVIDIA-V100 GPU, and the final reported performance for each setting (e.g., ACE 2-shot) is the averaged result w.r.t ten distinct few-shot training datasets which are sampled with different random seeds. We further detail the implementation of all methods.

1. In-context learning

Instruction

User Assume you are an event-instance annotator. Given a sentence, you need to (1) identify the word or phrase triggering the event in the sentence, and (2) classify its event type. The possible event types are listed as below: Transport, Attack, Injure, ...

ChatGPT Yes, I understand.

User Please note that your annotation results must follow such format: "Answer: ([Event_1], [Trigger_1]) <SEP> ([Event_2], [Trigger_2]) <SEP> ...". If you do not find any event in this sentence, just output "Answer: No events found."

ChatGPT No problem. Let's start!

Demonstration

User Sentence: He 's retired with the US Army.

ChatGPT Answer: (End-Position, retired)

User Sentence: I wouldn't be surprised if this Arroyo character (who was killed by police as he tried to flee) had a serious criminal past but has been allowed to walk free due to our lax judicial process.

ChatGPT Answer: (Die, killed) <SEP> (Acquit, walk free)

User Sentence: Granting a local sheik power to set up an administrative committee representing the groups in the region.

ChatGPT Answer: No events found.

Question

User Hariri submitted his resignation during a 10-minute meeting with the head of state at the Baabda presidential palace, outside the capital.

We use ChatGPT (`gpt-3.5-turbo-0301`) provided by OpenAI APIs [9] for in-context learning.

[8] https://huggingface.co/

[9] https://platform.openai.com/docs/api-reference

The prompt simulates and records the chatting history between the **user** and the **model**. We show one example as above. The prompt consists of three parts: (1) the instruction telling LLMs the task purposes and input-output formats, (2) the demonstration showcasing several input-output pairs to teach LLMs the task and (3) the input of test instance. We feed the prompt into LLMs and expect them to generate extracted answers. Specifically, we set the temperature as 0 and maximum output token as 128. We make all samples in few-shot train set as demonstration samples if their total length is smaller than the maximum input token length (4096). Otherwise we retrieve similar demonstration samples for each test instance to fill up the input prompt. The similarity between two instances are measured from their embeddings (Gao et al., 2021). For MAVEN dataset, we further sample a test subset, with 1000 instances, from the original one for our evaluation.

2. Prompt-based methods We keep all other hyperparameters the same as in their original papers, except learning rates and epochs. We grid-search best learning rates in [1e-5, 2e-5, 5e-5, 1e-4] for each setting. As for epochs, we find the range of appropriate epochsis highly affected by the prompt format. Therefore we search for epochs method by method without a unified range.

EEQA (Du and Cardie, 2020): We use their original code[10] and train it on our datasets.

EDTE (Lyu et al., 2021): We use their original code[11] and train it on our datasets.

PTE (Schick and Schütze, 2021): We implement this method on OpenPrompt (Ding et al., 2022).

UIE (Lu et al., 2022): We use their original code[12] and train it on our datasets.

DEGREE (Hsu et al., 2022): We reproduce this method based on their original code[13] and train it on our datasets. And we drop event keywords not occurring in few-shot training dataset from prompt to avoid information leakage.

3. Prototype-base methods We build a codebase based on the unified view. We then implement these methods directly on the unified framework, by having different choices for each design element. To ensure the correctness of our codebase, we also compare between results obtained from our implementation and original code for each method, and find they achieving similar performance on few-shot ED datasets.

For all methods (including *unified baseline*), we train them with the AdamW (Loshchilov and Hutter, 2017) optimizer with linear scheduler and 0.1 warmup step. We set weight-decay coefficient as 1e-5 and maximum gradient norms as 1.0. We add a 128-long window centering on the trigger words and only encode the words within the window; in other words, the maximum encoding sequence length is 128. The batch size is set as 128, and training steps as 200 if the transfer function is scaled (see Section 5.2) otherwise 500. We grid-search best learning rates in [1e-5, 2e-5, 5e-5, 1e-4] for each setting. For ProtoNet and its variants, we further split the sentences into support set and query set. The number in support set K_S and query set K_Q are (1, 1) for 2-shot settings, (2, 3) for 5-shot settings. The split strategy is (2, 8) for 10-shot dataset constructed from MAVEN and (5, 5) for others. For methods adopting MoCo-CL setting (also see Section 5.2), we maintain a queue storing sample representations with length 2048 for ACE/ERE 2-shot settings and 8192 for others. For methods adopting CRF, we follow default hyperparameters about CRF in their original papers. For methods adopting scaled transfer functions, we grid search the scaled coefficient τ in [0.1, 0.2, 0.3].

C Low-resource Setting-Extended

C.1 Transfer function and Distance function

We consider several combinations about distance and transfer functions listed in Table 7. We choose cosine similarity (S), negative euclidean distance (EU) and their scaled version (SS/SEU) as distance functions. And we pick out identify (I), down-projection (D) and their normalization version (N/DN) as transfer function. We additionally consider the KL-reparameterization combination (KL-R) used in CONTAINER.

We conduct experiments with four existing prototype-based methods[14] by only changing their transfer and distance functions. We illustrate their results on ACE dataset in Figure 9. (1) From comparison about performance in ProtoNet and TapNet, we find TapNet, i.e., the down-projection transfer, shows no significant improvement on few-shot ED tasks. (2) A scaled coefficient in distance function

[10] https://github.com/xinyadu/eeqa

[11] https://github.com/veronica320/Zeroshot-Event-Extraction

[12] https://github.com/universal-ie/UIE

[13] https://github.com/PlusLabNLP/DEGREE

[14] We *degrade* L-TapNet-CDT to TapNet, and do not include PA-CRF here, because CRF and label-enhancement are not the factors considered in this subsection.

Table 7: Variants on distance function $d(u, v)$ (top) and transfer function $f(h)$ (bottom).

Distance function	$d(u, v)$
Cosine similarity (S)	$u^T v$
Scaled cosine similarity (SS)	$u^T v/\tau$
JS Divergence (KL)	JSD$(u\|\|v)$
Euclidean distance (EU)	$-\|\|u - v\|\|_2$
Scaled euclidean distance (SEU)	$-\|\|u - v\|\|_2/\tau$
Transfer function	$f(h)$
Identify (I)	h
Down-projection (D)	$\mathcal{M}h$
Reparameterization (R)	$\mathcal{N}(\mu(h), \Sigma(h))$
Normalization (N)	$h/\|\|h\|\|$
Down-projection + Normalization (DN)	$\mathcal{M}h/\|\|h\|\|$

achieves strong performance with normalization transfer function, while the performance collapses (failing to converge) without normalization. (3) For ProtoNet and TapNet, scaled euclidean distance (SEU) is a better choice for distance function, while other methods prefer scaled cosine similarity (SS). Based on the findings above, we substitute d and f to the most appropriate for all existing methods and observe a significant improvement on all three datasets, as shown in Table 8.

C.2 CRF module

We explore whether CRF improves the performance of few-shot ED task. Based on *Ll-MoCo* model we developed in Section 5.2, we conduct experiment with three different CRF variants, CDT (CRF inference Hou et al. 2020), vanilla CRF (Lafferty et al., 2001) and PA-CRF (Cong et al., 2021), on ACE05 and MAVEN datasets. Their results are in Figure 10. It shows different CRF variants achieve similar result compared with model without CRF, while a trained CRF (and its prototype-enhanced variant) slightly benefits multiple-word triggers when the sample is extremely scarce (see ACE05 2-shot). These results are inconsistent with other similar sequence labeling tasks such as NER or slot tagging, in which CRF usually significantly improves model performance. We speculate it is due to that the pattern of triggers in ED task is relatively simple. To validate such assumption, we count all triggers in ACE05 and MAVEN datasets. We find that above 96% of triggers are single words, and most of the remaining triggers are verb phrases Thus the explicit modeling of transfer dependency among different event types is somewhat not very meaningful under few-shot ED task. Hence, we drop CRF module in the *unified baseline*.

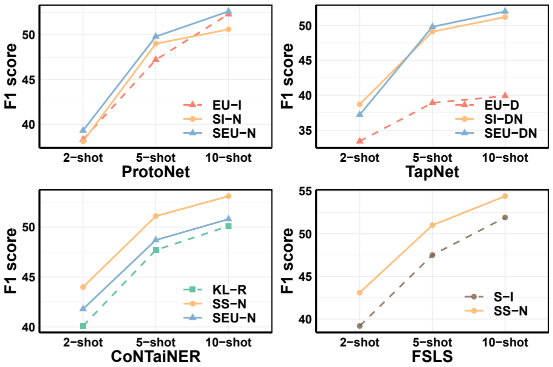
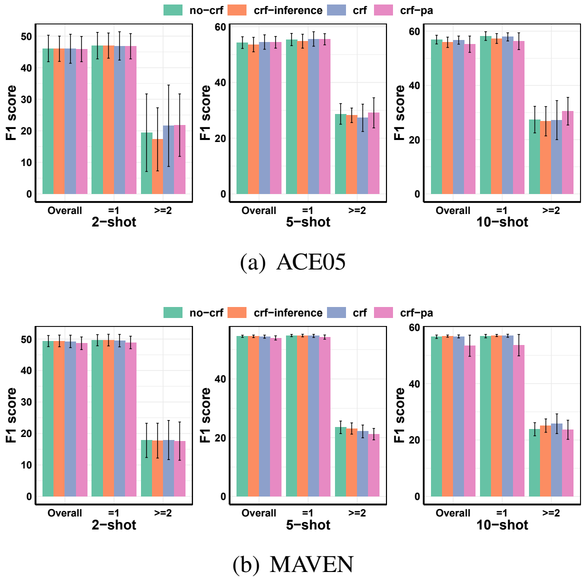


Figure 9: Performance of different (d, f) combinations on ACE05.

C.3 Prototype source

We discuss the benefit of combining two kinds of prototype sources in Section 5.2, i.e., label semantic and event mentions, and show some results in Figure 4. Here we list full results on all three datasets in Table 9. The results further validate our claims: (1) leveraging both label semantics and mentions as prototype sources improve performance under almost all settings. (2) Merging the two kinds of sources at the loss-level is the best choice among the three aggregation alternatives.



(a) ACE05

(b) MAVEN

Figure 10: Overall performance of different CRF variants on ACE05 and MAVEN datasets. We also provide performance grouped by trigger word length: $= 1$: single trigger words. ≥ 2: trigger phrases.

C.4 Contrastive Learning

Contrastive Learning (CL Hadsell et al. 2006) is initially developed for self-supervised representation

Table 8: Performance comparison of methods w/ and w/o adjustment on distance function d and transfer function f. The most appropriate distance functions are scaled euclidean distance (SEU) for ProtoNet and TapNet and scaled cosine similarity (SS) for other two. The most appropriate transfer function is normalization (N) for all four existing methods. The results are averaged among 10 repeated experiments and sample standard deviations are in round brackets. We highlight the better one for each method *w/* and *w/o* adjustment.

Methods		**ACE05**			**MAVEN**			**ERE**		
		2-shot	5-shot	10-shot	2-shot	5-shot	10-shot	2-shot	5-shot	10-shot
ProtoNet	w/o adjust	38.3(5.0)	47.2(3.9)	52.3(2.4)	44.5(2.2)	51.7(0.6)	55.4(0.2)	31.6(2.7)	39.7(2.4)	**44.3**(2.3)
	w/ adjust	**39.3**(4.6)	**49.8**(4.3)	**52.6**(1.9)	**46.7**(1.6)	**52.8**(0.6)	**56.5**(0.6)	**32.6**(3.0)	**40.1**(1.9)	44.2(1.9)
TapNet	w/o adjust	**38.7**(4.3)	49.1(4.5)	51.2(1.7)	45.7(1.8)	51.7(1.1)	55.0(0.7)	35.3(3.8)	40.2(2.5)	44.7(2.9)
	w/ adjust	37.2(5.6)	**49.8**(3.1)	**52.0**(1.9)	**46.1**(1.9)	**51.9**(0.6)	55.0(0.6)	**37.0**(4.0)	**43.4**(1.9)	**46.4**(2.9)
CONTAINER	w/o adjust	40.1(3.8)	47.7(3.3)	50.1(1.8)	44.2(1.4)	50.8(0.9)	52.9(0.3)	34.4(3.6)	39.3(1.9)	44.5(2.3)
	w/ adjust	**44.0**(3.2)	**51.1**(1.1)	**53.1**(1.8)	**44.6**(1.7)	**52.1**(0.5)	**55.1**(0.4)	**36.5**(4.1)	**42.0**(1.9)	**45.4**(1.5)
FSLS	w/o adjust	39.2(3.4)	47.5(3.2)	51.9(1.7)	46.7(1.2)	51.5(0.5)	**56.2**(0.2)	34.5(3.1)	39.8(2.5)	44.0(2.0)
	w/ adjust	**43.1**(3.4)	**51.0**(2.4)	**54.4**(1.5)	**48.3**(1.6)	**53.4**(1.6)	56.1(0.7)	**35.7**(2.1)	**40.6**(2.4)	**45.4**(1.7)

Table 9: Performance with different (1) prototype sources and (2) aggregation form. **ProtoNet**: only event mentions. **FSLS**: label semantic. **Lf-ProtoNet**: aggregate two types of prototype sources at feature-level. **Ls-ProtoNet**: at score-level. **Ll-ProtoNet**: at loss-level. The results are averaged over 10 repeated experiments and sample standard deviations are in round brackets.

Methods	**ACE05**			**MAVEN**			**ERE**		
	2-shot	5-shot	10-shot	2-shot	5-shot	10-shot	2-shot	5-shot	10-shot
ProtoNet	39.3(4.6)	49.8(4.3)	52.6(1.9)	46.7(1.6)	52.8(0.6)	56.0(0.6)	32.6(3.0)	40.1(1.9)	44.2(1.9)
FSLS	43.0(3.4)	50.6(2.4)	**54.1**(1.5)	48.3(1.6)	53.4(0.2)	56.1(0.7)	35.7(2.1)	40.6(2.4)	**45.4**(1.7)
Lf-ProtoNet	41.9(3.8)	50.8(3.0)	52.9(2.4)	49.0(1.1)	53.4(1.0)	56.3(0.7)	35.3(3.6)	41.8(1.8)	45.3(2.2)
Ls-ProtoNet	42.7(4.8)	**51.2**(2.9)	52.7(1.7)	49.3(1.9)	53.5(0.7)	56.5(0.1)	36.0(2.5)	41.3(3.6)	44.8(2.5)
Ll-ProtoNet	**43.3**(4.0)	50.9(2.7)	53.0(2.1)	**50.2**(1.5)	**54.3**(0.8)	**56.7**(0.6)	**37.6**(3.1)	**43.0**(2.4)	45.3(1.9)

learning and is recently used to facilitate supervised learning as well. It pulls samples with same labels together while pushes samples with distinct labels apart in their embedding space. We view CL as a *generalized* format of prototype-based methods and include it to the unified view. Under such view, every sample is a prototype and each single event type could have multiple prototypes. Given an event mention, its distances to the prototypes are computed and aggregated by event types to determine the overall distance to each event type.

Two types of Contrastive Learning

We name the **representation** of event mention as query and prototypes (i.e., other event mentions) as keys. Then CL could be further split into two cases, in-batch CL (Chen et al., 2020) and MoCo CL (He et al., 2020), according to where their **keys** are from. In-batch CL views other event mentions within the same batch as the keys, and the encoder for computing the queries and keys in batch-CL is updated end-to-end by back-propagation. For MoCo CL, the encoder for key is momentum-updated along the encoder for query, and it accordingly maintains a queue to store keys and utilizes them multiple times once they are previously computed. We refer readers to MoCo CL (He et al., 2020) for the details of in-batch CL and MoCo CL.

CONTAINER (Das et al., 2022) adopts in-batch CL setting for few-shot NER model and we transfer it to ED domain in our empirical study. We further compare the two types of CL for our *unified baseline* with effective components in Section 5.2 and present the full results in Table 10. We observe in-batch CL outperforms MoCo-CL when the number of the sentence is small, and the situation reverses with the increasing of sentence number. We speculate it is due to two main reasons: (1) When all sentences could be within the single batch, in-batch CL is a better approach since it computes and updates all representations of keys and queries end-to-end by back propagation, while MoCo-CL computes the key representation by a momentum-updated encoder with gradient stopping. When the sentence number is larger than batch size, however, in-batch CL lose the information of some samples in each step, while MoCo-CL keeps all samples within the queue and leverages these approximate representations for a more extensive comparison and learning. (2) MoCo-CL also has an effect of data-augmentation under few-shot ED task, since the sentence number is usually much smaller than the queue size. Then the queue would store multiple representations for each sample, which are computed and stored in different previous steps. The benefits of such data augmentation take effect when there are relatively abundant sentences and accordingly diverse augmentations.

D Class-transfer Setting-Extended

D.1 Prompt-based methods

We list the results of existing prompt-based methods on class-transfer setting in Table 11. See detailed analysis in Section 6.1.

D.2 Prototype-based methods

We list the results of existing prototype-based methods plus our developed *unified baseline* under class-transfer setting in Table 12. Note that we substitute the appropriate distance functions d and transfer functions f obtained in Section 5.2 for existing methods. See detailed analysis in Section 6.2.

Table 10: Performance with three label-enhanced approaches. The number in square bracket represents (average) sentence number under this setting. Averaged F1-scores with sample standard deviations on 10 repeated experiments are shown.

Method	ACE05 2-shot [48]	ACE05 5-shot [111]	ACE05 10-shot [212]	MAVEN 2-shot [153]	MAVEN 5-shot [360]	MAVEN 10-shot [705]	ERE 2-shot [44]	ERE 5-shot [103]	ERE 10-shot [197]
Ll-ProtoNet	43.3(4.0)	50.9(2.7)	53.0(2.1)	**50.2**(1.5)	54.3(0.8)	56.7(0.6)	37.6(3.1)	43.0(2.4)	45.3(1.9)
Ll-CONTAINER	**45.9**(3.7)	**54.0**(2.6)	55.8(1.3)	49.2(1.6)	54.3(0.6)	57.3(0.7)	**39.5**(2.4)	45.5(2.8)	46.9(1.8)
Ll-MoCo	42.8(4.1)	53.6(4.1)	**56.9**(1.6)	49.5(1.7)	**54.7**(0.8)	**57.8**(1.2)	38.8(2.4)	**46.0**(3.0)	**48.4**(2.6)

Table 11: Prompt-based methods under class-transfer setting. Averaged F1-scores with sample standard deviations on 10 repeated experiments are shown. We also list results of *w/o* and *w/* transfer for comparison.

Method		ACE05 2-shot	ACE05 5-shot	ACE05 10-shot	MAVEN 2-shot	MAVEN 5-shot	MAVEN 10-shot	ERE 2-shot	ERE 5-shot	ERE 10-shot
EEQA	*w/o transfer*	17.6(4.9)	33.2(3.8)	41.9(2.9)	14.9(4.4)	44.8(3.1)	53.9(0.7)	19.6(7.5)	36.8(3.1)	44.2(4.3)
	w/ transfer	35.1(8.5)	52.5(6.1)	**59.1**(2.5)	35.0(4.7)	54.7(1.7)	60.0(0.7)	26.8(5.2)	39.1(3.1)	45.9(2.8)
PTE	*w/o transfer*	39.7(4.1)	51.1(5.4)	54.5(3.0)	52.0(1.3)	**61.0**(1.4)	62.5(2.3)	47.1(4.9)	51.0(5.7)	54.1(4.1)
	w/ transfer	49.1(4.9)	55.4(5.8)	54.2(4.4)	52.0(2.9)	60.8(1.0)	61.5(1.5)	42.6(3.7)	**51.0**(3.1)	**55.3**(2.3)
UIE	*w/o transfer*	24.5(3.9)	39.3(3.2)	40.6(3.9)	25.3(8.1)	49.2(2.2)	57.4(2.3)	22.9(9.0)	35.1(4.2)	39.3(2.3)
	w/ transfer	47.0(5.4)	54.0(4.2)	54.7(7.3)	40.3(1.7)	49.8(1.6)	54.1(1.5)	36.9(4.6)	41.1(4.2)	41.9(4.6)
DEGREE	*w/o transfer*	33.4(6.6)	44.2(2.2)	50.5(6.3)	53.6(1.9)	56.9(5.7)	63.8(1.2)	39.1(5.9)	41.8(3.2)	43.9(6.2)
	w/ transfer	**52.4**(3.7)	**56.7**(4.6)	59.0(4.7)	**54.5**(5.1)	59.6(6.3)	**65.1**(2.7)	**50.1**(3.6)	50.3(2.8)	48.5(2.5)

Table 12: Full results about prototype-based methods under class transfer setting. Averaged F1-scores with sample standard deviations on 10 repeated experiments are shown. We enumerate all possible combinations on models of source and target datasets.

Method Source	Target	ACE05 2-shot	ACE05 5-shot	ACE05 10-shot	MAVEN 2-shot	MAVEN 5-shot	MAVEN 10-shot	ERE 2-shot	ERE 5-shot	ERE 10-shot
_	Fine-tuning	28.1(9.9)	37.0(8.3)	45.8(4.0)	21.2(11.5)	46.6(4.2)	55.3(4.8)	40.4(3.8)	45.9(3.8)	48.2(2.2)
Fine-tuning		39.1(6.7)	49.5(11.9)	51.4(9.3)	44.4(1.8)	58.3(1.9)	63.0(1.9)	34.1(6.9)	47.0(4.5)	50.0(2.3)
CONTAINER		28.7(5.8)	37.4(11.6)	42.7(8.0)	49.4(2.8)	59.3(1.4)	63.6(1.7)	36.3(8.9)	47.3(3.7)	47.3(4.0)
L-TapNet		31.7(5.7)	41.5(4.2)	43.1(2.6)	40.0(1.8)	54.3(1.4)	59.9(1.4)	36.8(4.7)	44.0(5.3)	48.7(2.1)
FSLS		42.3(8.5)	51.6(6.9)	56.7(8.6)	47.1(2.7)	58.1(1.1)	62.9(1.6)	41.2(4.7)	49.8(3.6)	53.2(3.4)
Unified Baseline		39.8(6.0)	47.4(6.2)	54.3(6.4)	48.8(1.7)	58.8(1.0)	63.9(1.0)	39.8(5.2)	46.1(3.5)	50.8(3.4)
_	CONTAINER	40.1(3.0)	47.3(5.8)	49.1(4.7)	47.9(3.5)	63.5(1.1)	68.5(2.1)	46.5(4.9)	49.2(3.0)	53.5(3.3)
Fine-tuning		37.2(9.5)	45.0(8.1)	52.7(8.7)	54.3(3.4)	64.3(1.1)	66.8(2.9)	35.0(4.0)	42.1(4.6)	47.6(4.0)
CONTAINER		30.6(5.4)	38.3(5.4)	37.6(4.5)	47.5(6.4)	57.1(3.4)	54.7(2.2)	42.1(4.8)	46.6(4.9)	51.7(2.9)
L-TapNet		33.0(2.7)	38.3(4.9)	41.6(3.6)	36.8(5.6)	43.4(3.1)	50.0(6.0)	39.6(4.4)	44.0(4.0)	48.5(2.7)
FSLS		42.8(8.0)	49.0(10.5)	53.4(11.8)	52.7(2.5)	62.2(1.5)	65.2(2.7)	39.0(5.5)	48.8(1.7)	50.8(3.1)
Unified Baseline		39.0(6.1)	45.9(9.4)	47.0(8.3)	52.8(2.1)	60.8(3.4)	60.0(4.9)	37.6(6.8)	45.9(4.5)	47.8(4.2)
_	L-TapNet	42.6 (3.8)	50.8(4.1)	50.8(2.8)	53.2(2.3)	63.3(1.6)	68.5(0.7)	44.5(4.5)	52.3(2.1)	52.5(2.5)
Fine-tuning		43.9(11.4)	54.8(9.4)	57.2(5.0)	52.2(3.2)	64.4(2.1)	68.5(0.7)	38.8(3.7)	48.1(2.5)	51.7(3.6)
CONTAINER		34.4(4.7)	43.6(4.6)	45.3(4.2)	44.9(10.8)	63.4(2.8)	**69.4**(1.1)	39.5(4.6)	49.2(4.7)	52.8(3.3)
L-TapNet		37.2(4.6)	45.4(2.8)	45.1(3.7)	52.1(2.2)	62.6(2.6)	68.0(1.4)	44.9(5.4)	49.7(2.9)	52.0(5.2)
FSLS		51.8(6.4)	59.1(6.3)	60.4(6.7)	51.1(10.2)	63.8(2.2)	68.5(1.6)	45.0(5.6)	53.6(3.1)	54.2(2.2)
Unified Baseline		45.8(5.6)	52.7(6.9)	59.4(5.3)	**56.1**(2.1)	63.6(2.5)	68.0(1.8)	45.8(4.6)	51.2(2.9)	55.3(2.2)
_	FSLS	42.9(4.0)	49.9(4.3)	52.5(2.7)	43.5(4.9)	58.2(1.1)	64.1(0.7)	46.1(7.0)	49.3(3.9)	53.5(3.5)
Fine-tuning		49.6(5.2)	56.0(7.7)	56.5(6.5)	44.9(5.0)	59.2(2.0)	64.2(1.5)	39.1(5.0)	45.7(3.2)	51.3(3.6)
CONTAINER		32.0(4.5)	40.9(4.1)	45.1(3.8)	48.0(1.6)	59.2(3.2)	64.1(2.5)	40.0(3.6)	45.6(4.6)	48.9(4.5)
L-TapNet		36.8(3.0)	43.3(3.4)	47.1(2.7)	43.9(2.1)	55.9(1.9)	62.4(1.5)	44.1(4.6)	47.3(3.1)	51.0(2.7)
FSLS		51.7(7.3)	**61.5**(7.9)	66.2(4.3)	50.8(1.9)	59.3(1.9)	65.5(1.4)	46.4(3.4)	54.4(3.5)	56.2(2.2)
Unified Baseline		44.5(8.5)	53.4(7.2)	57.7(6.4)	50.6(3.3)	59.7(0.7)	64.0(0.8)	46.1(4.4)	50.4(4.4)	55.1(2.1)
_	Unified Baseline	47.4(5.8)	55.9(3.4)	56.8(3.4)	49.1(1.2)	63.9(1.1)	68.2(1.3)	**51.7**(5.9)	**57.1**(2.0)	56.8(4.0)
Fine-tuning		51.2(4.8)	58.6(8.3)	61.9(8.7)	52.0(1.1)	63.6(2.2)	68.1(1.4)	40.0(5.9)	51.8(4.5)	57.1(3.4)
CONTAINER		34.3(3.5)	43.9(4.9)	50.9(3.1)	51.7(2.0)	63.7(1.4)	67.8(1.5)	47.5(4.6)	51.7(3.7)	55.0(2.9)
L-TapNet		42.3(4.0)	49.0(4.6)	51.6(3.7)	49.1(3.2)	63.5(2.1)	67.5(1.3)	47.2(6.1)	53.4(2.0)	55.0(3.6)
FSLS		**56.4**(5.6)	61.4(6.7)	**67.3**(4.2)	55.7(2.7)	**64.8**(1.7)	68.9(1.4)	47.6(4.1)	57.1(2.8)	**58.6**(4.0)
Unified Baseline		49.6(6.5)	60.0(6.0)	64.1(7.2)	52.9(3.3)	63.8(2.6)	69.2(0.7)	45.4(4.4)	53.5(2.3)	57.4(3.8)

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
After the acknowledgement, before the reference.

☒ A2. Did you discuss any potential risks of your work?
To our best knowledge, our work is an empirical study based on previous work and there is no potential risks of our work.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
In abstract and Section 1.

☑ A4. Have you used AI writing assistants when working on this paper?
We use ChatGPT to polish our paper, mainly on abstract and limitation part.

B ☑ Did you use or create scientific artifacts?

In Section 4.1, Appendix B.1 and Appendix B.2

☑ B1. Did you cite the creators of artifacts you used?
In Section 4.1 and Appendix B.1

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
In Appendix B.2

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
In Appendix B.2

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
To our best knowledge, no such problems in three datasets we use.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
In Appendix B.1

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
In Appendix B.2

C ☑ Did you run computational experiments?

In Section 4.2, Section 4.3 and Appendix B.4

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
In Section 4.3 and Appendix B.4

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
In Appendix B.4

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
In Section 4.2 and Appendix B.4

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
In Appendix B.4

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

□ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

□ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

□ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

□ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

□ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.