Event Detection with Neural Networks: A Rigorous Empirical Evaluation

J. Walker Orr *
George Fox University
414 N Meridian St.
Newberg, OR 97132
jorr@georgefox.edu

Prasad Tadepalli and **Xiaoli Fern**
Oregon State University
Corvallis OR 97331
tadepall@eecs.oregonstate.edu,
xfern@eecs.oregonstate.edu

Abstract

Detecting events and classifying them into predefined types is an important step in knowledge extraction from natural language texts. While the neural network models have generally led the state-of-the-art, the differences in performance between different architectures have not been rigorously studied. In this paper we present a novel GRU-based model that combines syntactic information along with temporal structure through an attention mechanism. We show that it is competitive with other neural network architectures through empirical evaluations under different random initializations and training-validation-test splits of ACE2005 dataset.

1 Introduction

Events are the lingua franca of news stories and narratives and describe important changes of state in the world. Identifying events and classifying them into different types is a challenging aspect of understanding text. This paper focuses on the task of *event detection*, which includes identifying the "trigger" words that indicate events and classifying the events into refined types. Event detection is the necessary first step in inferring more semantic information about the events including extracting the arguments of events and recognizing temporal and causal relationships between different events.

Neural network models have been the most successful methods for event detection. However, most current models ignore the syntactic relationships in the text. One of the main contributions of our work is a new DAG-GRU architecture (Chung et al., 2014) that captures the context and syntactic information through a bidirectional reading of the text with dependency parse relationships. This generalizes the GRU model to operate on a graph by novel use of an attention mechanism.

[1]Also associated with Oregon State University.

Following the long history of prior work on event detection, ACE2005 is used for the precise definition of the task and the data for the purposes of evaluation. One of the challenges of the task is the size and sparsity of this dataset. It consists of 599 documents, which are broken into a training, development, testing split of 529, 30, and 40 respectively. This split has become a de-facto evaluation standard since (Li et al., 2013). Furthermore, the test set is small and consists only of newswire documents, when there are multiple domains within ACE2005. These two factors lead to a significant difference between the training and testing event type distribution. Though some work had been done comparing method across domains (Nguyen and Grishman, 2015), variations in the training/test split including all the domains has not been studied. We evaluate the sensitivity of model accuracy to changes in training and test set splits through a randomized study.

Given the limited amount of training data in comparison to other datasets used by neural network models, and the narrow margin between many high performance methods, the effect of the initialization of these methods needs to be considered. In this paper, we conduct an empirical study of the sensitivity of the system performance to the model initialization.

Results show that our DAG-GRU method is competitive with other state-of-the-art methods. However, the performance of all methods is more sensitive to the random model initialization than expected. Importantly, the ranking of different methods based on the performance on the standard training-validation-test split is sometimes different from the ranking based on the average over multiple splits, suggesting that the community should move away from single split evaluations.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 999–1004
Brussels, Belgium, October 31 - November 4, 2018.

2 Related Work

Event detection and extraction are well-studied tasks with a long history of research.

Nguyen and Grishman (2015) used CNNs to represent windows around candidate triggers. Each word is represented by a concatenation of its word and entity type embeddings with the distance to candidate trigger. Global max-pooling summarizes the CNN filter and the result is passed to a linear classifier.

Nguyen and Grishman (2016) followed up with a skip-gram based CNN model which allows the filter to skip non-salient or otherwise unnecessary words in the middle of word sequences.

Feng et al. (2016) combined a CNN, similar to (Nguyen and Grishman, 2015), with a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) to create a hybrid network. The output of both networks was concatenated together and fed to a linear model for final predictions.

Nguyen et al. (2016) uses a bidirectional gated recurrent units (GRUs) for sentence level encoding, and in conjunction with a memory network, to jointly predict events and their arguments.

Liu et al. (2016) created a probablistic soft logic model incorporating the semantic frames from Framenet (Baker et al., 1998) in the form of extra training examples. Based on the intuition that entity and argument information is important for event detection, Liu et al. (2017) built an attention model over annotated arguments and the context surrounding event trigger candidates.

Liu et al. (2018) created a cross language attention model for event detection and used it for event detection in both the English and Chinese portions of the ACE2005 data.

Recently, Nguyen and Grishman (2018) used graph-CNN (GCCN) where the convolutional filters are applied to syntactically dependent words in addition to consecutive words. The addition of the entity information into the network structure produced the state-of-the-art CNN model.

Another neural network model that includes syntactic dependency relationships is DAG-based LSTM (Qian et al., 2018). It combines the syntactic hidden vectors by weighted average and adds them through a dependency gate to the output gate of the LSTM model. To the best of our knowledge, none of the neural models combine syntactic information with attention, which motivates our research.

3 DAG GRU Model

Event detection is often framed as a multi-class classification problem (Chen et al., 2015; Ghaeini et al., 2016). The task is to predict the event label for each word in the test documents, and *NIL* if the word is not an event. A sentence is a sequence of words $x_1 \dots x_n$, where each word is represented by a k-length vector. The standard GRU model works as follows:

$$
\begin{aligned}
r_t &= \sigma(W_r x_t + U_r h_{t-1} + b_r) \\
z_t &= \sigma(W_z x_t + U_z h_{t-1} + b_z) \\
\tilde{h}_t &= \tanh(W_h x_t + r_t \odot U_h h_{t-1} + b_h) \\
h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t
\end{aligned}
$$

The GRU model produces a hidden vector h_t for each word x_t by combining its representation with the previous hidden vector. Thus h_t summarizes both the word and its prior context.

Our proposed DAG-GRU model incorporates syntactic information through dependency parse relationships and is similar in spirit to (Nguyen and Grishman, 2018) and (Qian et al., 2018). However, unlike those methods, DAG-GRU uses attention to combine syntactic and temporal information. Rather than using an additional gate as in (Qian et al., 2018), DAG-GRU creates a single combined representation over previous hidden vectors and then applies the standard GRU model. Each relationship is represented as an edge, (t, t', e), between words at index t and t' with an edge type e. The standard GRU edges are included as $(t, t-1, temporal)$.

Each dependency relationship may be between any two words, which could produce a graph with cycles. However, back-propagation through time (Mozer, 1995) requires a directed acyclic graph (DAG), Hence the sentence graph, consisting of temporal and dependency edges E, is split into two DAGs: a "forward" DAG G_f that consists of only of edges (t, t', e) where $t' < t$ and a corresponding "backward" DAG G_b where $t' > t$. The dependency relation between t and t' also includes the parent-child orientation, e.g., $nsubj\text{-}parent$ or $nsubj\text{-}child$ for a $nsubj$ (subject) relation.

An attention mechanism is used to combine the multiple hidden vectors. The matrix D_t is formed at each word x_t by collecting and transforming all the previous hidden vectors coming into node t, one per each edge type e. α gives the attention, a distribution weighting importance over the edges.

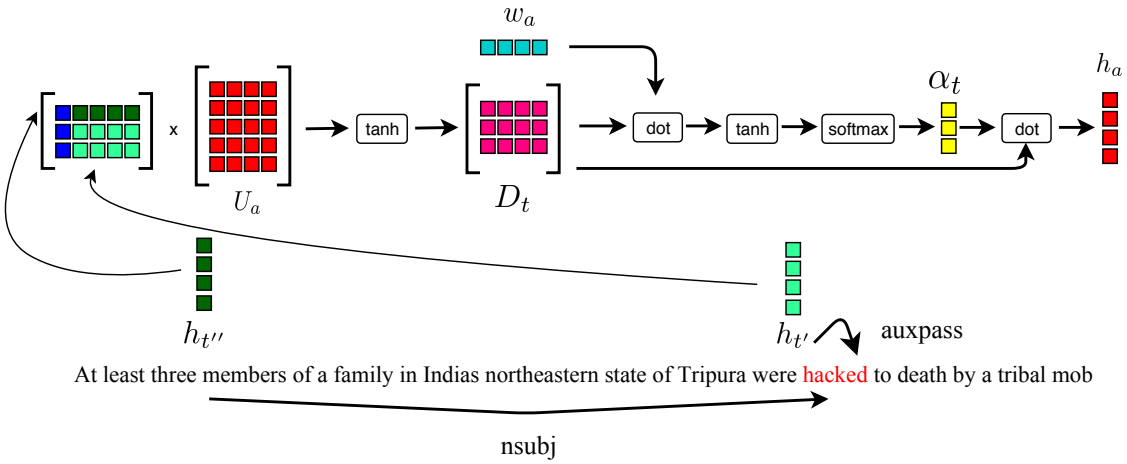

Figure 1: The hidden state of "hacked" is a combination of previous output vectors. In this case, three vectors are aggregated with DAG-GRU's attention model. $h_{t''}$, is included in the input for the attention model since it is accessible through the "subj" dependency edge. $h_{t'}$ is included twice because it is connected through a narrative edge and a dependency edge with type "auxpass." The input matrix is non-linearly transformed by U_a and tanh. Next, w_a determines the importance of each vector in D_t. Finally, the attention a_t is produced by tanh followed by softmax then applied to D_t. The subject "members" would be distant under a standard RNN model, however the DAG-GRU model can focus on this important connection via dependency edges and attention.

Finally, the combined hidden vector h_a is created by summing D_t weighted by attention.

$$D_t^T = [\tanh(U_e h_{t'}) | (t, t', e) \in E]$$
$$\alpha = \text{softmax}(\tanh(D_t w_a))$$
$$h_a = D_t^T \alpha$$

However, having a set of parameters U_e for each edge type e is over-specific for small datasets. Instead a shared set of parameters U_a is used in conjunction with an edge embedding v_e.

$$D_t^T = [\tanh(U_a h_{t'} \circ v_e) | (t, t', e) \in E]$$

The edge type embedding v_e is concatenated with the hidden vector $h_{t'}$ and then transformed by the shared weights U_a. This limits the number of parameters while flexibly weighting the different edge types. The new combined hidden vector h_a is used instead of h_{t-1} in the GRU equations.

The model is run forward and backward with the output concatenated, $h_{c,t} = h_{f,t} \odot h_{b,t}$, for a representation that includes the entire sentence's context and dependency relations. After applying dropout (Srivastava et al., 2014) with 0.5 rate to $h_{c,t}$, a linear model with softmax is used to make predictions for each word at index t.

4 Experiments

We use the ACE2005 dataset for evaluation. Each word in each document is marked with one of the thirty-three event types or *Nil* for non-triggers. Several high-performance models were reproduced for comparison. Each is a good faith reproduction of the original with some adjustments to level the playing field.

For word embeddings, Elmo was used to generate a fixed representation for every word in ACE2005 (Peters et al., 2018). The three vectors produced per word were concatenated together for a single representation. We did not use entity type embeddings for any method. The models were trained to minimize the cross entropy loss with Adam (Kingma and Ba, 2014) with L2 regularization set at 0.0001. The learning rate was halved every five epochs starting from 0.0005 for a maximum of 30 epochs or until convergence as determined by F1 score on the development set.

The same training method and word embeddings were used across all the methods. Based on preliminary experiments, these settings resulted in better performance than those originally specified. However, notably both GRU (Nguyen et al., 2016) and DAG-LSTM (Qian et al., 2018) were not used as joint models. Further, the GRU implementation did not use a memory network, instead we used the final vectors from the forward and backward pass concatenated to each timestep's output for additional context. For CNNs (Nguyen and Grishman, 2015) the number of filters was reduced to 50 per filter size. The CNN+LSTM (Feng et al.,

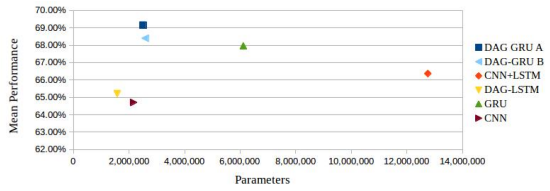

Figure 2: A comparison of mean performance versus number of parameters.

2016) model had no other modifications. DAG-GRU used a hidden layer size of 128. Variant A of the DAG-GRU model utilized the attention mechanism, while variant B used averaging, that is:

$$D_t = \frac{1}{|E(t)|} \sum_{(t',e) \in E(t)} \tanh(U_a h_{t'} \circ v_e)$$

4.1 Effects of Random Initialization

Given that ACE2005 is small as far as neural network models are concerned, the effect of the random initialization of these models needs to be studied. Although some methods include tests of significance, the type of statistical test is often not reported. Simple statistical significance tests, such as the t-test, are not compatible with a single F1 score, instead the average of F1 scores should be tested (Wang et al., 2015).

We reproduced and evaluated five different systems with different initializations to empirically assess the effect of initialization. The experiments were done on the standard ACE2005 split and the aggregated results over 20 random seeds were given in Table 1. The random initializations of the models had a significant impact on their performance. The variation was large enough that the observed range of the F1 scores overlapped across almost all the models. However the differences in average performances of different methods, except for CNN and DAG-LSTM, were significant at $p < 0.05$ according to the t-test, not controlling for multiple hypotheses.

Both the GRU (Nguyen et al., 2016) and CNN (Nguyen and Grishman, 2015) models perform well with their best scores being close to the reported values. The CNN+LSTM model's results were significantly lower than the published values, though this method has the highest variation. It is possible that there is some unknown factor such as the preprocessing of the data that significantly impacted the results or that the value is an outlier. Likewise, the DAG-LSTM model underperformed. However, the published results were based on a joint event and argument extraction model and probably benefited from the additional entity and argument information.

DAG-GRU A consistently and significantly outperforms the other methods in this comparison. The best observed F1 score, 71.1%, for DAG-GRU is close to the published state-of-the-art scores of DAG-LSTM and GCNN at 71.9% and 71.4% respectively. With additional entity information, GCNN achieves a score of 73.1%. Also, the attention mechanism used in DAG-GRU A shows a significant improvement over the averaging method of DAG-GRU B. This indicates that some syntactic links are more useful than others and that the weighting attention applies is necessary to utilize that syntactic information.

Another source of variation was the distributional differences between the development and testing sets. Further, the testing set only include newswire articles whereas the training and dev. sets contain informal writing such as web log (WL) documents. The two sets have different proportions of event types and each model saw at least a 2% drop in performance between dev. and test on average. At worst, the DAG-LSTM model's drop was 5.26%. This is a problem for model selection, since the dev. score is used to choose the best model, hyperparameters, or random initialization. The distributional differences mean that methods which outperform others on the dev. set do not necessarily perform as well on the test set. For example, DAG-GRU A performs worse that DAG-GRU B on the dev. set, however it achieves a higher mean score on the testing set.

One method of model selection over random initializations is to train the model k times and pick the best one based on the dev. score. Repeating this model selection procedure many times for each model is prohibitively expensive, so the experiment was approximated by bootstrapping the twenty samples per model (Efron, 1992). For each model, 5 dev. & test score pairs were sampled with replacement from the twenty available pairs. The initialization with the best dev. score was selected and the corresponding test score was taken. This model selection process of picking the best of 5 random samples was repeated 1000 times and the results are shown in Table 2. This process did not substantially increase the average performance beyond the results in Table 1, although it did reduce the variance, except for the CNN model. It ap-

Model	Dev Mean	Mean	Min	Max	Std. Dev.	Published
DAG-GRU A	70.3%	69.2% ± 0.42	67.8%	71.1%	0.91%	-
DAG-GRU B	71.2%	68.4% ± 0.45	67.39%	70.53%	0.96%	-
GRU	70.3%	68.0% ± 0.42	66.4%	69.4%	0.86%	69.3%†
CNN+LSTM	69.6%	66.4% ± 0.62	63.6%	68.1%	1.32%	73.4%
DAG-LSTM	70.5%	65.2% ± 0.44	63.0%	66.8%	0.94%	71.9%†
CNN	68.5%	64.7% ± 0.65	61.6%	67.2%	1.38%	67.6%

Table 1: The statistics of the 20 random initializations experiment. † denotes results are from a joint event and argument extraction model.

Model	Dev Mean	Mean	Std. Dev.
DAG-GRU A	72.0%	69.2% ± 0.04%	0.68%
DAG-GRU B	72.0%	67.9% ± 0.04%	0.60%
GRU	71.5%	68.4% ± 0.05%	0.80%
CNN+LSTM	70.8%	66.8% ± 0.07%	1.08%
DAG-LSTM	70.4%	65.5% ± 0.02%	0.40%
CNN	69.6%	65.4% ± 0.09%	1.49%

Table 2: Bootstrap estimates on 1000 samples for each model after model selection based on dev set scores.

pears that using the dev. score for model selection is only marginally helpful.

4.2 Randomized Splits

In order to explore the effect of the training/testing split popularized by (Li et al., 2013), a randomized cross validation experiment was conducted. From the set of 599 documents in ACE2005, 10 random splits were created maintaining the same 529, 30, 40 document counts per split, training, development, testing, respectively. This method was used to evaluate the effect of the standard split, since it maintains the same data proportions while only varying the split. The results of the experiment are found in Table 3.

The effect of the split is substantial. Almost all models' performance dropped except for DAG-LSTM, however the variance increased across all models. In the worst case, the standard deviation increased threefold from 0.86% to 2.60% for the GRU model. In fact, the increased variation of the splits means that the confidence intervals for all the models overlap. This aligns with cross domain analysis, some domains such as WL are known to be much more difficult than the newswire domain which comprises all of the test data under the standard split (Nguyen and Grishman, 2015). Further, the effect of the difference in splits also negates the benefits of the attention mechanism of DAG-GRU A. This is likely due to the test partitions' inclusion of WL and other kinds of informal writing. The syntactic links are much more likely to be noisy for informal writing, reducing the syntactic information's usefulness and reliability.

Method	Dev Mean	Mean	Min	Max	Std. Dev.
DAG-GRU A	71.4%	68.4% ± 1.85%	65.7%	74.1%	2.59%
DAG-GRU B	70.9%	68.4% ± 1.88%	64.19%	73.59	2.63%
DAG-LSTM	68.9%	67.3% ± 1.43%	63.5%	70.7%	2.00%
GRU	69.8%	66.6% ± 1.86%	62.5%	71.1%	2.60%
CNN+LSTM	69.8%	66.3% ± 2.03%	60.1%	70.3%	2.83%
CNN	68.0%	65.4% ± 1.59%	60.7%	69.2%	2.22%

Table 3: Average results on 10 randomized splits.

All these sources of variation are greater than most advances in event detection, so quantifying and reporting this variation is essential when assessing model performance. Further, understanding this variation is important for reproducibility and is necessary for making any valid claims about a model's relative effectiveness.

5 Conclusions

We introduced and evaluated a DAG-GRU model along with four previous models in two different settings, the standard ACE2005 split with multiple random initializations and the same dataset with multiple random splits. These experiments demonstrate that our model, which utilizes syntactic information through an attention mechanism, is competitive with the state-of-the-art. Further, they show that there are several significant sources of variation which had not been previously studied and quantified. Studying and mitigating this variation could be of significant value by itself. At a minimum, it suggests that the community should move away from evaluations based on single random initializations and single training-test splits.

Acknowledgments

This work was supported by grants from DARPA XAI (N66001-17-2-4030), DEFT (FA8750-13-2-0033), and NSF (IIS-1619433 and IIS-1406049).

References

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *ACL (1)*, pages 167–176.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Bradley Efron. 1992. Bootstrap methods: another look at the jackknife. In *Breakthroughs in statistics*, pages 569–593. Springer.

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 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 66–71.

Reza Ghaeini, Xiaoli Z Fern, Liang Huang, and Prasad Tadepalli. 2016. Event nugget detection with forward-backward recurrent neural networks. In *The 54th Annual Meeting of the Association for Computational Linguistics*, page 369.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *ACL (1)*, pages 73–82.

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

Shulin Liu, Yubo Chen, Shizhu He, Kang Liu, and Jun Zhao. 2016. Leveraging framenet to improve automatic event detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2134–2143.

Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2017. Exploiting argument information to improve event detection via supervised attention mechanisms. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1789–1798.

Michael C Mozer. 1995. A focused backpropagation algorithm for temporal. *Backpropagation: Theory, architectures, and applications*, page 137.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of NAACL-HLT*, pages 300–309.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *ACL (2)*, pages 365–371.

Thien Huu Nguyen and Ralph Grishman. 2016. Modeling skip-grams for event detection with convolutional neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 886–891.

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

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Feng Qian, Lei Sha, Baobao Chang, and Zhifang Sui. 2018. Jointly extracting event triggers and arguments by dependency-bridge rnn and tensor-based argument interaction. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Yu Wang, Jihong Li, Ruibo Wang, and Wingli Yang. 2015. Confidence interval for f1 measure of algorithm performance based on blocked 32 cross-validation. volume 27, pages 651–659.