Structurally-Informed Transformer

Zhengbao Jiang zhengbaj@andrew.cmu.edu
Libing Zhang libingz@andrew.cmu.edu
Runyu Xiao rxiao@andrew.cmu.edu

Abstract

Transformer [1] has been widely used in sequence modeling tasks, such as language modeling and machine translation, and proves to be superior than recurrent neural network on modeling long term dependency because every pairs of tokens are connected using self-attention to mitigate gradient vanishing or exploding problem [2]. However, the attentions are trained from scratch, not aware of the intrinsic structure of the sequence. In this paper, we propose explicitly exposing structural information to Transformer through relative positional encoding, which informs the model about the relations between the objects in the sequence. We use dependency structure as a case to investigate how useful it is for open information extraction task. Experiments demonstrate the effectiveness of the structural information.

1 Introduction

Transformer [1] has recently been proposed to learn representations for a set of objects (e.g., words in natural language sentence) solely based on attention mechanism. It achieves state-of-the-art performance for a wide range of tasks, such as language modeling [3, 4] and machine translation [1]. Instead of using recurrent computations, Transformer directly connects distant object pairs using attention mechanism, which circumvents gradient vanishing and explosion problem [2] and has the potential to model long-term dependency.

Despite its strong capacity, Transformer is currently structure-agnostic, with positional encoding [1, 3, 4] only considering the order of input. However, most real-word data exhibits structures more complex than linear chain structure, such as the syntactic structure [5] of a sentences and relational structure [6] of a graph. A natural question is: can we leverage structure information within Transformer to learn better representation? In this paper, we work towards incorporating prior knowledge of the structure of the input into Transformer and compare its performance with structure-agnostic sequential models on open information extraction (IE), a task similar to semantic role labeling (SRL) that aims to determine predicate-argument structure of a given sentence, such as "who did what to whom", "when", and "where." [7, 8] In particular, comparing to the explicit manner, where the attention in Transformer is constrained to attend to specific tokens according to the structure prior [9], we propose an implicit manner, where the relations between any two tokens is represented using relative positional encodings and are taken by Transformer as inputs. Experiments on open IE datasets demonstrate the effectiveness of our model.

2 Related Works

2.1 Neural Networks with Structural Inductive Bias

A lot of neural networks take the structure of the inputs into consideration when designing the architecture. A notable example in computer vision is deformable ConvNets [10], where the convolution and pooling are augmented with spatial offsets to better model geometric transformations. It replaces standard grid locations with movable locations to implicitly detect regions in the feature maps. In natural language processing, where recurrent neural networks are heavily used, hierarchical and tree

10707 Class Project (2019).

structures are successfully incorporated to allow the model to go beyond the simple linear chain structure. [11] proposes hierarchical multiscale recurrent neural network to automatically discover the latent units in the token sequence, such as phrases and clauses. [5] proposes tree-structured LSTM, where the recurrent transformation is not conducted along the token sequence, but instead following the sentence's parse tree. Superior performance is achieved by enabling the model to be structure-aware.

2.2 Transformer

Recurrent neural networks, long short-term memory and gated recurrent neural networks in particular, have been known as great sequential learning network, with superior ability of implementing multiple tasks such as language modeling and machine translation. In the mean time, as the distance between input and output sequences elongates, the dependencies of model grows more complicated, which leads to the conjunction of attention mechanisms. Furthermore, to reduce computation of sequential learning which use convolutional neural networks as basic building block and compute hidden layers in parallel, the Transformer occurs to satisfy the present need. Because the Transformer is the first transduction model relying entirely on self-attention to compute representations of its input and output without using sequence-aligned RNNs or convolution. To briefly describe the model structure, it shows similar structure to most of RNNs with encoder-decoder structure.

For the encoder, it begins with a combination of input embedding and positional embedding to present both information itself an their dependencies. And what follows are the stacked self-attention layer, a residual connection around the two sub-layers and a normalization.

For the decoder, similar to the encoder, it inserts another sub-layer to the end of encoder which begins with a masked stacked self-attention layer and followed by residual adding and normalization as well. The masked stacked self-attention is basically offset by one position to achieve current positional dependency rather than global dependency which partially eliminate the effect from previous positional dependency.

Attention could be concluded as a mapping function which takes the query, keys, values as the input and compute a weighted sum of values applying to the corresponding query and keys. The stacked self-attention could also be presented as multi-head attention due to the fact that it is composed of three linear units using mapping function mentioned above to generate a scaled dot-product matrix based on the query, keys and values and concatenate them along parallel layers followed by a linearization to compute the result. In particular (Q, K, V refers to query, key, and value):

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V,$$
$$\text{Multihead}(Q, K, V) = \text{Concat}(\text{Head}_1, \text{Head}_2, ..., \text{Head}_h)W^O,$$

where $\text{Head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$. The point-wise fully connected layer, known as feed-forward networks, is applied to each sub-layer end is implemented as two linear transformations with a ReLU activation: $\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$.

2.3 Transformer-XL

Transformer networks have been a potential of learning longer-term dependency, but are limited by a fixed-length context in the setting of language modeling. There are two main shortcomings of original Transformer model. First is that the largest possible dependency length is bounded by segment length. Secondly, it has been widely adopted to chunk long text fixed-length segments. However, this will lead to the context fragmentation problem that the model cannot capture any longer-term dependency beyond the fixed fragmentation length.

To capture the long-range context in language modeling, there has been a lot of work that feeds a representation of the wider context into the network as an additional input. To overcome the shortcomings that listed above, Transformer-XL introduced two new mechanisms based on vanilla Transformer: recurrence mechanism and relative positional encoding.

The segment-level recurrence with state reuse mechanism is to address the limitations of context fragmentation problem. That is, fix and cache the previous hidden states and reuse them as an

My	dog	also	likes	eating	sausage
B-ARG0	I-ARG0	O	B-V	B-ARG1	I-ARG1

Figure 1: Open IE as a BIO tagging problem.

extended context for the next new segment. This allows the network to have the ability of modeling longer-term dependency and avoiding context fragmentation.

To keep the positional information coherent when reusing the states, Transformer-XL introduces relative positional encoding. The original Transformer takes the element-wise addition of the word embeddings and the positional encodings as input. However, different hidden states are associated with the same positional encoding . Hence the model cannot distinguish the positional difference. Transformer-XL proposes three main changes to the original Transformer attention score. Attention score of standard Transformer:

$$A^{\text{abs}}_{i,j} = q_i^T k_j = \mathbf{E}_{x_i}^T \mathbf{W}_q^T \mathbf{W}_k \mathbf{E}_{x_j} + \mathbf{E}_{x_i}^T \mathbf{W}_q^T \mathbf{W}_k \mathbf{U}_j + \mathbf{U}_i^T \mathbf{W}_q^T \mathbf{W}_k \mathbf{E}_{x_j} + \mathbf{U}_i^T \mathbf{W}_q^T \mathbf{W}_k \mathbf{U}_j,$$

Re-parameterized attention score with relative positional information:

$$A^{\text{rel}}_{i,j} = \mathbf{E}_{x_i}^T \mathbf{W}_q^T \mathbf{W}_{k,E} \mathbf{E}_{x_j} + \mathbf{E}_{x_i}^T \mathbf{W}_q^T \mathbf{W}_{k,R} \mathbf{R}_{i-j} + u^T \mathbf{W}_{k,E} \mathbf{E}_{x_j} + v^T \mathbf{W}_{k,R} \mathbf{R}_{i-j},$$

The new parameterization replace absolute positional embedding $\mathbf{U_j}$ with $\mathbf{R}_{i-j}$ to reflects that only the relative distance matters for where to attend. It also introduces a trainable parameter $\mathbf{u}$ and separates the $\mathbf{W}_k$ into $\mathbf{W}_{k,E}$ and $\mathbf{W}_{k,R}$ for content-based key vectors and location-based key vectors respectively.

2.4 Positional Encoding in Transformer

To allow Transformer to be order-sensitive, which is an essential property of human language, a lot of works have been focusing on designing efficient positional encoding to inform the model of the position information. Sinusoidal positional encoding is first proposed in [1] to encode the absolute position of the object using different wavelengths. [3] and [12] propose relative positional encoding to generalize to attention length longer than the one observed during training. The basic idea is that the distance of two objects is used to calculate positional encoding, which is invariant to the total sequence length. [9] first proposes calibrating one self-attention layer by encouraging it to attend to syntactic parents of each token. There methods either only consider order information or fine-tune the attention in a shallow way.

3 Structurally-Informed Transformer

Open IE [13] aims to extract open-domain assertions represented in the form of n-tuples (e.g., *likes; My dog; eating sausage*) from natural language sentences (e.g., *My dogs also likes eating sausage*). Open IE started from rule-based [14] and syntax-driven systems [15], and recently has used neural networks for supervised learning [7, 16].

Given sentence $\mathbf{s} = (w_1, w_2, ..., w_N)$, the goal of open IE is to extract predicate-argument structures from natural language sentences. It could be formulated as a sequence labeling problem [7, 8]. In particular, the task is to predict a tag sequence $\mathbf{y}$ given a sentence-predicate pair $(\mathbf{s}, v)$ as input. Each $y_i \in \mathbf{y}$ belongs to a discrete set of BIO tags $\mathcal{T}$. For example, the sentence in Figure 1 could be labeled as using a sequence of tags, where the predicate is *likes*, the first argument is *My dog*, and the second argument is *eating sausage*.

We adopt both BiLSTM and Transformer to solve this sequence labeling problem. The stacked BiLSTM has highway connections [17, 18] and recurrent dropout [19]. The Transformer employs

multiple layers of self-attention. Input of the model is the concatenation of word embedding and another embedding indicating whether this word is predicate:

$$\mathbf{x}_t = [\mathbf{W}_{\text{emb}}(w_t), \mathbf{W}_{\text{mask}}(w_t = v)].$$

The probability of the label at each position is calculated independently using a softmax function:

$$P(y_t|\mathbf{s}, v) \propto \exp(\mathbf{W}_{\text{label}}\mathbf{h}_t + \mathbf{b}_{\text{label}}),$$

where $\mathbf{h}_t$ is the hidden state of the last layer of BiLSTM or Transformer. At decoding time, we use the Viterbi algorithm.

3.1 Relative Positional Encoding

In this section, we propose using syntactic dependency to encode the structural information of the sentence. Specifically, given a sentence $\mathbf{s}$, we first extract the dependency tree using spaCy[1]. Then, we use the path between w_i and w_j in $\mathbf{s}$ to describe their structural relation. Dependency tree is a directed acyclic graph, which guarantees that there is a unique path between any two tokens. For example, the relational path between *My* and *eating* in the sentence shown in Figure 2 consists of three edges:

$$\text{-poss}, \text{-nsubj}, \text{+xcomp},$$

where -dep means moving along dep in the reverse direction and +dep means moving along dep in the same direction. We use $*$ to denote the relation between any token and itself. After preprocessing, we can obtain a $N \times N$ relational matrix M for sentence $\mathbf{s}$, where $M_{i,j}$ is the relational path between w_i and w_j, as shown in Table 1. Note that we can exactly reconstruct the dependency tree using M.

	My	dog	also	likes	...
My	*	-poss	-poss, -nsubj, +advmod	-poss, -nsubj	...
dog	+poss	*	-nsubj, +advmod	-nsubj	...
also	-advmod, +nsubj, +poss	-advmod, +nsubj	*	-advmod	...
likes	+nsubj, +poss	+nsubj	+advmod	*	...
...	...	...	...	...	...

Table 1: Relational path matrix (partial).

Having defined the relations between tokens using dependency tree, the next question is how to encode the relations so as to inform the Transformer of the structural information. We consider two ways to do this. The first one is to directly embed the relational path using embedding lookup tables. The embedding of a relational path $M_{i,j} = (d_1, d_2, ..., d_T)$ is computed by the average over the embedding of each dependency edge:

$$\mathbf{R}_{i,j} = \frac{\sum_{i=1}^{T} \mathbf{e}_{d_i}}{T},$$

where $\mathbf{e}_{d_i}$ is the embedding of the dependency edge d_i. The second way is to use LSTMs to encode the relational path, which explicitly considers the orders of the sequence:

$$\mathbf{h}_i^r = \text{LSTM}(\mathbf{h}_{i-1}^r, \mathbf{e}_{d_i}),$$
$$\mathbf{R}_{i,j} = \mathbf{h}_T^r.$$

We implement both methods and conduct experiments to investigate their performance in subsection 4.3. $\mathbf{R}_{i,j}$ for all pairs of tokens are fed to the self-attention model as relative positional encoding to inform Transformer about the relations between pairs of tokens.

3.2 Structurally-Informed Attention

Having obtained the relative positional encoding between any two tokens, we modify standard Transformer to incorporate this structural information. Our modification is very similar to Transformer-XL [3]. At each layers of self-attention, the relative positional encoding $\mathbf{R} \in \mathbb{R}^{N \times N \times E}$ is feed as input

[1] `https://spacy.io`

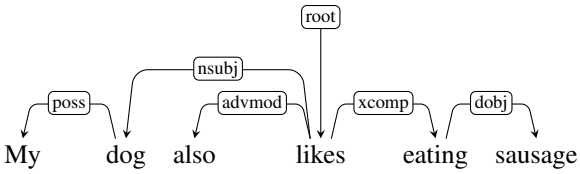

Figure 2: Dependency parse tree.

Figure 3: Input to Transformer.

to compute the key embedding key and value vectors for each token, where $\mathbf{R}_{i,j}$ is the relative positional encoding of the relational path between w_i and w_j. In particular, the attention value $A_{i,j}$ is computed by:

$$\begin{aligned} A^{\text{rel}}_{i,j} =& \mathbf{E}^T_{x_i}\mathbf{W}^T_q\mathbf{W}_{k,E}\mathbf{E}_{x_j} + \mathbf{E}^T_{x_i}\mathbf{W}^T_q\mathbf{W}_{k,R}(\mathbf{R}_{i,j} + \mathbf{U}_j) + \\ & \mathbf{U}^T_i\mathbf{W}^T_q\mathbf{W}_{k,E}\mathbf{E}_{x_j} + \mathbf{U}^T_i\mathbf{W}^T_q\mathbf{W}_{k,R}(\mathbf{R}_{i,j} + \mathbf{U}_j). \end{aligned}$$

Comparing to the relative positional encoding in Transformer-XL, ours in a combination of both standard absolute positional encoding ($\mathbf{U}_j$) and relative positional encoding derived from dependency structures ($\mathbf{R}_{i,j}$), which has the potential to model both order information and structural information. The input to the Transformer is thus the combination of word embedding, relative positional embedding, and absolute positional embedding. For example, if we want to calculate the attention between *likes* and other tokens in the sentence of Figure 2, the embedding is depicted in Figure 3. With this simple modification, we can make the Transformer structurally informed.

4 Experiments

4.1 Experimental Settings

Dataset We use four datasets, namely OIE2016, WEB, NYT, and PENN [7] to evaluate the performance of our model. The details of these datasets are illustrated in Table 2. These datasets are annotated from different domains, such as Wikipedia, news, and web. They contain different numbers of arguments for each predicate, which can be used to test different aspects of the extraction model. Gold predicates, namely their index in the sentence, are provided. Therefore, each predicate-sentence pair corresponds to one training sample. We follow the train-development-test split for these datasets using in previous works [7].

Datasets	Domain	#Sentences	#Tuples
OIE2016	News,Wiki	3200	8477
WEB	News,Web	500	461
NYT	News,Wiki	222	222
PENN	Mixed	100	51

Table 2: Datasets statistics

Evaluation Metrics We follow the evaluation metrics described by [20]: area under the precision-recall curve (AUC) and F1 score. An extraction is judged as correct if the predicate and arguments include the syntactic head of the gold standard counterparts. Note that this matching function is very challenging, because an extraction counts as a successful extraction only if it makes correct predictions on all the arguments and predicate. That is part of the reason why the performance is still relatively low.

Baselines We use both BiLSTM and Transformer as our baselines in this paper. For the BiLSTM model, 100-dimensional word embedding [21] is concatenated with another 100-dimensional predicate indicator embedding as inputs. The network consists of 4 BiLSTM layers (2 forward and 2 backward) with 256 hidden units. The probability of recurrent dropout [19] is set to 0.1. Batch size is 80. Adadelta optimizer [22] uses $\epsilon = 10^{-6}$ and $\rho = 0.95$.

For Transformer, the inputs is exactly the same as those of the BiLSTM model. We use 4 self-attention layers with 4 50-dimensional heads in each layer. The feedforward layer has 200 hidden units. Dropout probability for the residual connections is 0.2. Batch size is 80 and the number of warm-up steps is 1000. After training under several different combination of parameters, the best performance we achieved on Transformer of which the number of attention heads is 8, number of layers is 6, number of warm-up steps is 1500 and residual dropout probability is 0.05.

4.2 Overall Experimental Results

Methods	OIE2016		WEB		NYT		PENN	
	AUC	F1	AUC	F1	AUC	F1	AUC	F1
BiLSTM	.019	.104	.025	.140	.002	.042	.002	.047
Transformer	.005	.054	.009	.080	.002	.032	.001	.031
StructTransformer	.024	.133	.018	.107	.001	.029	.002	.042

Table 3: Overall performance of baseline models and StructTransformer.

Table 3 shows the overall performance of the baseline models and the proposed structurally-informed Transformer (denoted by StructTransformer). Because the training data is very small and the evaluation metric is very harsh, the performance is relatively low, which indicates that open IE is a challenging problem, especially when there are multiple arguments within an extraction. Overall, BiLSTM and StructTransformer yield better performance than vanilla Transformer, and StructTransformer is slightly better than BiLSTM on OIE2016 dataset.

The advantage of leveraging structural information in Transformer is clearly demonstrated by the improvement on OIE2016 and WEB datasets over vanilla Transformer. It shows that explicit structural information has the potential to calibrate the attention mechanism in Transformer, which if beneficial when the training data is very limited. The structural information is exposed to the model so that the attentions is aware of the underlying structure of the sentence, which is crucial for identifying predicate-argument relations. Typically, Transformer is more data hungry than LSTM, and that is why vanilla Transformer performs worse than BiLSTM on our small datasets.

4.3 Experiments with Different Structural Information

Methods	OIE2016		WEB		NYT		PENN	
	AUC	F1	AUC	F1	AUC	F1	AUC	F1
3-hop + Lookup	.024	.133	.018	.107	.001	.029	.002	.042
5-hop + Lookup	.023	.116	.013	.098	.004	.065	.001	.031
3-hop + LSTM	.021	.121	.014	.111	.003	.051	.002	.052

Table 4: Performance of StructTransformer with different settings.

We also tried different configurations of our model to investigate how the structural information affects the performance. n-hop means that we truncate the path between any two words by only using the first n steps. Lookup means that we embed the path using embedding lookup table and LSTM means that we encode the path using LSTM. The results suggest that it is not always beneficial to use more hops, because the more steps we use, the more likely the model overfits. Embedding lookup is simpler than LSTM and more efficient on our small datasets.

overgenerated predicate	wrong argument	missing argument
41%	38%	21%

Table 5: Proportions of three errors.

A CEN forms an important but small part of a Local Strategic Partnership .
A Democrat , he became the youngest mayor in Pittsburgh's history in September 2006 at the age of 26 .
An animal that cares for its young but shows no other sociality traits is said to be " subsocial" .

Table 6: Case study of extractions. Green for arguments and red for predicates.

4.4 Case Study and Error Analysis

Table 6 showcase some extractions generated by our system. We can see that be aware of structure of the sentence is of central importance in open IE, because arguments usually consist of syntactic units that are semantically related to the predicate.

Why is the performance still relatively low? We randomly sample 50 extractions generated by our model and conduct an error analysis to answer this question. To count as a correct extraction, the number and order of the arguments should be exactly the same as the ground truth and syntactic heads must be included, which is challenging considering that the open IE datasets have complex syntactic structures and multiple arguments per predicate. We classify the errors into three categories: (1) "Overgenerated predicate" is where predicates not included in ground truth are overgenerated, because all the verbs are used as candidate predicates. An effective mechanism should be designed to reject useless candidates. (2) "Wrong argument" is where extracted arguments do not coincide with ground truth, which is mainly caused by merging multiple arguments in ground truth into one. (3) "Missing argument" is where the model fails to recognize arguments. These two errors usually happen when the structure of the sentence is complicated and coreference is involved.

5 Conclusion

In this paper, we investigate the effectiveness of encoding structural information to enable Transformer to take intrinsic structures of natural language sentence into account. Experiments demonstrates that the structural information is useful to Transformer model and more accurate assertions can be generated. However, there are still a lot of different ways to incorporate structural information in Transformer, and it might be interesting to let Transformer learn the structure of sentence and open IE simultaneously.

6 Teammates and Work Division

- Zhengbao Jiang: model implementation; experiments of StructTransformer; paper writing.
- Libing Zhang: model implementation; experiments of Transformer; paper writing.
- Runyu Xiao: model implementation; experiments of BiLSTM; paper writing.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6000–6010, 2017.

[2] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pages 1310–1318, 2013.

[3] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *CoRR*, abs/1901.02860, 2019.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[5] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1556–1566, 2015.

[6] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Graph neural networks: A review of methods and applications. *CoRR*, abs/1812.08434, 2018.

[7] Gabriel Stanovsky, Julian Michael, Luke Zettlemoyer, and Ido Dagan. Supervised open information extraction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 885–895, 2018.

[8] Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 473–483, 2017.

[9] Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 5027–5038, 2018.

[10] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 764–773, 2017.

[11] Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. Hierarchical multiscale recurrent neural networks. *CoRR*, abs/1609.01704, 2016.

[12] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, pages 464–468, 2018.

[13] Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. Open information extraction from the web. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 2670–2676, 2007.

[14] Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545, 2011.

[15] Luciano Del Corro and Rainer Gemulla. Clausie: clause-based open information extraction. In *22nd International World Wide Web Conference*, pages 355–366, 2013.

[16] Lei Cui, Furu Wei, and Ming Zhou. Neural open information extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 407–413, 2018.

[17] Yu Zhang, Guoguo Chen, Dong Yu, Kaisheng Yao, Sanjeev Khudanpur, and James R. Glass. Highway long short-term memory RNNS for distant speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2016, Shanghai, China, March 20-25, 2016*, pages 5755–5759, 2016.

[18] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2377–2385, 2015.

[19] Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 1019–1027, 2016.

[20] Gabriel Stanovsky and Ido Dagan. Creating a large benchmark for open information extraction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2300–2305, 2016.

[21] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543, 2014.

[22] Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012.