Learning Prototype Representations Across Few-Shot Tasks for Event Detection

Viet Dac Lai[1], **Franck Dernoncourt**[2], **Thien Huu Nguyen**[1,3]
[1] Dept. of Computer and Information Science, University of Oregon, Eugene, OR, USA
[2] Adobe Research, San Jose, CA, USA
[3] VinAI Research, Vietnam
{vietl,thien}@cs.uoregon.edu
dernonco@adobe.com

Abstract

We address the sampling bias and outlier issues in few-shot learning for event detection, a subtask of information extraction. We propose to model the relations between training tasks in episodic few-shot learning by introducing cross-task prototypes. We further propose to enforce prediction consistency among classifiers across tasks to make the model more robust to outliers. Our extensive experiment shows a consistent improvement on three few-shot learning datasets. The findings suggest that our model is more robust when labeled data of novel event types is limited. The source code is available at http://github.com/laiviet/fsl-proact.

1 Introduction

In Information Extraction, Event Detection (ED) is an important task that aims to identify and classify event triggers of predefined event types in text (Walker et al., 2006). Event triggers are words/phrases that most clearly indicate the occurrence of events. For example, an event detector should recognize the word *homicide* in the following sentence as a trigger word of event type *life.die.death-caused-by-violent-events*:

... the medical examiner believed the manner of death was an accident rather than a **homicide**.

Typical ED systems follow a supervised learning scheme that requires a large amount of labeled data for each predefined event type (Ji and Grishman, 2008; Nguyen and Grishman, 2015; Chen et al., 2015; Nguyen et al., 2021). Unfortunately, this requirement is usually too costly to achieve in real applications where novel event types emerge and only a few examples are available (Huang et al., 2018). As such, an ED model should be prepared to extract triggers of novel event types (i.e., beyond those provided in the training data) for which only a few examples are provided. This learning schema is known as **Few-Shot Learning** (FSL) for ED.

To emulate the learning from few examples in ED, N-way K-shot episodic training is often used to exploit existing datasets (Lai et al., 2020b; Deng et al., 2020; Lai et al., 2020a, 2021). In each training iteration, a small subset (i.e. **support set**) of N event types with K examples per type is sampled from the training data. Unfortunately, the sample size is so small ($K \in [1, 10]$) that the FSL models might suffer from sample bias, thus hindering the generalization to novel event types.

Prototypical network is a popular metric-based few-shot learning model (Snell et al., 2017) that has been explored for FSL ED (Lai et al., 2020b; Deng et al., 2020). It introduces a prototype vector for each event type by averaging the representations of the instances of that type. A non-parametric classifier then predicts the event type of a query instance based on its distances from the prototypes (Snell et al., 2017). Hence, an outlier in the support set might significantly change the prototypes and flip the label of the query instance. In addition, in ED, a NULL class is introduced to represent non-eventive mentions. This type covers every domains and every surface form except the relevant event types. Thus, this unbounded class might also present a great source of outliers for the support set.

In this paper, we mitigate the effects of poor sampling and outliers by modeling cross-task relation. First, we propose to augment the support data of the current task with those from prior tasks that essentially helps increase the population of the current support set. Therefore, it can mitigate the sample bias in the support set. Second, the averaging in prototypical network allows outliers to contribute equally to the prototype representation. We propose to use soft-attention to select the most related data samples as well as reduce the contribution of the outliers to the prototype representation. Third, a FSL model that is resistant to the outliers should produce consistent predictions regardless of support data. To implement this, we produce two prototypical-based classifiers from the two support sets of the two tasks. After that, we enforce the consistency of their predictions on query instances.

2 Model

Preliminary: In this paper, the event detection problem is formulated as a $N+1$-way K-shot episodic few-shot learning problem (Vinyals et al., 2016; Lai et al., 2020b). The model is given two sets of data: a support set $\mathcal{S}$ of labeled data, and a query set $\mathcal{Q}$ of unlabeled data. $\mathcal{S}$ consists of $(N+1)\times K$ data points in which N is the number of positive event types and K is the number of samples per event type. The model is supposed to predict the labels of the data in the query set based on the observation of the novel event types given in the support set. Formally, a FSL task with a support set and a query set is defined as follows:

$$\mathcal{S} = \{(s_i^j, a_i^j, y^j)|i \in [1,K]; j \in [0,N]\}$$
$$\mathcal{Q} = \{(s_q^j, a_q^j, y_q^j)|q \in [1,Q]; j \in [0,N]\}$$
$$\mathcal{T} = (\mathcal{S},\mathcal{Q}); \quad \mathcal{Y} = \{y^j|j \in [0,N]\}$$

where a data point (s_i^j, a_i^j, y^j) denotes a sentence s_i^j with trigger candidate a_i^j and event type y^j. Similar to prior studies in event detection, we add $y^0 = NULL$ to represent non-eventive type.

During training, development, and testing, the task $\mathcal{T}$ is sampled from three sets of data $\mathcal{D}^{train}$, $\mathcal{D}^{dev}$, and $\mathcal{D}^{test}$ whose sets of classes are $\mathcal{Y}^{train}$, $\mathcal{Y}^{dev}$, and $\mathcal{Y}^{test}$, respectively. These sets of classes are mutually disjoint to ensure that the model observes no more than K examples from a novel class.

A typical FSL model has two main modules: an encoder and a few-shot classifier. An encoder, denoted as ϕ, encodes an instance into a fixed-dimension vector $v_i^j = \phi(s_i^j, a_i^j) \in R^u$ where u is the dimension of the representation vector. A few-shot classifier classifies a query instance among classes appearing in the support set. For instance, in a prototypical network, a prototype v^j is a class-representative instance that is an average of all vectors of the j-th class $v^j = \frac{1}{K}\sum_{i=1}^{K}\phi(s_i^j, a_i^j)$.

Then the distance distribution of the query instance $q = \{s_q, a_q, y_q\}$ (Snell et al., 2017) is:

$$P(q = y^j; \mathcal{S}) = \frac{e^{-d(v_q, v^j)}}{\sum_{k=1}^{N} e^{-d(v_q, v^k)}} \qquad (1)$$

The training minimizes the cross-entropy loss, denoted by L_{ce}, over all query instances:

$$L_1(\mathcal{S},\mathcal{Q}) = \sum_{q\in\mathcal{Q}} L_{ce}(y_q, P(q;\mathcal{S})) \qquad (2)$$

Cross-task data augmentation: In conventional episode training, two consecutive training tasks $\mathcal{T}_1$ and $\mathcal{T}_2$ are not likely to share an identical event type sets, $\mathcal{Y}_1 \neq \mathcal{Y}_2$. We assume that our training process has a memory to save the latest samples of every event type used in prior tasks. Using this memory, after a certain number of training iterations, for a new task $\mathcal{T}_1$, a second sample $\mathcal{T}_2$ can always be sampled from the memory such that $\mathcal{Y}_2 = \mathcal{Y}_1$. The expected value of delaying iterations for 5-way on ACE dataset is 13 iterations ($stdev = 4$) and RAM dataset is 98 iterations ($stdev = 24$) based on 1M simulations.

Prototype Across Task We are given two tasks $\mathcal{T}_1 = (\mathcal{S}_1, \mathcal{Q}_1)$ and $\mathcal{T}_2 = (\mathcal{S}_2, \mathcal{Q}_2)$ sampled with the same set of event type $\mathcal{Y}$. The prototypes are induced from both tasks as follow:

Let $E_1^S, E_2^S, E_1^Q, E_2^Q$ be the representation vectors of $\mathcal{S}_1, \mathcal{S}_2, \mathcal{Q}_1, \mathcal{Q}_2$, respectively, where $E_1^S, E_2^S \in R^{(N+1)K\times u}$ and $E_1^Q, E_2^Q \in R^{(N+1)Q\times u}$ (returned by ϕ). Then, an attention module, denoted by att, induces intermediate representations for the support and query instances of $\mathcal{T}_1$ via weighted sums of the support vectors of the $\mathcal{T}_2$, and vice versa:

$$\hat{H}_1^{(\cdot)} = att(E_1^{(\cdot)}, E_2^S) = \frac{1}{\sqrt{u}}\text{sm}(E_1^{(\cdot)}(E_2^S)^T)E_2^S$$
$$\hat{H}_2^{(\cdot)} = att(E_2^{(\cdot)}, E_1^S) = \frac{1}{\sqrt{u}}\text{sm}(E_2^{(\cdot)}(E_1^S)^T)E_1^S$$

The final representations for both tasks are then the sum of their original representations and the cross-task representations: $H^{(\cdot)} = E^{(\cdot)} + \hat{H}^{(\cdot)}$. Then, the prototypes for tasks $\mathcal{T}_1$ and $\mathcal{T}_2$ are computed by averaging vectors of the same class from H_1^S and H_2^S, respectively (Snell et al., 2017).

Cross Task Consistency The Cross Task Consistency (CTC) further reduces the sample bias by introducing prediction consistency between classifiers generated from two tasks. Without loss of generation, we assume that one of the classifiers is impaired by the poor sampling. We employ the knowledge distillation technique (Hinton et al., 2015) that helps transfer knowledge from the stronger classifier to the weaker one. This thus makes the model more robust to the sample bias. We enforce the cross task consistency by minimizing the differences between predicted label distributions from the classifiers of two tasks as follow:

$$L_2 = KL(f_{\mathcal{S}_1}(\mathcal{Q}_1), f_{\mathcal{S}_2}(\mathcal{Q}_1)) + KL(f_{\mathcal{S}_1}(\mathcal{Q}_2), f_{\mathcal{S}_2}(\mathcal{Q}_2)) \quad (3)$$

where $f_{\mathcal{S}}$ is a prototypical classifier trained from a support set $\mathcal{S}$ and KL denotes the Kullback–Leibler divergence.

Finally, to train the model, we minimize the total loss (α is a hyper-parameter):

$$L = L_1(\mathcal{S}_1, \mathcal{Q}_1) + L_1(\mathcal{S}_2, \mathcal{Q}_2) + \alpha L_2 \quad (4)$$

Testing: As the model does not have access to the prior task of the novel class, the prototypes are computed based on the vectors of the current task only. Hence, the model turns into the original Prototypical Network (Snell et al., 2017). Our proposed methods only appy to the training process, hence, it provides a fair performance compared with prior FSL ED models.

3 Experiment

Dataset: We evaluate the proposed model on three event detection datasets. **RAMS** is a recently released large scale dataset; it provides 9124 human-annotated event triggers for 139 event subtypes (Ebner et al., 2020). **ACE** is a benchmark dataset in event extraction with 33 event subtypes (Walker et al., 2006). **LR-KBP** is a large scale event detection dataset for FSL. It merges ACE-2005 and TAC-KBP datasets and extends some event types by automatically collecting data from Freebase and Wikipedia (Deng et al., 2020). Since RAMS and ACE datasets are designed for supervised learning, we need to resplit them for FSL training. We use the exact training/development/testing split for ACE as presented in a prior study (Lai et al., 2020b). Following the same method, for RAMS, we merge the original training/development and testing splits. Then we discard 5 event subtypes whose number of samples are not sufficient for sampling. Finally, we use event types: (*Artifact-Existence, Conflict, Contact, Disaster, Government, Inspection, Manufacture, Movement*) for training, (*Justice, Life*) for development, and (*Personnel, Transaction*) for testing. For the LR-KBP dataset, we follow the same 5-fold cross-validation procedure as (Deng et al., 2020), then report the average performance. The numbers of event subtypes for the development and testing sets are set to 10 (Deng et al., 2020). The details of the splits are presented in table 1.

Split	**RAMS**		**ACE**		**LR-KBP**[*]	
	$\#C$	$\#S$	$\#C$	$\#S$	$\#C$	$\#S$
Train	95	5,340	18	2,865	72	6,732
Dev	17	1,934	11	1,227	10	561
Test	22	1,793	11	1,226	10	1,291

Table 1: Statistics of three datasets. $\#C$ and $\#S$ denote the number of classes and the number of samples, respectively. [*] The figures of one of the five folds.

FSL setting: We evaluate the model on 5+1-way 5-shot and 10+1-way 10-shot FSL settings. As it has been observed that training with more classes helps improve the model performance, we use 18+1 classes during training, while keeping 5+1 and 10+1 novel classes during testing.

Baseline: We consider three strong baselines for FSL ED. **Proto** features a prototype for each novel class and Euclidean distance function, presented in equation 1 (Snell et al., 2017). **InterIntra** is an extension of the prototypical network with two auxiliary training signals. It minimizes the distances among data points of the same class and maximizes the distances among prototypes (Lai et al., 2020b). **DMB-Proto** extends the prototypical network in a way that the representation vector for each data point is induced by a dynamic memory network running on the data of the same class (Deng et al., 2020). Since the source code of DMB-Proto is not published, we reimplement the few-shot classifier with a dynamic memory module (Xiong et al., 2016). We examine two state-of-the-art BERT-based sentence encoders ϕ for ED, i.e. BERTMLP (Yang et al., 2019) and BERTGCN (Lai et al., 2020c).

Hyperparameters: In this paper, stochastic gradient decent optimizer is used with learning rate $1e^{-4}$. The training/evaluation are set to 6,000 and 500 iterations respectively; the evaluation is done after every 500 training iterations. The dimension of the final representation is set to 512. We use dropout rate of 0.5 to prevent overfitting. The coefficient of the cross-task consistency loss is set to $\alpha = 10$ based on the best development performance ($\alpha \in \{1, 10, 100, 1000\}$.

We evaluate our ED model using the micro F1-score. The training and evaluation are done on a single Nvidia GTX 2080Ti with 11GB of GPU RAM. The training and evaluation take approximately 4 hours. We implement the model using Pytorch version 1.6.0.

Result: Table 2 reports the F-scores on the de-

Encoder	Model	5+1-way 5-shot						10+1-way 10-shot					
		RAMS		ACE		LR-KBP		RAMS		ACE		LR-KBP	
		Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Test
BERTMLP	Proto	**79.7**	68.2	82.9	79.3	83.9	82.1	73.4	61.7	81.5	78.4	**80.7**	78.0
	InterIntra	**79.7**	69.2	82.7	79.8	84.9	82.4	**74.3**	61.8	81.4	78.5	80.2	78.4
	DMB-Proto	73.2	66.9	72.9	71.9	79.8	75.2	60.1	53.8	69.5	68.2	67.4	66.2
	ProAcT	**79.7**	**74.3**	**84.5**	**83.0**	**84.1**	**83.1**	73.2	**62.3**	**82.5**	**80.5**	**80.7**	**78.7**
BERTGCN	Proto	82.0	71.0	83.5	82.1	87.2	84.8	72.4	60.7	83.3	80.4	83.2	80.0
	InterIntra	81.3	72.4	82.8	82.3	87.1	85.0	**73.7**	61.9	83.0	80.7	82.8	80.5
	DMB-Proto	54.9	47.2	61.4	60.9	70.8	63.3	54.3	43.0	69.4	69.7	65.8	60.4
	ProAcT	**82.1**	**75.7**	**86.7**	**84.7**	**88.7**	**87.3**	73.6	**62.9**	**83.7**	**81.9**	**85.4**	**83.1**

Table 2: F1-score on development and test sets of models on RAMS, ACE and LR-KBP datasets on 5+1-way 5-shot and 10+1-way 10-shot settings

velopment and testing sets of the baselines and our proposed model (called **ProAcT**) on three datasets. There are two significant points from the table. First, using the same sentence encoders, ProAcT achieves the best performance on all three datasets and settings. The improvement margins are in range [1.0%-6.1%] on the 5-shot setting and [0.7%-3.1%] on the 10-shot setting. Second, the F-score margin between ProAcT and Proto decreases as the shot number increases. This indicates that the proposed model performs better when the number of observed samples is small. As the number of shots increases, the improvement gets saturated. This finding is parallel with the fact that sample bias is more likely when the number of shots is small. Hence, our proposed method is more suitable to event detection in few-shot learning schema, especially in the case where the number of shots is limited.

Ablation study: Our proposed model involves three factors: the cross-task data (**data**), the cross-task attentive prototype (**attention**) and the cross-task consistency (**consistency**). To analyze the efficiency of these modules, we incrementally eliminate these modules from the full ProAcT model and evaluate the remaining model on 5+1-way 5-shot setting. If *attention* and *loss* are removed while *data* remains, the model and setting become a prototypical network with 5+1-way 10-shot setting during the training. This model has the same amount of support data that our model has during the training process. Note that the testing with novel classes remains 5+1-way 5-shot setting for every model. If the cross-task data is eliminated, the attentive prototype and consistency loss are also removed and the model and setting return to a prototypical network with 5+1-way 5-shot setting.

Model	**P**	**R**	**F**
ProAcT (full model)	74.9	76.7	75.7
−attention	74.1	76.0	74.9
−consistency	73.3	75.7	74.4
−attention −consistency	72.5	74.5	73.4
−data (−attention −consistency)	69.9	72.4	71.0

Table 3: Ablation study of our proposed components on 5+1 ways 5-shot setting on the RAMS dataset with BERTGCN encoder. P, R, F denote precision, recall, and f-score metrics.

Table 3 reports the performance on 5+1-way 5-shot FSL setting on RAMS with BERTGCN encoder. As shown in the table, removing any module leads to a decrease between [0.8%-1.3%] in performance. When both *attention* and *consistency* are eliminated, the performance drops of 2.3%. A further drop of 2.4% is seen if the cross-task data is eliminated. These suggest that the improvement originates from the use of cross-task data, the attention for prototype computation and the consistency of cross-task predictions.

Analysis: To further analyze the efficiency of our proposed method, we aim to discover which classes benefit the most. To do that, we compute two confusion matrices for ProAcT and Proto models on the test set of RAMS. We fix the random seed to make sure the sampling during testing are identical between two runs, hence ensuring that the proportion of classes are identical. Figure 1 presents the difference of two confusion matrices exhibited by the proposed model ProAct and the prototypical network Proto. There are two major observations from the figure. First, overall ProAcT produces more accurate predictions than Proto, as shown on the diagonal. Second, ProAcT involves remarkably more correct predictions for negative examples than Proto. In the mean time, it generates significantly lower number of errors in both false positive and false negative related to the NULL class, i.e. *Other* class in Figure 1, suggesting that our proposed model effectively mitigates the effect of noise introduced by the NULL class.

4 Related works

Prior studies in ED mainly follow the supervised learning scheme. The early work focuses on feature engineering with statistical models (Ahn, 2006; Ji and Grishman, 2008; Liao and Grishman, 2010; Hong et al., 2011). Recently, many deep learning architectures have been explored for automatic feature learning (Nguyen and Grishman, 2015; Chen et al., 2015; Nguyen et al., 2016; Feng et al., 2016; Nguyen and Grishman, 2018; Lai et al., 2020c; Veyseh et al., 2021). Some recent studies have also introduced methods to extending ED to new event types (Liao and Grishman, 2011; Huang and Riloff, 2012; Nguyen et al., 2016b,g; Chen et al., 2017; Huang et al., 2018; Tong et al., 2020; Lai et al., 2020b).

FSL has been extensively studied in computer vision (Vinyals et al., 2016; Snell et al., 2017; Finn et al., 2017; Lee et al., 2019; Fei et al., 2021). Recent work has also considered FSL for tasks in natural language processing (Han et al., 2018; Bao et al., 2020). For ED, prior FSL work has mostly relied on Prototypical network (Lai et al., 2020b; Deng et al., 2020). However, these models do not explore cross-task modeling as we do.

5 Conclusion

In this paper, we propose to exploit the relationship between training tasks for few-shot learning event detection. We compute prototypes based on cross-task modeling and present a regularization to enforce prediction consistency of classifiers across tasks. The experiment results show that exploiting cross-task relation can alleviate the poor sampling and outliers in the support set for FSL in ED. In the future, we will extend our method to other tasks in information extraction such as named entity recognition and argument extraction.

Acknowledgement

This research has been supported by the Army Research Office (ARO) grant W911NF-21-1-0112 and the NSF grant CNS-1747798 to the IUCRC Center for Big Learning. This research is also based upon work supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via IARPA Contract No. 2019-19051600006 under the Better Extraction from Text Towards Enhanced Retrieval (BETTER) Program. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ARO, ODNI, IARPA, the Department of Defense, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein. This document does not contain technology or technical data controlled under either the U.S. International Traffic in Arms Regulations or the U.S. Export Administration Regulations. We thank the anonymous reviewers and Tracy King for their helpful feedback.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*.

Yujia Bao, Menghua Wu, Shiyu Chang, and Regina Barzilay. 2020. Few-shot text classification with distributional signatures. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Yubo Chen, Shulin Liu, Xiang Zhang, Kang Liu, and Jun Zhao. 2017. Automatically labeled data generation for large scale event extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

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*.

Seth Ebner, Patrick Xia, Ryan Culkin, Kyle Rawlins, and Benjamin Van Durme. 2020. Multi-sentence argument linking. In *Proceedings of the Annual Meet-*

ing of the Association for Computational Linguistics (ACL).

Nanyi Fei, Zhiwu Lu, Tao Xiang, and Songfang Huang. 2021. Melr: Meta-learning via modeling episode-level relationships for few-shot learning. In *Proceedings of the International Conference on Learning Representations (ICLR).*

Xiaocheng Feng, Lifu Huang, Duyu Tang, Heng Ji, Bing Qin, and Ting Liu. 2016. A language-independent neural network for event detection. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL).*

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML).*

Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2018. FewRel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4803–4809, Brussels, Belgium. Association for Computational Linguistics.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *Proceedings of the NeurIPS Deep Learning and Representation Learning Workshop.*

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL).*

Lifu Huang, Heng Ji, Kyunghyun Cho, and Clare R Voss. 2018. Zero-shot transfer learning for event extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL).*

Ruihong Huang and Ellen Riloff. 2012. Modeling textual cohesion for event extraction. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI).*

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL).*

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

Viet Dac Lai, Minh Van Nguyen, Thien Huu Nguyen, and Franck Dernoncourt. 2021. Graph learning regularization and transfer learning for few-shot event detection. In *Proceddings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval.*

Viet Dac Lai, Thien Huu Nguyen, and Frank Dernoncourt. 2020b. Extensively matching for few-shot learning event detection. In *Proceedings of the First Joint Workshop on Narrative Understanding, Storylines, and Events.*

Viet Dac Lai, Tuan Ngo Nguyen, and Thien Huu Nguyen. 2020c. Event detection: Gate diversity and syntactic importance scores for graph convolution neural networks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5405–5411, Online. Association for Computational Linguistics.

Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. 2019. Meta-learning with differentiable convex optimization. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR).*

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL).*

Shasha Liao and Ralph Grishman. 2011. Acquiring topic features to improve event extraction: in preselected and balanced collections. In *RANLP.*

Minh Van Nguyen, Viet Dac Lai, and Thien Huu Nguyen. 2021. Cross-task instance representation interactions and label dependencies for joint information extraction with graph convolutional networks. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT).*

Thien Nguyen and Ralph Grishman. 2018. Graph convolutional networks with argument-aware pooling for event detection. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI).*

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT).*

Thien Huu Nguyen, Lisheng Fu, Kyunghyun Cho, and Ralph Grishman. 2016b. A two-stage approach for extending event detection to new types via neural networks. In *Proceedings of the 1st ACL Workshop on Representation Learning for NLP (RepL4NLP).*

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of the 53rd Annual*

Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP).

Thien Huu Nguyen, Adam Meyers, and Ralph Grishman. 2016g. New york university 2016 system for kbp event nugget: A deep learning approach. In *Proceedings of Text Analysis Conference (TAC).*

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS).*

Meihan Tong, Bin Xu, Shuai Wang, Yixin Cao, Lei Hou, Juanzi Li, and Jun Xie. 2020. Improving event detection via open-domain trigger knowledge. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL).*

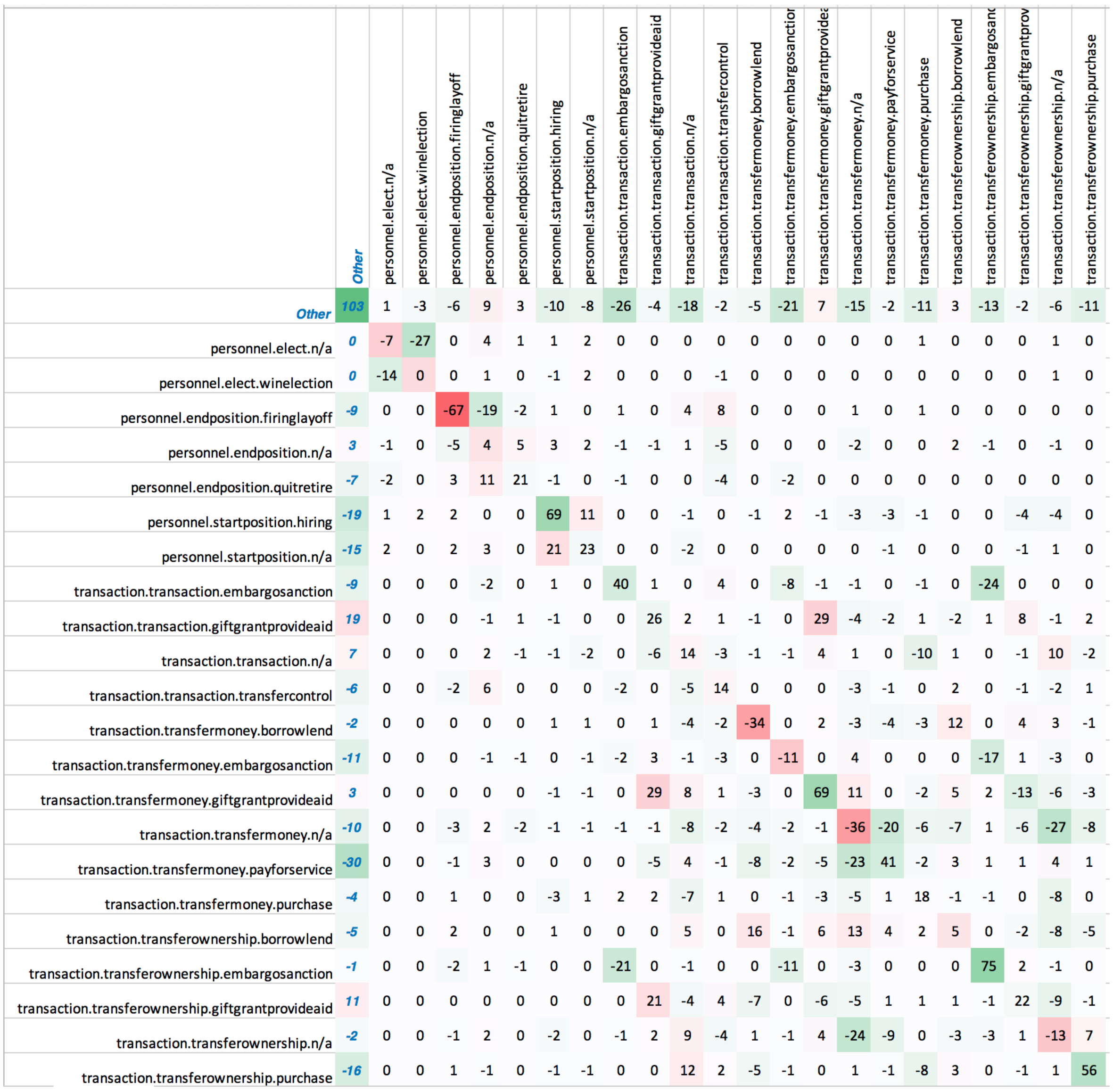
Amir Pouran Ben Veyseh, Viet Lai, Franck Dernoncourt, and Thien Huu Nguyen. 2021. Unleash GPT-2 power for event detection. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP).*

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS).*

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. Ace 2005 multilingual training corpus. In *Technical report, Linguistic Data Consortium.*

Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. In *Proceedings of the International Conference on Machine Learning (ICML).*

Sen Yang, Dawei Feng, Linbo Qiao, Zhigang Kan, and Dongsheng Li. 2019. Exploring pre-trained language models for event extraction and generation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL).*



	Other	personnel.elect.n/a	personnel.elect.winelection	personnel.endposition.firinglayoff	personnel.endposition.n/a	personnel.endposition.quitretire	personnel.startposition.hiring	personnel.startposition.n/a	transaction.transaction.embargosanction	transaction.transaction.giftgrantprovideaid	transaction.transaction.n/a	transaction.transaction.transfercontrol	transaction.transfermoney.borrowlend	transaction.transfermoney.embargosanctio	transaction.transfermoney.giftgrantprovidea	transaction.transfermoney.n/a	transaction.transfermoney.payforservice	transaction.transfermoney.purchase	transaction.transferownership.borrowlend	transaction.transferownership.embargosanc	transaction.transferownership.giftgrantprov	transaction.transferownership.n/a	transaction.transferownership.purchase
Other	*103*	1	-3	-6	9	3	-10	-8	-26	-4	-18	-2	-5	-21	7	-15	-2	-11	3	-13	-2	-6	-11
personnel.elect.n/a	*0*	-7	-27	0	4	1	1	2	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0
personnel.elect.winelection	*0*	-14	0	0	1	0	-1	2	0	0	0	-1	0	0	0	0	0	0	0	0	0	1	0
personnel.endposition.firinglayoff	*-9*	0	0	-67	-19	-2	1	0	1	0	4	8	0	0	0	1	0	1	0	0	0	0	0
personnel.endposition.n/a	*3*	-1	0	-5	4	5	3	2	-1	-1	1	-5	0	0	0	-2	0	0	2	-1	0	-1	0
personnel.endposition.quitretire	*-7*	-2	0	3	11	21	-1	0	-1	0	0	-4	0	-2	0	0	0	0	0	0	0	0	0
personnel.startposition.hiring	*-19*	1	2	2	0	0	69	11	0	0	-1	0	-1	2	-1	-3	-3	-1	0	0	-4	-4	0
personnel.startposition.n/a	*-15*	2	0	2	3	0	21	23	0	0	-2	0	0	0	0	0	-1	0	0	0	-1	1	0
transaction.transaction.embargosanction	*-9*	0	0	0	-2	0	1	0	40	1	0	4	0	-8	-1	-1	0	-1	0	-24	0	0	0
transaction.transaction.giftgrantprovideaid	*19*	0	0	0	-1	1	-1	0	0	26	2	1	-1	0	29	-4	-2	1	-2	1	8	-1	2
transaction.transaction.n/a	*7*	0	0	0	2	-1	-1	-2	0	-6	14	-3	-1	-1	4	1	0	-10	1	0	-1	10	-2
transaction.transaction.transfercontrol	*-6*	0	0	-2	6	0	0	0	-2	0	-5	14	0	0	0	-3	-1	0	2	0	-1	-2	1
transaction.transfermoney.borrowlend	*-2*	0	0	0	0	0	1	1	0	1	-4	-2	-34	0	2	-3	-4	-3	12	0	4	3	-1
transaction.transfermoney.embargosanction	*-11*	0	0	0	-1	-1	0	-1	-2	3	-1	-3	0	-11	0	4	0	0	0	-17	1	-3	0
transaction.transfermoney.giftgrantprovideaid	*3*	0	0	0	0	0	-1	-1	0	29	8	1	-3	0	69	11	0	-2	5	2	-13	-6	-3
transaction.transfermoney.n/a	*-10*	0	0	-3	2	-2	-1	-1	-1	-1	-8	-2	-4	-2	-1	-36	-20	-6	-7	1	-6	-27	-8
transaction.transfermoney.payforservice	*-30*	0	0	-1	3	0	0	0	0	-5	4	-1	-8	-2	-5	-23	41	-2	3	1	1	4	1
transaction.transfermoney.purchase	*-4*	0	0	1	0	0	-3	1	2	2	-7	1	0	-1	-3	-5	1	18	-1	-1	0	-8	0
transaction.transferownership.borrowlend	*-5*	0	0	2	0	0	1	0	0	0	5	0	16	-1	6	13	4	2	5	0	-2	-8	-5
transaction.transferownership.embargosanction	*-1*	0	0	-2	1	-1	0	0	-21	0	-1	0	0	-11	0	-3	0	0	0	75	2	-1	0
transaction.transferownership.giftgrantprovideaid	*11*	0	0	0	0	0	0	0	0	21	-4	4	-7	0	-6	-5	1	1	1	-1	22	-9	-1
transaction.transferownership.n/a	*-2*	0	0	-1	2	0	-2	0	-1	2	9	-4	1	-1	4	-24	-9	0	-3	-3	1	-13	7
transaction.transferownership.purchase	*-16*	0	0	1	-1	0	-1	-1	0	0	12	2	-5	-1	0	1	-1	-8	3	0	-1	1	56

Figure 1: The differences of confusion matrices between ProAcT and Proto models. On the main diagonal, a positive value implies that ProAcT predicts more accurate than Proto, whereas on the rest of the matrix, a negative value indicates that ProAcT creates less error than Proto. Visually, a green cell indicates that the prediction of ProAcT is more accurate than those from Proto. Red cells suggests the cases where Proto is better than ProAcT.