Extensively Matching for Few-shot Learning Event Detection

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

Abstract

Current event detection models under supervised learning settings fail to transfer to new event types. Few-shot learning has not been explored in event detection even though it allows a model to perform well with high generalization on new event types. In this work, we formulate event detection as a few-shot learning problem to enable to extend event detection to new event types. We propose two novel loss factors that matching examples in the support set to provide more training signals to the model. Moreover, these training signals can be applied in many metric-based few-shot learning models. Our extensive experiments on the ACE-2005 dataset (under a few-shot learning setting) show that the proposed method can improve the performance of few-shot learning.

1 Introduction

Event Detection (ED) is an important task in Information Extraction (IE) in Natural Language Processing (NLP). Event Detection is the task to detect event triggers from a given text (e.g. a sentence) and classify it into one of the event types of interest. The following sentence is an example of ED:

In 1997, the company **hired** *John D. Idol to take over as chief executive.*

In this example, an ideal event detection system should detect the word *hired* as an event, and classify it to class of *Personnel:Start-Position*, assuming that *Personnel:Start-Position* is in the set of interested classes.

The current works in ED typically employ traditional supervised learning based on feature engineering (Li et al., 2014; Chen et al., 2017) and neural networks (Nguyen et al., 2016a; Chen et al., 2018; Lu and Nguyen, 2018). The main problem with supervised learning models is that they can not perform well on unseen classes (e.g. training a model to classify daily events, then run this model to classify laboratory operations). As a result, supervised learning ED can not extend to unseen event types. A trivial solution is to annotate more data for unseen event types, then retraining the model with newly annotated data. However, this method is usually impractical because of the extremely high cost of annotation (Liu et al., 2019).

A human can learn about a new concept with limited supervision e.g. one can detect and classify events with 3-5 examples (Grishman et al., 2005). This motivates the setting we aim for event detection: **few-shot learning** (FSL). In FSL, a trained model rapidly learns a new concept from a few examples while keeping great generalization from observed examples (Vinyals et al., 2016). Hence, if we need to extend event detection into a new domain, a few examples are needed to activate the system in the new domain without retraining the model. By formulating ED as FSL, we can significantly reduce the annotation cost and training cost while maintaining highly accurate results.

In a few shot learning iteration, the model is given a support set and a query instance. The support set consists of examples from a small set of classes. A model needs to predict the label of the query instance in accordance with the set of classes appeared in the support set. Typical methods employ a neural network to embed the samples into a low-dimension vector space (Vinyals et al., 2016; Snell et al., 2017), then, classification is done by matching those vectors based on vector distances (Vinyals et al., 2016; Snell et al., 2017; Sung et al., 2018). One potential problem of prior FSL methods is that the model relies solely on training signals between query instance and the support set (Vinyals et al., 2016; Snell et al., 2017; Sung et al., 2018). Thus, the matching information between samples in the support set has not been exploited yet. We believe that this is not an efficient use of training data because dataset in ED is very small

Proceedings of the 1st Joint Workshop on Narrative Understanding, Storylines, and Events, pages 38–45
July 9, 2020.

(Grishman et al., 2005). Therefore, in this study, we propose to train an ED model using matching information (1) between query instance and the support set and (2) between the samples in the support themselves. This is implemented by adding two auxiliary factors into the loss function to constrain the learning process.

We apply the proposed training signals to different FSL models on the benchmark event detection dataset (Grishman et al., 2005). The experiments show that the training signal can improve the performance of the examined FSL models. To summarize, our contributions to this work include:

- We formulate event detection as a few-shot learning problem to extend ED to new event types and provide a baseline for this new research direction. To our best knowledge, this is a new branch of research that has not been explored.
- We propose two novel training signals for FSL. These signals can remarkably improve the performance of existing FSL models. As these signals do not require any additional information (e.g. dependency tree or part-of-speech), they can be applied in any metric-based FSL models.

2 Related work

Early studies in event detection mainly address feature engineering for statistical models (Ahn, 2006; Ji and Grishman, 2008; Hong et al., 2011; Li et al., 2014, 2015) including semantic features and syntactic features. Recently, due to the advances with deep learning, many neural network architectures have been presented for ED, e.g. convolutional neural networks (CNN) (Chen et al., 2015; Nguyen and Grishman, 2015, 2016; Nguyen et al., 2016b), recurrent neural networks (RNN) (Liu et al., 2017; Chen et al., 2018; Nguyen et al., 2016a; Nguyen and Nguyen, 2018) and graph convolutional neural networks (GCN) (Nguyen and Grishman, 2018; Pouran Ben Veyseh et al., 2019). These methods formulate ED as a supervised learning problem which usually fails to predict the labels of new event types.

By transitioning the symbolic event types to descriptive event types in the form of bags of keywords (Bronstein et al., 2015; Peng et al., 2016; Lai and Nguyen, 2019), the adaptibility of event detection can be formed as a supervised-learning problem. However, these studies have not examined FSL as we do in this work. One can also address this problem in zero-shot learning with data generated from abstract meaning representation (Huang et al., 2018) or two-stage pipeline (trigger identification and few-shot event classification) based on dynamic memory network (Deng et al., 2020). A recent study has employed few-shot learning for event classification (Lai et al., 2020). Our work is similar in terms of formulation, however, we consider it in a larger extent of event detection where the *NULL* event is also included.

Few-shot learning has been studied early in the literature (Thrun, 1996). Before the era of the deep neural network, FSL approaches focused on building generative models that can transfer priors across classes. However, these methods are hard to apply to real applications because they require a subject-dedicated design such as handwritten characters (Lake et al., 2013; Wong and Yuille, 2015). As a result, they cannot capture the nature of the distribution (Salimans et al., 2016). Later studies, based on deep neural network, proposed metric learning to model the distribution of distance among classes, (Koch et al., 2015) with many incremental improvements in distance functions such as cosine similarity (Vinyals et al., 2016), Euclidean distance (Snell et al., 2017) and learnable distance function (Sung et al., 2018). Metric-based FSL presents its advantages in two dimensions. First, it is based on the well-studied theory in distance functions. Second, the simplicity in architecture and training processes can encourage its application in practice. Recently, meta-learning with parameter update strategy is also proposed to enable the models to learn quickly in few training iterations (Santoro et al., 2016; Finn et al., 2017).

3 Methodology

Our goal in this work is to formulate ED as a FSL problem, which has not been done in prior work. In order to achieve this, this section is divided into three parts. In the section 3.1 we present the overall framework that formulate Event Detection as an Few-Shot Learning problem. Then, we present popular models for FSL in the prior work and common sentence encoders which have been widely used in ED in section 3.2. Finally, we present two novel reguarlization technique to further improve the FSL model for ED in section 3.3.

3.1 Event Detection as Few-shot Learning

In few-shot learning, models learn to predict the label of a query instance x given a support set S (a set of well-classified instances) and a set of classes C, which appears in the support set S. Prior studies in FSL employ N-way K-shot setting, in which there are N clusters, which represent N classes, each cluster contains K data points (i.e., examples).

However, this setting is designed for problems that do not involve the "NULL" class (e.g., image classification and event classification). In event detection, the systems need to predict whether a query instance is an event (positive event type) or not (negative event type – the "NULL" type) before it is further classified into one of the classes of interest. To this end, we propose to extend the N-way K-shot setting to be N+1-way K-shot setting. In this setting, the support set contains N clusters representing N positive event types and 1 cluster representing the NULL event type. The support set is denoted as follows:

$$\begin{aligned} S = &\{(s_1^1, a_1^1, t_1), \ldots, (s_1^K, a_1^K, t_1), \\ &\cdots \\ &(s_N^1, a_N^1, t_N), \ldots, (s_N^K, a_N^K, t_N), \\ &(s_{N+1}^1, a_{N+1}^1, t_{null}), \ldots, (s_1^K, a_{N+1}^K, t_{null})\} \end{aligned}$$

where:

- $\{t_1, t_2, \cdots t_N\}$ is the set of positive labels, which indicate an event
- t_{null} a special label for non-event.
- (s_i^j, a_i^j, t_i) indicates that the a_i^j-th word in the sentence s_i^j is the trigger word of an event mention with the event type t_i

3.2 Framework

Follow prior studies in FSL (Gao et al., 2019), we employ the metric-based FSL framework with three components: instance encoder, prototype encoder, and classification module.

3.2.1 Instance Encoder

Given a sentence of L words $\{w_1, w_2, \cdots, w_L\}$ and the event mention w_a, which is the a-th word of the sentence, we first map discrete words to a continuous high dimensional vector space to facilitate neural network using both pre-trained word embedding and position embedding as follow:

- In order to capture the syntactic and semantic of the word itself, we map each word in the sentence to a single vector using pre-trained word embedding, following previous studies in ED (Nguyen and Grishman, 2015). After this step, we derive a sequence of vectors $\{e_1, e_2, \cdots, e_L\}$ where $e_i \in R^u$.
- To provide a sense of the relative position of a word regarding the position of the anchor word, we further provide position embedding. It is mapped from the relative distance, $i - a$, of the i-th word with respect to the anchor word, a-th word to a single vector $p_i \in R^v$. We randomly initialize this word embedding and update the embedding during the training process.
- Following previous work (Nguyen and Grishman, 2015), the final embedding of a word w_i is derived by concatenating word embedding and position embedding $m_i = [e_i, p_i] \in R^{u+v}$.

Once we get the embedding for the whole sentence $E(s) = \{m_1, m_2, \cdots, m_L\}$, we employ a neural network, denoted as f, to encode the information of an instance (s, a) of the anchor w_a under the context in the sentence s into a single vector $v = f(E(s), a)$. In this work, consider the three following neural network architectures for this encoding purpose:

- Convolution Neural Network (CNN) (Kim, 2014) encodes the sentence by convolution operation on k consecutive vectors representing k-gram. Follow (Nguyen and Grishman, 2015), we use multiple kernel sizes $k \in \{2, 3, 4, 5\}$ to cover the context with 150 filters for each kernel size. To squeeze the information of the sentence, we apply max pooling to the top convolution layer to get a pooled vector p. We also introduce local embedding $e_{[a-w,a+w]}$ with window size $w = 2$. We concatenate pooled vector and local embeddings, and feed them through multiple dense layer to get the final representation:

$$v = W[p, e_{[a-w,a+w]}]$$

- Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997), at each step i,

computes a hidden vector h_i from the hidden vector of the previous step h_{i-1} and the current input vector e_i. To capture the context from both sides a word in the sentence, we employ two separate LSTMs running on forward and backward directions. Eventually, we can obtain two sequence of hidden vector $\{h_i^{forward}, \cdots, h_L^{forward}\}$ and $\{h_i^{backward}, \cdots, h_L^{backward}\}$. Finally, we concatenate the a-th vectors, at the position of the anchor, to form the representation of the instance:

$$v = \text{concat}(h_a^{forward}, h_a^{backward})$$

- Graph Convolutional Neural Network features graph convolution (Kipf and Welling, 2017) on syntactic dependency graph, which allows the model to access to the nonconsecutive words based on the connection on the syntactic dependency tree. Following (Nguyen and Grishman, 2018), we transform the dependency tree into a syntactic graph by making it an undirected graph and adding node loops. The hidden vectors h_i^l of the l-th vector is obtained by feeding hidden vectors of the $l-1$-th layer through a GCN layer (Kipf and Welling, 2017). The final representation is the hidden vector in the top layer at the position of the trigger h_a^L where $L = 2$ is the number of GCN layers.

3.2.2 Prototype Encoder

This module computes a representative vector, called **prototype**, for each class $t \in T$ in the support set S from its instances' vectors. We employ two variants of prototype computation.

The first version, proposed in the original Prototypical Network (Snell et al., 2017), considers all representation vectors are equally important. To calculate the prototype for a class t_i, it aggregates all the representation vectors of the instance of class t_i, and then perform averaging over all vectors :

$$\mathbf{c}_i = \frac{1}{K} \sum_{(s_i^j, a_i^j, t_i) \in S} f(E(s_i^j), a_i^j) \quad (1)$$

On the other hand, it was claimed that the supporting vectors are conditionally important with respect to the query (q, p). Thus, the second version computes the prototype as a weighted sum of the supporting vectors. The weights are obtained by attention mechanism according to the representational vector of the query as follow:

$$\mathbf{c}_i = \sum_{(s_i^j, a_i^j, t_i) \in S} \alpha_{ij} f(E(s_i^j), a_i^j)$$
$$\text{where } \alpha_{ij} = \frac{\exp(b_{ij})}{\sum_{(s_i^k, a_i^k, t_i) \in S} \exp(b_{ik})};$$
$$b_{ij} = \sum \Big[\sigma(f(E(s_i^j), a_i^j) \odot f(E(q), p))\Big];$$
$$\odot \text{ denotes the element-wise product.} \quad (2)$$

3.2.3 Classification Module

This module computes the distribution on all the event types T of a query instance $x = (q, p)$ using a distance/similarity function $d : R \leftarrow R^d$.

$$P(y = t_i | x, S) = \frac{\exp(-d(f(q,p), \mathbf{c}^i))}{\sum_{j=1}^{N} \exp(-d(f(q,p), \mathbf{c}^j))} \quad (3)$$

where d is a distance/similarity function, and $\mathbf{c}^i$ and $\mathbf{c}^j$ are the prototype vectors obtained in either Equation (1) or Equation (2) from the support set S.

In this paper, we examine three kinds of distance/similarity function with prototype module to form 4 model as follow:

- Cosine similarity with averaging prototype as **Matching** network (Vinyals et al., 2016).
- Euclidean distance with averaging prototype as **Proto** network (Snell et al., 2017).
- Euclidean distance with weighted sum prototype as **Proto+Att** network (Gao et al., 2019).
- Learnable distance function with averaging prototype as **Relation** network (Sung et al., 2018).

3.3 Training Objectives

In the literature, a metric-based FSL model is typically trained by minimizing the negative log-likelihood as follow:

$$L_{query}(x, S) = -\log P(y = t | x, S) \quad (4)$$

where x, t, S are query instance, ground truth label, and support set, respectively.

Model	5+1-way5-shot			10+1-way 10-shot		
Encoder	CNN	LSTM	GCNN	CNN	LSTM	GCNN
Proto	70.85	68.77	71.30	61.43	57.89	62.36
Proto+Att	71.23	69.32	72.76	63.50	59.56	65.08
Relation	54.36	68.33	58.37	41.37	62.85	44.43
Matching	34.71	49.40	32.49	23.05	33.84	21.51

Table 1: F1-score (micro) of models using CNN, LSTM and GCN encoders without proposed losses.

Encoder	Model	5+1-way 5-shot		10+1-way 10-shot	
		Original	+ $L_{inter} + L_{intra}$	Original	$+L_{inter} + L_{intra}$
CNN	Proto	70.85	**72.07**	61.43	**62.84**
LSTM	Proto	68.77	**78.09**	57.89	**72.78**
GCN	Proto	71.30	**71.82**	62.36	**63.49**
CNN	Proto+Att	71.23	**72.46**	63.5	**64.38**
LSTM	Proto+Att	69.32	**78.44**	59.56	**72.94**
GCN	Proto+Att	72.76	**72.92**	65.08	**66.10**

Table 2: F1-score (micro) of models using CNN, LSTM, and GCN. *Original* columns show the models without additional training signal. $L_{inter} + L_{intra}$ columns demonstrate the models with additional inter and intra loss functions.

This loss function exploits the signal of matching information between the query instance and the supporting instances. It can work efficiently in computer vision because the number of samples in computer vision datasets are typically huge. However, in NLP tasks, the dataset is commonly relatively much smaller (e.g. ACE 2005 contains 4000 positive examples). So using this loss function is not enough to deliver a good system.

Therefore, providing more training signals is crucial to the problem which involves a small dataset. Fortunately, the support set is a well-classified set of instances with K examples per class in a total of N classes. In this paper, we proposed two ways to exploit this resourceful set as follow:

- Intra-cluster matching: We argue that the representational vectors in the same class should be close to each other. Therefore, we minimize the distance between instance in the same class.

$$L_{intra} = \sum_{i=1}^{N} \sum_{k=1}^{K} \sum_{j=k+1}^{K} \text{mse}(v_i^j, v_i^k) \quad (5)$$

- Inter-cluster information: We also argue that the clusters should distribute far away from each other. Hence, their prototypes are also distant from the other. Hence, we maximize the distances between pairs of prototypes.

$$L_{inter} = 1 - \sum_{i=1}^{N} \sum_{j=i+1}^{N} \text{cosine}(c_i, c_j) \quad (6)$$

In this work, we train our model using a combination of the loss functions in equations 4, 5,6. We control the contribution of the additional losses by two hyperparameters β and γ as follow:

$$L = L_{query} + \beta \hat{L}_{intra} + \gamma \hat{L}_{inter} \quad (7)$$

where $\hat{L}_{intra}$ and $\hat{L}_{inter}$ are scaled losses with respect to L_{query}, and β and γ are the trade-off parameters.

4 Experiments

4.1 Data

We use the ACE-2005 dataset to evaluate all of the models in this study. ACE-2005 is a benchmark dataset in event detection with 33 positive event subtypes, which are grouped into 8 event types *Business, Contact, Conflict, Justice, Life, Movement, Personnel, and Transaction*. Although the dataset is split into training, development, and testing sets, we cannot use these splits directly because, in FSL, the set of event types in the training set and testing sets are disjoint. Therefore, we further split these datasets to satisfy three conditions for FSL:

- The set of event types in the training set T^{train} are disjoint to those in the development and the testing set:
  $T^{dev} \equiv T^{test}; T^{train} \cap T^{test} = \emptyset;$
- In order to run FSL with the 10-way 10-shot setting, the set of event subtypes should contain at least 10 subtypes.
- The training set should contain as many samples as possible.

Based on these criteria, we use all samples belonging to 4 event types: *Business, Contact, Conflict* and *Justice* as the training set. While the rest (*Life, Movement, Personnel* and *Transaction*) are used for the development and testing sets. We split the sample by ratio 50:50 in every subtype to ensure the balance of the development and the testing set. Finally, since there are event types that have less than 15 examples, we eliminate all of these from the training, development, and testing set.

4.2 Hyper-parameters

We evaluate using 5+1-way 5-shot and 10+1-way 10-shot FSL settings. Although it was seen that the higher number of classes we have during the training time, the better performance on testing (Snell et al., 2017), we avoid feeding all event types in every iteration during training time. We manage to sample 20 positive classes (over 21 in the training set) in each training iteration.

We initialize the embedding vectors with 300-dimension GLoVe embedding, trained from 6 billion tokens. We use 50-dimension position embedding and initialize it randomly. These embedding vectors are updated during training time.

We train Proto, Proto+Att, and Matching using Stochastic Gradient Decent (SGD) optimizer while Relation is trained with AdaDelta optimizer because SGD hardly converges with Relation network. The learning rate is initialized to 0.03 and decays after every 500 iterations. We trained our models in 2500 iterations and evaluation at every 200 iterations.

In order to find the best set of β and γ, we do grid search with with $(\beta, \gamma) \in \{0.0, 0.1, 0.2, 0.3\}^2$.

4.3 Result

In this section, we perform our experiment in three steps:(1) find the best FSL models among Proto, Proto+Att, Matching, Relation models; (2) evaluate the proposed additional training factors and (3) analyze the effectiveness of each training factor in an ablation study.

Table 1 shows the F-scores of four models using three kinds of sentence encoders on the ACE-2005 dataset under 5+1-way 5-shot and 10+1-way 10-shot FSL settings without our proposed losses. As can be seen from the Table 1, the performance of the models on 5+1-way 5-shot is always better than 10+1-way 10-shot because the number of classes needs to be classified in the 10+1-way setting is almost twice as much of in 5+1-way setting. Second, we can see that Prototypical-based (Proto and Proto+Att) models outperform the Matching network and the Relation network on both FSL settings. Among Prototypical network models, Proto+Att is slightly better than Proto with a 0.8% performance gap in the 10+1-way 10-shot setting.

Most importantly, Table 2 presents the F-scores of Proto and Proto+A with the proposed loss functions (i.e., L_{intra}, L_{inter}). As we can see from the table, the proposed loss functions can significantly improve the performance of Proto and Proto+Att models over different encoders (i.e., CNN, LSTM, and GCN), clearly demonstrating the benefits of the intra and inter-similarity constraints in this work.

4.4 Ablation Study

In this study, we introduce two penalization factors, presented in Equations 5 and 6.

Besides the FSL formulation for event detection, a major contribution in this work involves the two loss functions L_{intra} and L_{inter} to improve the representation vectors for the models. To evaluate the contribution of these terms, Table 3 shows the performance of the FSL models with different combinations of loss functions on the development set. In particular, we focus on the prototypical-based FSL model on the 5+1-way 5-shot setting in this analysis (although the similar trends of the performance are also observed for the other models and settings). The "*Original*" column corresponds to the models where both L_{inter} and L_{intra} are not applied. The other columns, on the other hand, report the performance of the models when the combinations L_{inter}, L_{intra}, and $L_{inter} + L_{intra}$ of the loss terms are introduced.

It is clear from the table that both loss terms are important for the FSL models for ED as eliminating any of them would significantly hurt the performance excepting the Proto+Att model with GCN encoder. The best performance is achieved with

Encoder	FSL Model	Original	+Inter	+Intra	+Intra+Inter
CNN	Proto	67.92	68.78	68.83	69.37
LSTM	Proto	65.94	65.28	72.07	77.56
GCN	Proto	69.28	70.05	69.49	70.11
CNN	Proto+Att	69.90	70.23	70.06	70.43
LSTM	Proto+Att	67.26	67.48	72.00	77.81
GCN	Proto+Att	71.65	71.75	71.56	71.18

Table 3: Ablation study: F1-score (micro) of Prototypical-based models on dev set with 5+1-way 5-shot FSL setting

both loss terms are applied, thus testifying to the benefits of the proposed regularization techniques in this work.

5 Conclusion

In this paper, we address the problem of extending event detection to unseen event types through few-shot learning. We investigate four metric-based few-shot learning models with different encoder types (CNN, LSTM, and GCN). Moreover, we propose two novel loss functions to provide more training signals to the model exploiting domain-matching information in the support set. Our extensive experiments show that our method increases the efficiency of using training data, resulting in better classification performance. Our ablation study shows that both intra-cluster matching and inter-cluster matching contributes to the improvement.

Acknowledgments

This research is based upon work supported in part 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 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.

References

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

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 *ACL-IJCNLP*.

Yubo Chen, Shulin Liu, Xiang Zhang, Kang Liu, and Jun Zhao. 2017. Automatically labeled data generation for large scale event extraction. In *ACL*.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *ACL-IJCNLP*.

Yubo Chen, Hang Yang, Kang Liu, Jun Zhao, and Yantao Jia. 2018. Collective event detection via a hierarchical and bias tagging networks with gated multi-level attention mechanisms. In *EMNLP*.

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*, pages 151–159.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*.

Tianyu Gao, Xu Han, Zhiyuan Liu, and Maosong Sun. 2019. Hybrid attention-based prototypical networks for noisy few-shot relation classification. In *AAAI*.

Ralph Grishman, David Westbrook, and Adam Meyers. 2005. Nyu's english ace 2005 system description. In *ACE 2005 Evaluation Workshop*.

Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long short-term memory. In *Neural Computation*.

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *ACL*.

Lifu Huang, Heng Ji, Kyunghyun Cho, and Clare R Voss. 2018. Zero-shot transfer learning for event extraction. In *ACL*, pages 2160–2170.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *ACL*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. 2015. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2.

Viet Dac Lai, Franck Dernoncourt, and Thien Huu Nguyen. 2020. Exploiting the matching information in the support set for few shot event classification. In *PAKDD*.

Viet Dac Lai and Thien Huu Nguyen. 2019. Extending event detection to new types with learning from keywords. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*.

Brenden M Lake, Ruslan R Salakhutdinov, and Josh Tenenbaum. 2013. One-shot learning by inverting a compositional causal process. In *NIPS*.

Qi Li, Heng Ji, Yu Hong, and Sujian Li. 2014. Constructing information networks using one single model. In *EMNLP*.

Xiang Li, Thien Huu Nguyen, Kai Cao, and Ralph Grishman. 2015. Improving event detection with Abstract Meaning Representation. In *Proceedings of the First Workshop on Computing News Storylines*.

Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2017. Exploiting argument information to improve event detection via supervised attention mechanisms. In *ACL*.

Shulin Liu, Yang Li, Feng Zhang, Tao Yang, and Xinpeng Zhou. 2019. Event detection without triggers. In *NAACL:HLT*, pages 735–744.

Weiyi Lu and Thien Huu Nguyen. 2018. Similar but not the same: Word sense disambiguation improves event detection via neural representation matching. In *EMNLP*.

Thien Nguyen and Ralph Grishman. 2018. Graph convolutional networks with argument-aware pooling for event detection. In *AAAI*.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016a. Joint event extraction via recurrent neural networks. In *NAACL*.

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 *ACL-IJCNLP*.

Thien Huu Nguyen and Ralph Grishman. 2016. Modeling skip-grams for event detection with convolutional neural networks. In *EMNLP*.

Trung Minh Nguyen and Thien Huu Nguyen. 2018. One for all: Neural joint modeling of entities and events. In *AAAI*.

Haoruo Peng, Yangqiu Song, and Dan Roth. 2016. Event detection and co-reference with minimal supervision. In *EMNLP*.

Amir Pouran Ben Veyseh, Thien Huu Nguyen, and Dejing Dou. 2019. Graph based neural networks for event factuality prediction using syntactic and semantic structures. In *ACL*.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training gans. In *NIPS*.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. Meta-learning with memory-augmented neural networks. In *ICML*.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *NIPS*.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. 2018. Learning to compare: Relation network for few-shot learning. In *CVPR*.

Sebastian Thrun. 1996. Is learning the n-th thing any easier than learning the first? In *NIPS*.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *NIPS*.

Alex Wong and Alan L Yuille. 2015. One shot learning via compositions of meaningful patches. In *ICCV*.