Similar but not the Same: Word Sense Disambiguation Improves Event Detection via Neural Representation Matching

Weiyi Lu[†] **and Thien Huu Nguyen**[#‡]
[†] Computer Science Department, New York University, USA
[#] Montreal Institute for Learning Algorithms, University of Montreal, Canada
[‡] Department of Computer and Information Science, University of Oregon, USA
weiyi.lu@nyu.edu,thien@cs.uoregon.edu

Abstract

Event detection (ED) and word sense disambiguation (WSD) are two similar tasks in that they both involve identifying the classes (i.e. event types or word senses) of some word in a given sentence. It is thus possible to extract the knowledge hidden in the data for WSD, and utilize it to improve the performance on ED. In this work, we propose a method to transfer the knowledge learned on WSD to ED by matching the neural representations learned for the two tasks. Our experiments on two widely used datasets for ED demonstrate the effectiveness of the proposed method.

1 Introduction

An important aspect of natural language processing involves understanding events mentioned in text. Towards this end, event detection (ED) is the task of locating event triggers (usually verbs or nouns) within a given text, and classifying them among a given set of event types. This task remains challenging due to the inherent ambiguity and flexibility of natural languages. The current state-of-the-art methods for ED have involved applying deep learning (DL) models to automatically extract feature representations of the text, and then treating the task as a classification problem (Chen et al., 2015; Nguyen and Grishman, 2015b).

The major intuition in this paper is that the task of ED is closely related to the task of word sense disambiguation (WSD) whose datasets can help to improve the performance of the DL models for ED. This is due to the goal of WSD to determine the sense of a word within a particular context, given a set of possible senses that the word can take on. Our intuition is based on the two following aspects:

(i) Similar Context Modeling: Given a word in a context/sentence, both ED and WSD models need to select/predict a correct label in a list of candidate labels for the word. For WSD, the candidate labels are the possible senses (e.g, sense ids in WordNet) that the word of interest can have, while for ED, they are the set of predetermined event types (e.g, the event subtypes in the ACE 2005 dataset[1]). Consider the word "*fired*" in the following sentence as an example:

*The boss **fired** his secretary today.*

For WSD, there are 12 possible senses for the verb "*fire*" in WordNet in which the correct label for the word "*fired*" in this case is the sense id "*fire%2:41:00::*" (i.e, "*terminate the employment of*"). The ED task in the ACE 2005 dataset, on the other hand, involves 33 possible event subtypes with "*End-Position*" as the correct event subtype/label for the word "*fired*" in our example.

In order to make such label predictions, both ED and WSD need to model the word itself and its context (i.e, the words "*fired*", "*boss*", and "*secretary*" in the example). This similar modeling allows the same DL model to be adopted for both ED and WSD, facilitating the use of WSD data to improve the feature representations for ED via parameter/representation tying.

(ii) Close Semantic Consideration: As there are some overlaps between the semantic differentiation in WSD and ED, the knowledge/information from WSD about a particular word in a context can help to make a better prediction for that word in ED. For instance, in the example above, the knowledge from WSD that the word "*fired*" is referring to a termination of employment would clearly help ED to identify "*End-Position*" as the correct event type (rather than the incorrect event type "*Attack*") for "*fired*" in this case.

How can we exploit this intuition to improve the performance of the DL models for ED with WSD

[1] https://www.ldc.upenn.edu/collaborations/past-projects/ace

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4822–4828
Brussels, Belgium, October 31 - November 4, 2018.

data? In this work, we propose a novel method based on representation matching to transfer the knowledge learned from the WSD data to the DL models for ED. In particular, two separate deep learning models are employed to model the context for WSD and ED. The two models share the network architecture, but involve different parameters that are specific to the tasks. We then transfer the knowledge from the WSD network to the ED network by ensuring that the feature representations learned by the two networks on the same contexts are similar to each other.

We demonstrate the effectiveness of the proposed method on two widely used datasets for ED. To the best of our knowledge, this is the first work to study the transfer learning/multi-task learning methods for WSD and ED with DL.

2 Model

We consider the typical setting where we have two separate datasets $D^{wsd} = \{W_i^{wsd}, p_i^{wsd}, y_i^{wsd}\}$ for WSD and $D^{ed} = \{W_i^{ed}, p_i^{ed}, y_i^{ed}\}$ for ED. Here, W_i^{ed} is the i-the sentence of D^{ed}, p_i^{ed} is the index of the word of interest for event type prediction in W_i^{ed}, and y_i^{ed} is the corresponding event type label. The same conventions apply for $W_i^{wsd}, p_i^{wsd}, y_i^{wsd}$. Also, let Y^{wsd} and Y^{ed} be the label sets for WSD and ED respectively (i.e, $y_i^{wsd} \in Y^{wsd}$ and $y_i^{ed} \in Y^{ed}$). Our goal is to transfer the knowledge learned from the D^{wsd} dataset to improve the performance of the ED models trained on the D^{ed} dataset (multi-task learning).

In the following, we will first describe the deep learning architectures to transform the sentences W in the datasets D^{wsd} and D^{ed} into representation vectors. We only focus on the deep learning architectures proposed for ED in the literature to achieve compatible comparisons for ED. The proposed multi-task learning method for ED with the WSD dataset will follow.

2.1 Computing the Feature Representations

Consider a sentence W in the datasets D^{wsd} or D^{ed} that is represented as a sequence of tokens $W = [w_0, w_1, \ldots, w_t]$. Let p be the index of the word of interest in this sentence. The context for w_p in W is constructed by taking the word itself, the n preceding words, and the n following words (padding or truncating when necessary). The tokens in the context are re-indexed to form an instance $V = [v_0, v_1, \ldots, v_n, \ldots, v_{2n-1}, v_{2n}]$, where v_n corresponds to w_p in W.

Encoding

The first step to prepare the instance V for the deep learning models is to map each token v_j in V into two real-valued vectors, which are then concatenated to form a vector representation x_j for v_j (Nguyen and Grishman, 2015b; Chen et al., 2015):

1. The word embedding of v_j obtained by looking up the token v_j in the pre-trained word embedding table (Mikolov et al., 2013a).

2. The position embedding vector for v_j: obtained by looking up the relative distance $j - n$ of v_j with respect to the token of interest v_n in a position embedding table (randomly initialized) (Chen et al., 2015; Nguyen and Grishman, 2015a).

It is important to note that, different from the prior works (Nguyen and Grishman, 2015b; Liu et al., 2017), we do not include the entity type label of each token into its representation. This is a more realistic setting for our work as the golden entity mentions do not always exist in practice, especially for the datasets in WSD.

Once each token v_j is converted into the representation vector x_j, the instance V becomes a sequence of vectors $X = [x_0, x_1, \ldots, x_n, \ldots, x_{2n-1}, x_{2n}]$ that would be fed into the one of the following deep learning models to learn a feature representation R for V.

Typical Deep Learning Models for ED

1. *CNN*: This is the convolutional neural networks in(Nguyen and Grishman, 2015b; Chen et al., 2015). It features convolution operations that are performed over the k consecutive vectors (k-grams) in X and followed by a max-pooling layer to generate the representation vector R for V. Multiple window values k are used to enhance the coverage of the model over the hidden k-grams in the context.

2. *NCNN* (Nguyen and Grishman, 2016d): This model is similar to *CNN*. The only difference is instead of running the convolution over the k consecutive vectors, *NCNN* convolutes over the k arbitrarily non-consecutive k vectors in V. This helps *NCNN* to explicitly model the non-consecutive words in the context to improve ED.

3. *BiRNN*: This is the bidirectional recurrent neural network (RNN) for event extraction in (Nguyen et al., 2016a). The model is

composed of two recurrent neural networks (RNN), where one runs forward and the other runs backward through the input sequence V. The hidden vectors produced by the two networks are then concatenated at each position in the context. The vector at the position of n for the word of interest is used as the representation vector R for V. Due to the property of RNN, R encodes the information over the whole input V with a greater focus on v_n.

4. *CNN+BiRNN*: In this model (Feng et al., 2016), X is passed through both a *CNN* and a *BiRNN* whose results are concatenated to produce the hidden representation R for ED. The expectation is to take advantage of the modeling abilities from both the *CNN* and *BiRNN* architectures for ED.

In practice, the representation vector R (obtained from one of the deep learning models above) is also concatenated with the word embeddings of the tokens surrounding the token of interest w_n to improve its expressiveness (Chen et al., 2015; Nguyen and Grishman, 2016d). We would use this extended version when we refer to R in the following.

In the final step, the representation vector R is fed into a feed-forward neural network followed by a softmax layer to perform predictions for ED and WSD.

For convenience, we denote the whole process that a DL model M is used to compute the representation vector R for the input sentence W with the token index p of interest as: $R = M(W, p)$.

2.2 Multi-task Learning Models

The previous section has described the deep learning methods that can be employed to train the models for ED and WSD separately. This section presents our proposed method to transfer the knowledge from the WSD dataset to improve the performance for ED.

A typical method for transfer learning/multi-task learning in NLP is to alternate the training process for the parameter-shared models of the related tasks (possibly with different datasets) (Guo et al., 2016; Li et al., 2015; Liu et al., 2016). For instance, in (Guo et al., 2016), the authors use the same deep learning model to learn the feature representations for the text inputs of two related tasks. This is then followed by task-specific output layers to perform the corresponding tasks. Note that the two tasks in (Guo et al., 2016) are provided with two different datasets of different text inputs, thereby being similar to the setting we consider in this work. In order to learn the parameters for this model, in each iteration, (Guo et al., 2016) select one of the tasks with some probabilities, sample a mini-batch of examples in the dataset of the chosen task, and update the model parameters using the objective function specific to the chosen task. Consequently, the model parameters for feature representation learning are updated at every iteration while only the model parameters in the output layer for the chosen task are updated at the current iteration.

It has been demonstrated in (Guo et al., 2016) that the alternating method (called *ALT*) is more effective than pre-training the network on a related task and fine-tuning it on the expected task. We thereby consider *ALT* as the baseline for multi-task learning in our work. However, we argue that this baseline is not effective enough to transfer the knowledge from the WSD dataset to ED in our case. This stems from its employment of a single DL model to induce the representations for the text inputs in both tasks. In our case of WSD and ED, although there are some overlap between the semantic differentiation of the two tasks, the labels in the WSD datasets (i.e, the sense ids) tend to be more fine-grained and exhaustive than those in ED. For instance, for the word "*fire*", there might be 12 WSD labels for it in WordNet while the number of possible event types for "*fire*" in the ACE 2005 dataset is only 2 (i.e, "*End-Position*" and "*Attack*"). Eventually, if a single DL model is used to compute the representations for the text inputs in both WSD and ED, the model would suffer from a confusion to distinguish such subtlety in the semantic differentiation.

In order to overcome this issue, we propose to employ two versions M^{wsd} and M^{ed} of the same DL model (with different model parameters) to compute the feature representations for WSD and ED respectively. We then transfer the knowledge from M^{wsd} to M^{ed} by encouraging the representations generated by the two versions M^{wsd} and M^{ed} on the same text inputs to be similar. Formally, let (W^t, p^t, y^t) be an example in the D^{wsd} or D^{ed} dataset ($t \in \{wsd, ed\}$). Also, let R^{wsd} and R^{ed} be the representations for (W^t, p^t) induced by M^{wsd} and M^{ed} respectively:

$$R^{wsd} = M^{wsd}(W^t, p^t), R^{ed} = M^{ed}(W^t, p^t)$$

Such representation vectors are then followed by a task-specific output layer F^t (i.e, feed-forward neural networks followed by a softmax layer) to compute the probability distribution over the possible labels for (W^t, p^t): $P^t(Y^t|R^t) = F^t(R^t)$ where Y^t is the label set for the t task.

If the two models M^{wsd} and M^{ed} were trained separately, the objective function for the t task for the current example would be the negative log-likelihood: $C^t(W^t, p^t, y^t) = -\log P^t(y^t|R^t)$. In this work, instead of just optimizing this objective, we optimize the joint function:

$$C^t(W^t, p^t, y^t) = -\log P^t(y^t|R^t) + \lambda \frac{1}{d_R} \sum_{i=0}^{d_R} \left(R_i^{wsd} - R_i^{ed}\right)^2$$

where λ is a trade-off parameter and d_R is the dimension of the representation vectors.

The second term in the joint objective function enforces that the feature representations learned by M^{wsd} and M^{ed} on the same input context (W^t, p^t) are close to each other ($t \in \{wsd, ed\}$). One the one hand, this representation matching schema helps the two models to communicate to each other so the knowledge from one model can be passed to the other one. On the other hand, the use of two separate models leaves a flexibility for the models to induce the task-specific structures.

Presumably, the objective function (2.2) can simultaneously improve the performance for both tasks of consideration. However, in our case of ED and WSD, it turns out this mechanism actually worsen the performance of the WSD models that were trained separately. We attribute this to the fact that the semantic differentiation in ED is more coarse-grained that that of WSD, causing the ineffectiveness of the datasets for ED to improve WSD performance. Eventually, we will just focus on the ED performance in the experiments.

3 Experiments

3.1 Parameters and Datasets

We use the Semcor dataset (Miller et al., 1994) as the dataset for WSD in this work. This dataset was extracted from the Brown Corpus, and manually annotated with WordNet senses. We evaluate the models on two different datasets for ED:

1. **ACE 2005**: This dataset has 33 event subtypes. We use the same data split with the prior work (Chen et al., 2015; Nguyen and Grishman, 2015b). In particular, 40 newswire documents are used for testing, 30 other documents are reserved for validation, and the 529 remaining documents form the training data.

2. **TAC 2015**: This dataset was released in the Event Nugget Detection Evaluation of the 2015 Text Analysis Conference (TAC) (Mitamura et al., 2015). It comes with 38 event subtypes. We follow the data split in the official evaluation to achieve compatible comparison. As TAC 2015 does not have a development set, we use the best parameters tuned on ACE 2005 for the experiments with TAC 2015.

We use the pre-trained word embeddings provided by (Nguyen and Grishman, 2016d). For *CNN*, *NCNN* and *CNN+BiRNN*, we employ filter sizes of $\{2, 3, 4, 5\}$ with 300 filters for each size as in (Nguyen and Grishman, 2015b), while Gated Recurrent Units (Cho et al., 2014) with 300 hidden units are applied in *BiRNN* and *CNN+BiRNN* (as do (Nguyen and Grishman, 2016d)). For the other parameters, the best values suggested by the development data include: a dropout rate of 0.5, a feed-forward neural network with one hidden layer of 1200 hidden units for the output layers, and the penalty rate λ of 0.01 for both CNN and BiRNN, 0.6 for NCNN, and 0.7 for CNN+BiRNN in the proposed transfer learning method (called *MATCHING*). For simplicity, the same hyper-parameters are used for the two versions of the same network architecture in the *MATCHING* method. We utilize Adadelta (Zeiler, 2012) with back-propagation to train the models in this work.

3.2 Experiments

In this section, we compare the proposed *MATCHING* method with the transfer learning baseline *ALT* in (Guo et al., 2016) and the separate training mechanism for ED (called *SEPARATE*) employed in the previous work for ED (Chen et al., 2015; Nguyen and Grishman, 2015b). Note that in the *SEPARATE* method, the models are only trained on the datasets for ED without utilizing any transfer learning techniques with external datasets. We report the performance when each of the DL methods in Section 2.1 is used as the network to learn the feature representations for ED and WSD.

Tables 1 and 2 present the performance (i.e, F1 scores) of the models on the ACE 2005 and TAC 2015 datasets respectively. The first observation is that the proposed transfer learning method *MATCHING* is consistently better than the baseline method *ALT* across different deep learning models and datasets with large performance gap. This is significantly with $p < 0.05$ and confirms our hypothesis in Section 2.2 about the advantage of the proposed *MATCHING* over the alternating training method *ALT* for ED and WSD. In fact, the performance of the *ALT* method is even worse than the traditional *SEPARATE* method also over different network architectures and datasets. Consequently, training a single deep learning model on a combination of ED and WSD data (as in *ALT*) does not automatically enable the model to learn to exploit the similar structures of the two tasks. In contrast, it hinders the model's ability to effectively extract hidden representations for ED.

Comparing *MATCHING* and *SEPARATE*, we see that *MATCHING* helps to improve *SEPARATE* with respect to difference choices of the DL models. The performance improvement is significant for *CNN* and *BiRNN* on ACE 2005 and for all the models on TAC 2015. Such results demonstrate the effectiveness of the WSD dataset for ED and the ability of the proposed method *MATCHING* to promote knowledge transferring between WSD and ED to improve ED performance.

Regarding the best reported performance, our best performance on ACE (i.e, 71.2% with *CNN*) is comparable with the recent state-of-the-art performance (i.e, Table 1). However, we note that such work heavily relies on the manual annotation of the entity mentions in the documents. Our current work do not employ such information to better reflect the realistic setting. For the TAC 2015 dataset, our best performance is 60.7% with *CNN+BiRNN* although the performance of the other models is also very close. This performance is better than the best performance that has been reported on the TAC 2015 (i.e, Table 2).

Method	CNN	BiRNN	NCNN	CNN+BiRNN
SEPARATE	67.6	67.6	69.3	68.1
ALT	65.1	66.4	65.0	65.2
MATCHING	**71.2**	**69.0**	**69.6**	**68.3**
(Nguyen and Grishman, 2016d)			71.3*	
(Liu et al., 2017)			71.9*	
(Liu et al., 2018)			72.4*	
(Nguyen and Grishman, 2018a)			73.1*	

Table 1: Performance on the ACE 2005 dataset. * indicates the use of entity mention annotation.

Method	CNN	BiRNN	NCNN	CNN+BiRNN
SEPARATE	57.6	59.4	58.3	58.0
ALT	57.6	54.9	48.5	57.5
MATCHING	**60.0**	**60.4**	**60.0**	**60.7**
TAC TOP (Mitamura et al., 2015)			58.4*	
(Nguyen and Grishman, 2018a)			58.8*	

Table 2: Performance on the TAC 2015 dataset. * indicates the use of entity mention annotation.

4 Related Work

Prior works on ED include statistical models with manual feature engineering(Ahn, 2006; Ji and Grishman, 2008; Hong et al., 2011; Li et al., 2013; Venugopal et al., 2014; Li et al., 2015), followed by neural network models, such as CNNs (Nguyen and Grishman, 2015b; Chen et al., 2015; Nguyen et al., 2016b,e; Chen et al., 2017), RNNs (Nguyen et al., 2016a; Jagannatha and Yu, 2016), and attention-based methods (Liu et al., 2017; Nguyen and Nguyen, 2018b).

A similar trend exists in methods proposed for WSD, with feature based methods (Miller et al., 1994; Zhong and Ng, 2010; Taghipour and Ng, 2015) succeeded recently by deep learning methods (Yuan et al., 2016; Raganato et al., 2017).

For multi-task learning in NLP, methods have been proposed for jointly modeling structured prediction tasks (Hatori et al., 2012; Li et al., 2011; Bohnet and Nivre, 2012; Henderson et al., 2013; Lluís et al., 2013; Duong et al., 2015), and for sequence-to-sequence problems (Dong et al., 2015; Luong et al., 2015; Liu et al., 2016; Klerke et al., 2016). The prior work to solve multiple NLP tasks using an unified architecture includes (Collobert and Weston, 2008; Guo et al., 2016).

5 Conclusion

We present a method that improves the performance of deep learning models for ED by training two different versions of the same network architecture for ED and WSD, while encouraging the knowledge transfer between the two versions via representation matching. The proposed method produces better results across a variety of deep learning models.

References

David Ahn. 2006. The stages of event extraction. In *ACL*.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *EMNLP-CoNLL*.

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*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *ICML*.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *ACL-IJCNLP*.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *ACL*.

Xiaocheng Feng, Lifu Huang, Duyu Tang, Heng Ji, Bing Qin, and Ting Liu. 2016. A language-independent neural network for event detection. In *ACL*.

Jiang Guo, Wanxiang Che, Haifeng Wang, Ting Liu, and Jun Xu. 2016. A unified architecture for semantic role labeling and relation classification. In *COLING*.

Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2012. Incremental joint approach to word segmentation, pos tagging, and dependency parsing in chinese. In *ACL*.

James Henderson, Paola Merlo, Ivan Titov, and Gabriele Musillo. 2013. Multilingual joint parsing of syntactic and semantic dependencies with a latent variable model. *Computational linguistics*, 39(4):949–998.

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

Abhyuday N Jagannatha and Hong Yu. 2016. Bidirectional rnn for medical event detection in electronic health records. In *NAACL*.

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

Sigrid Klerke, Yoav Goldberg, and Anders Søgaard. 2016. Improving sentence compression by learning to predict gaze. *arXiv preprint arXiv:1604.03357*.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *ACL*.

Xiang Li, Thien Huu Nguyen, Kai Cao, and Ralph Grishman. 2015. Improving event detection with abstract meaning representation. In *Proceedings of ACL-IJCNLP Workshop on Computing News Storylines (CNewS)*.

Zhenghua Li, Min Zhang, Wanxiang Che, Ting Liu, Wenliang Chen, and Haizhou Li. 2011. Joint models for chinese pos tagging and dependency parsing. In *EMNLP*.

Jian Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2018. Event detection via gated multilingual attention mechanism. In *AAAI*.

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

Yang Liu, Sujian Li, Xiaodong Zhang, and Zhifang Sui. 2016. Implicit discourse relation classification via multi-task neural networks. In *AAAI*.

Xavier Lluís, Xavier Carreras, and Lluís Màrquez. 2013. Joint arc-factored parsing of syntactic and semantic dependencies. *ACL*.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *ICLR*.

George A Miller, Martin Chodorow, Shari Landes, Claudia Leacock, and Robert G Thomas. 1994. Using a semantic concordance for sense identification. In *Proceedings of the workshop on Human Language Technology*. Association for Computational Linguistics.

Teruko Mitamura, Zhengzhong Liu, and Eduard Hovy. 2015. Overview of tac kbp 2015 event nugget track. In *TAC*.

Minh Nguyen and Thien Huu Nguyen. 2018b. Who is killed by police: Introducing supervised attention for hierarchical lstms. In *COLING*.

Thien Huu Nguyen, , Adam Meyers, and Ralph Grishman. 2016e. New york university 2016 system for kbp event nugget: A deep learning approach. In *TAC*.

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. 2015a. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of the 1st NAACL Workshop on Vector Space Modeling for NLP (VSM)*.

Thien Huu Nguyen and Ralph Grishman. 2015b. Event detection and domain adaptation with convolutional neural networks. In *ACL-IJCNLP*.

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

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

Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. 2017. Neural sequence learning models for word sense disambiguation. In *EMNLP*.

Kaveh Taghipour and Hwee Tou Ng. 2015. One million sense-tagged instances for word sense disambiguation and induction. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*.

Deepak Venugopal, Chen Chen, Vibhav Gogate, and Vincent Ng. 2014. Relieving the computational bottleneck: Joint inference for event extraction with high-dimensional features. In *EMNLP*.

Dayu Yuan, Julian Richardson, Ryan Doherty, Colin Evans, and Eric Altendorf. 2016. Semi-supervised word sense disambiguation with neural models. *arXiv preprint arXiv:1603.07012*.

Matthew D. Zeiler. 2012. Adadelta: An adaptive learning rate method. In *CoRR, abs/1212.5701*.

Zhi Zhong and Hwee Tou Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. In *ACL*.