Interpretability Rules: Jointly Bootstrapping a Neural Relation Extractor with an Explanation Decoder

Zheng Tang, Mihai Surdeanu
Department of Computer Science
University of Arizona, Tucson, Arizona, USA
{zhengtang, msurdeanu}@email.arizona.edu

Abstract

We introduce a method that transforms a rule-based relation extraction (RE) classifier into a neural one such that both interpretability and performance are achieved. Our approach jointly trains a RE classifier with a decoder that generates explanations for these extractions, using as sole supervision a set of rules that match these relations. Our evaluation on the TACRED dataset shows that our neural RE classifier outperforms the rule-based one we started from by 9 F1 points; our decoder generates explanations with a high BLEU score of over 90%; and, the joint learning improves the performance of both the classifier and decoder.

1 Introduction

Information extraction (IE) is one of the key challenges in the natural language processing (NLP) field. With the explosion of unstructured information on the Internet, the demand for high-quality tools that convert free text to structured information continues to grow (Chang et al., 2010; Lee et al., 2013; Valenzuela-Escarcega et al., 2018).

The past decades have seen a steady transition from rule-based IE systems (Appelt et al., 1993) to methods that rely on machine learning (ML) (see Related Work). While this transition has generally yielded considerable performance improvements, it was not without a cost. For example, in contrast to modern deep learning methods, the predictions of rule-based approaches are easily explainable, as a small number of rules tends to apply to each extraction. Further, in many situations, rule-based methods can be developed by domain experts with minimal training data. For these reasons, rule-based IE methods remain widely used in industry (Chiticariu et al., 2013).

In this work we demonstrate that this transition from rule- to ML-based IE can be performed such that the benefits of both worlds are preserved. In particular, we start with a rule-based relation extraction (RE) system (Angeli et al., 2015) and bootstrap a neural RE approach that is trained jointly with a decoder that learns to generate the rules that best explain each particular extraction. The contributions of our idea are the following:

(1) We introduce a strategy that jointly learns a RE classifier between pairs of entity mentions with a decoder that generates explanations for these extractions in the form of Tokensregex (Chang and Manning, 2014) or Semregex (Chambers et al., 2007) patterns. The only supervision for our method is a set of input rules (or patterns) in these two frameworks (Angeli et al., 2015), which we use to generate positive examples for both the classifier and the decoder. We generate negative examples automatically from the sentences that contain positives examples.

(2) We evaluate our approach on the TACRED dataset (Zhang et al., 2017) and demonstrate that: (a) our neural RE classifier outperforms considerably the rule-based one we started from; (b) our decoder generates explanations with high accuracy, i.e., a BLEU overlap score between the generated rules and the gold, hand-written rules of over 90%; and, (c) joint learning improves the performance of both the classifier and decoder.

(3) We demonstrate that our approach generalizes to the situation where a vast amount of labeled training data is combined with a few rules. We combined the TACRED training data with the above rules and showed that when our method is trained on this combined data, the classifier obtains near state-of-art performance at 67.0% F1, while the decoder generates accurate explanations with a BLEU score of 92.4%.

2 Related Work

Relation extraction using statistical methods is well studied. Methods range from supervised, "traditional" approaches (Zelenko et al., 2003;

Bunescu and Mooney, 2005) to neural methods. Neural approaches for RE range from methods that rely on simpler representations such as CNNs (Zeng et al., 2014) and RNNs (Zhang and Wang, 2015) to more complicated ones such as augmenting RNNs with different components (Xu et al., 2015; Zhou et al., 2016), combining RNNs and CNNs (Vu et al., 2016; Wang et al., 2016), and using mechanisms like attention (Zhang et al., 2017) or GCNs (Zhang et al., 2018). To solve the lack of annotated data, distant supervision (Mintz et al., 2009; Surdeanu et al., 2012) is commonly used to generate a training dataset from an existing knowledge base. Jat et al. (2018) address the inherent noise in distant supervision with an entity attention method.

Rule-based methods in IE have also been extensively investigated. Riloff (1996) developed a system that learns extraction patterns using only a pre-classified corpus of relevant and irrelevant texts. Lin and Pantel (2001) proposed a unsupervised method for discovering inference rules from text based on the Harris distributional similarity hypothesis (Harris, 1954). Valenzuela-Escárcega et al. (2016) introduced a rule language that covers both surface text and syntactic dependency graphs. Angeli et al. (2015) further show that converting rule-based models to statistical ones can capture some of the benefits of both, i.e., the precision of patterns and the generalizability of statistical models.

Interpretability has gained more attention recently in the ML/NLP community. For example, some efforts convert neural models to more interpretable ones such as decision trees (Craven and Shavlik, 1996; Frosst and Hinton, 2017). Some others focus on producing a post-hoc explanation of individual model outputs (Ribeiro et al., 2016; Hendricks et al., 2016).

Inspired by these directions, here we propose an approach that combines the interpretability of rule-based methods with the performance and generalizability of neural approaches.

3 Approach

Our approach jointly addresses classification and interpretability through an encoder-decoder architecture, where the decoder uses multi-task learning (MTL) for relation extraction between pairs of named entities (Task 1) and rule generation (Task 2). Figure 1 summarizes our approach.

3.1 Task 1: Relation Classifier

We define the RE task as follows. The inputs consist of a sentence $W = [w_1, \dots, w_n]$, and a pair of entities (called "subject" and "object") corresponding to two spans in this sentence: $W_s = [w_{s_1}, \dots, w_{s_n}]$ and $W_o = [w_{o_1}, \dots, w_{o_n}]$. The goal is to predict a relation $r \in R$ (from a predefined set of relation types) that holds between the subject and object or "no relation" otherwise.

For each sentence, we associate each word w_i with a representation $\boldsymbol{x}_i$ that concatenates three embeddings: $\boldsymbol{x}_i = \boldsymbol{e}(w_i) \circ \boldsymbol{e}(n_i) \circ \boldsymbol{e}(p_i)$, where $\boldsymbol{e}(w_i)$ is the word embedding of token i, $\boldsymbol{e}(n_i)$ is the NER embedding of token i, $\boldsymbol{e}(p_i)$ is the POS Tag embedding of token i. We feed these representations into a sentence-level bidirectional LSTM encoder (Hochreiter and Schmidhuber, 1997):

$$[\boldsymbol{h}_1, \dots, \boldsymbol{h}_n] = \text{LSTM}([\boldsymbol{x}_1, \dots, \boldsymbol{x}_n]) \tag{1}$$

Following (Zhang et al., 2018), we extract the "K-1 pruned" dependency tree that covers the two entities, i.e., the shortest dependency path between two entities enhanced with all tokens that are directly attached to the path, and feed it into a GCN (Kipf and Welling, 2016) layer:

$$\boldsymbol{h}_i^{(l)} = \sigma(\sum_{j=1}^{n} \tilde{A}_{ij} \boldsymbol{W}^{(l)} \boldsymbol{h}_j^{(l-1)} / d_i + \boldsymbol{b}^{(l)}) \tag{2}$$

where $\boldsymbol{A}$ is the corresponding adjacency matrix, $\tilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}$ with $\boldsymbol{I}$ being the $n \times n$ identity matrix, $d_i = \sum_{j=1}^{n} \tilde{A}_{ij}$ is the degree of token i in the resulting graph, and $\boldsymbol{W}^{(l)}$ is linear transformation.

Lastly, we concatenate the sentence representation, the subject entity representation, and the object entity representation as follows:

$$\boldsymbol{h}_{sent} = f(\boldsymbol{h}^{(L)}) = f(\text{GCN}(\boldsymbol{h}^{(0)})) \tag{3}$$

$$\boldsymbol{h}_s = f(\boldsymbol{h}_{s_1:s_n}^{(L)}) \tag{4}$$

$$\boldsymbol{h}_o = f(\boldsymbol{h}_{o_1:o_n}^{(L)}) \tag{5}$$

$$\boldsymbol{h}_{final} = \boldsymbol{h}_{sent} \circ \boldsymbol{h}_s \circ \boldsymbol{h}_o \tag{6}$$

where $\boldsymbol{h}^{(l)}$ denotes the collective hidden representations at layer l of the GCN, and $f : \mathbb{R}^{d \times n} \rightarrow \mathbb{R}^d$ is a max pooling function that maps from n output vectors to the representation vector. The concatenated representation $\boldsymbol{h}_{final}$ is fed to a feed-forward layer with a softmax function to produce a probability distribution over relation types.

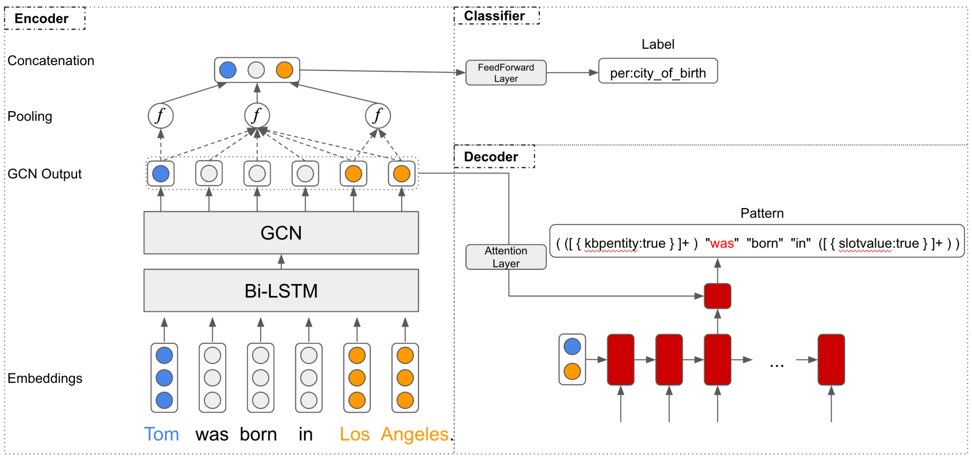

Figure 1: Neural architecture of the proposed multitask learning approach. The input is a sequence of words together with NER labels and POS tags. The pair of entities to be classified ("subject" in blue and "object" in orange) are also provided. We use a concatenation of several representations, including embeddings of words, NER labels, and POS tags. The encoder uses a sentence-level bidirectional LSTM (biLSTM) and graph convolutional networks (GCN). There are pooling layers for the subject, object, and full sentence GCN outputs. The concatenated pooling outputs are fed to the classifier's feedforward layer. The decoder is an LSTM with an attention mechanism.

3.2 Task 2: Rule Decoder

The rule decoder's goal is to generate the pattern P that extracted the corresponding data point, where P is represented as a sequence of tokens in the corresponding pattern language: $P = [p_1, \ldots, p_n]$. For example, the pattern `(([{kbpentity:true}]+)/was/ /born/ /on/([{slotvalue:true}]+))` (where `kbpentity:true` marks subject tokens, and `slotvalue:true` marks object tokens) extracts mentions of the `per:date_of_birth` relation.

We implemented this decoder using an LSTM with an attention mechanism. To center rule decoding around the subject and object, we first feed the concatenation of subject and object representation from the encoder as the initial state in the decoder. Then, in each timestep t, we generate the attention context vector $\boldsymbol{C}_t^D$ by using the current hidden state of the decoder, $\boldsymbol{h}_t^D$:

$$\boldsymbol{s}_t(j) = \boldsymbol{h}_{(L)}^E \boldsymbol{W}^A \boldsymbol{h}_t^D \tag{7}$$

$$\boldsymbol{a}_t = \text{softmax}(\boldsymbol{s}_t) \tag{8}$$

$$\boldsymbol{C}_t^D = \sum_j \boldsymbol{a}_t(j) \boldsymbol{h}_j^E \tag{9}$$

where $\boldsymbol{W}^A$ is a learned matrix, and $\boldsymbol{h}_{(L)}^E$ are hidden representations from the encoder's GCN.

We feed this $\boldsymbol{C}_t^D$ vector to a single feed forward layer that is coupled with a softmax function and use its output to obtain a probability distribution over the pattern vocabulary.

Approach	Precision	Recall	F1	BLEU
	Rule-only data			
Rule baseline	**86.9**	23.2	36.6	–
Our approach	60.0	**36.7**	**45.5**	**90.3**
w/o decoder	58.7	36.4	44.9	–
w/o classifier	–	–	–	88.3
	Rules + TACRED training data			
C-GCN	69.9	63.3	66.4	–
Our approach	70.2	**64.0**	**67.0**	**92.4**
w/o decoder	**71.2**	62.3	66.5	–
w/o classifier	–	–	–	91.6

Table 1: Results on the TACRED test partition, including ablation experiments (the "w/o" rows). We experimented with two configurations: *Rule-only data* uses only training examples generated by rules; *Rules + TACRED training data* applies the previous rules to the training dataset from TACRED.

We use cross entropy to calculate the losses for both the classifier and decoder. To balance the loss between classifier and decoder, we normalize the decoder loss by the pattern length. Note that for the data points without an existing rule, we only calculate the classifier loss. Formally, the joint loss function is:

$$\text{loss} = \text{loss}_c + \text{loss}_d / \mathit{length}(P) \tag{10}$$

4 Experiments

Data Preparation: We report results on the TACRED dataset (Zhang et al., 2017). We bootstrap

Hand-written Rule	Decoded Rule
(([{kbpentity:true}]+)""" based ""in"([{slotvalue:true}]+))	(([{kbpentity:true}]+)"in"([{slotvalue:true}]+))
(([{kbpentity:true}]+)" CEO "([{slotvalue:true}]+))	(([{kbpentity:true}]+)" president "([{slotvalue:true}]+))

Table 2: Examples of mistakes in the decoded rules. We highlight in the hand-written rules the tokens that were missed during decoding (false negatives) in green, and in the decoded rules we highlight the spurious tokens (false positives) in red.

Model	Precision	Recall	F1	BLEU
20% of rules	74.9	20.1	31.7	96.9
40% of rules	69.0	26.9	38.8	90.8
60% of rules	62.7	29.7	40.3	88.8
80% of rules	57.3	36.5	44.6	89.4

Table 3: Learning curve of our approach based on amount of rules used, in the *rule-only data* configuration. These results are on TACRED development.

our models from the patterns in the rule-based system of Angeli et al. (2015), which uses 4,528 surface patterns (in the Tokensregex language) and 169 patterns over syntactic dependencies (using Semgrex). We experimented with two configurations: *rule-only data* and *rules + TACRED training data*. In the former setting, we use solely positive training examples generated by the above rules. We combine these positive examples with negative ones generated automatically by assigning 'no_relation' to all other entity mention pairs in the same sentence where there is a positive example.[1] We generated 3,850 positive and 12,311 negative examples for this configuration. In the latter configuration, we apply the same rules to the entire TACRED training dataset.[2]

Baselines: We compare our approach with two baselines: the rule-based system of Zhang et al. (2017), and the best non-combination method of Zhang et al. (2018). The latter method uses an LSTM and GCN combination similar to our encoder.[3]

Implementation Details: We use pre-trained GloVe vectors (Pennington et al., 2014) to initialize our word embeddings. We use the *Adagrad* optimizer (Duchi et al., 2011). We apply entity masking to subject and object entities in the sentence, which is replacing the original token with a special `<NER>`–SUBJ or `<NER>`–OBJ token where `<NER>` is the corresponding name entity label provided by TACRED.

We used micro precision, recall, and F1 scores to evaluate the RE classifier. We used the BLEU score to measure the quality of generated rules, i.e., how close they are to the corresponding gold rules that extracted the same output. We used the BLEU implementation in NLTK (Loper and Bird, 2002), which allows us to calculate multi-reference BLEU scores over 1 to 4 grams.[4] We report BLEU scores only over the non 'no_relation' extractions with the corresponding testing data points that are matched by one of the rules in (Zhang et al., 2017).

Results and Discussion: Table 1 reports the overall performance of our approach, the baselines, and ablation settings, for the two configurations investigated. We draw the following observations from these results:

(1) The rule-based method of Zhang et al. (2017) has high precision but suffers from low recall. In contrast, our approach that is bootstrapped from the same information has 13% higher recall and almost 9% higher F1 (absolute). Further, our approach decodes explanatory rules with a high BLEU score of 90%, which indicates that it maintains almost the entire explanatory power of the rule-based method.

(2) The ablation experiments indicate that *joint* training for classification and explainability helps both tasks, in both configurations. This indicates that performance and explainability are interconnected.

(3) The two configurations analyzed in the table demonstrate that our approach performs well not only when trained solely on rules, but also when rules are combined with a training dataset annotated for RE. This suggests that our direction may

[1] During the generation of these negative examples we filtered out pairs corresponding to inverse and symmetric relations. For example, if a sentence contains a relation (Subj, Rel, Obj), we do not generate the negative (Obj, no_relation, Subj) if Rel has an inverse relation, e.g., `per:children` is the inverse of `per:parents`.

[2] Thus, some training examples in this case will be associated with a rule and some will not. We adjusted the loss function to use only the classification loss when no rule applies.

[3] For a fair comparison, we do not compare against ensemble methods, or transformer-based ones. Also, note that this baseline does *not* use rules at all.

[4] We scored longer n-grams to better capture rule syntax.

be a general strategy to infuse some explainability in a statistical method, when rules are available during training.

(4) Table 3 lists the learning curve for our approach in the *rule-only data* configuration when the amount of rules available varies.[5] This table shows that our approach obtains a higher F1 than the complete rule-based RE classifier even when using only 40% of the rules.[6]

(5) Note that the BLEU score provides an incomplete evaluation of rule quality. To understand if the decoded rules explain their corresponding data point, we performed a manual evaluation on 176 decoded rules. We classified them into three categories: (a) the rules correctly explain the prediction (according to the human annotator), (b) they approximately explain the prediction, and (c) they do not explain the prediction. Class (b) contains rules that do not lexically match the input text, but capture the correct semantics, as shown in Table 2. The percentages we measured were: (a) 33.5%, (b) 31.3%, (c) 26.1%. 9% of these rules were skipped in the evaluation because they were false negatives(which are labeled as no relation falsely by our model). These numbers support our hypothesis that, in general, the decoded rules do explain the classifier's prediction.

Further, out of 750 data points associated with rules in the evaluation data, our method incorrectly classifies only 26. Out of these 26, 16 were false negatives, and had no rules decoded. In the other 10 predictions, 7 rules fell in class (b) (see the examples in Table 2). The other 3 were incorrect due to ambiguity, i.e., the pattern created is an ambiguous succession of POS tags or syntactic dependencies without any lexicalization. This suggests that, even when our classifier is incorrect, the rules decoded tend to capture the underlying semantics.

5 Conclusion

We introduced a strategy that jointly bootstraps a relation extraction classifier with a decoder that generates explanations for these extractions, using as sole supervision a set of example patterns that match such relations. Our experiments on the TACRED dataset demonstrated that our approach outperforms the strong rule-based method that provided the training patterns by 9 F1 points, while decoding explanations at over 90% BLEU score. Further, we showed that the joint training of the classification and explanation components performs better than training them separately. All in all, our work suggests that it is possible to marry the interpretability of rule-based methods with the performance of neural approaches.

[5]For this experiment we sorted the rules in descending order of their match frequency in training, and kept the top $n\%$ in each setting.

[6] The high BLEU score in the 20% configuration is due to the small sample in development for which gold rules exist.

References

Gabor Angeli, Victor Zhong, Danqi Chen, A. Chaganty, J. Bolton, Melvin Jose Johnson Premkumar, Panupong Pasupat, S. Gupta, and Christopher D. Manning. 2015. Bootstrapped self training for knowledge base population. *Theory and Applications of Categories*.

Douglas E Appelt, Jerry R Hobbs, John Bear, David Israel, and Mabry Tyson. 1993. Fastus: A finite-state processor for information extraction from real-world text. In *IJCAI*, volume 93, pages 1172–1178.

Razvan Bunescu and Raymond Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 724–731.

Nathanael Chambers, Daniel Cer, Trond Grenager, David Hall, Chloe Kiddon, Bill MacCartney, Marie-Catherine de Marneffe, Daniel Ramage, Eric Yeh, and Christopher D. Manning. 2007. Learning alignments and leveraging natural logic. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 165–170, Prague. Association for Computational Linguistics.

Angel X Chang and Christopher D Manning. 2014. Tokensregex: Defining cascaded regular expressions over tokens. *Stanford University Computer Science Technical Reports. CSTR*, 2:2014.

Angel X Chang, Valentin I Spitkovsky, Eric Yeh, Eneko Agirre, and Christopher D Manning. 2010. Stanford-ubc entity linking at tac-kbp.

Laura Chiticariu, Yunyao Li, and Frederick Reiss. 2013. Rule-based information extraction is dead! long live rule-based information extraction systems! In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 827–832.

Mark Craven and Jude W Shavlik. 1996. Extracting tree-structured representations of trained networks. In *Advances in neural information processing systems*, pages 24–30.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7).

Nicholas Frosst and Geoffrey Hinton. 2017. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. 2016. Generating visual explanations. In *European Conference on Computer Vision*, pages 3–19. Springer.

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

Sharmistha Jat, Siddhesh Khandelwal, and Partha Talukdar. 2018. Improving distantly supervised relation extraction using word and entity based attention. *arXiv preprint arXiv:1804.06987*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2013. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, 39(4):885–916. Copyright: Copyright 2020 Elsevier B.V., All rights reserved.

Dekang Lin and P. Pantel. 2001. Dirt – discovery of inference rules from text.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. In *In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Philadelphia: Association for Computational Linguistics*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144. ACM.

Ellen Riloff. 1996. Automatically generating extraction patterns from untagged text. In *Proceedings of the national conference on artificial intelligence*, pages 1044–1049.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 455–465, Jeju Island, Korea. Association for Computational Linguistics.

Marco A. Valenzuela-Escarcega, Ozgun Babur, Gus Hahn-Powell, Dane Bell, Thomas Hicks, Enrique Noriega-Atala, Xia Wang, Mihai Surdeanu, Emek Demir, and Clayton T. Morrison. 2018. Large-scale automated machine reading discovers new cancer driving mechanisms. *Database: The Journal of Biological Databases and Curation*.

Marco A. Valenzuela-Escárcega, Gus Hahn-Powell, and Mihai Surdeanu. 2016. Odin's runes: A rule language for information extraction. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 322–329, Portorož, Slovenia. European Language Resources Association (ELRA).

Ngoc Thang Vu, Heike Adel, Pankaj Gupta, and Hinrich Schütze. 2016. Combining recurrent and convolutional neural networks for relation classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 534–539, San Diego, California. Association for Computational Linguistics.

Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. 2016. Relation classification via multi-level attention CNNs. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1298–1307, Berlin, Germany. Association for Computational Linguistics.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1785–1794, Lisbon, Portugal. Association for Computational Linguistics.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of machine learning research*, 3(Feb):1083–1106.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344.

Dongxu Zhang and Dong Wang. 2015. Relation classification via recurrent neural network. *arXiv preprint arXiv:1508.01006*.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2205–2215, Brussels, Belgium. Association for Computational Linguistics.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, pages 35–45.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 207–212, Berlin, Germany. Association for Computational Linguistics.

A Experimental Details

We use the dependency parse trees, POS and NER sequences as included in the original release of the TACRED dataset, which was generated with Stanford CoreNLP (Manning et al., 2014). We use the pretrained 300-dimensional GloVe vectors (Pennington et al., 2014) to initialize word embeddings. We use a 2 layers of bi-LSTM, 2 layers of GCN, and 2 layers of feedforward in our encoder. And 2 layers of LSTM and 1 layer of feedforward in our decoder. Table 4 shows the details of the proposed neural network. We apply the ReLU function for all nonlinearities in the GCN layers and the standard max pooling operations in all pooling layers. For regularization we use dropout with p = 0.5 to all encoder LSTM layers and all but the last GCN layers.

For training, we use Adagrad (Duchi et al., 2011) an initial learning rate, and from epoch 1 we start to anneal the learning rate by a factor of 0.9 every time the F1 score on the development set does not increase after one epoch. We tuned the initial learning rate between 0.01 and 1; we chose 0.3 as this obtained the best performance on development. We trained 100 epochs for all the experiments with a batch size of 50. There were 3,850 positive data points and 12,311 negative data in the rule-only data. For this dataset, it took 1 minute to finish one epoch in average. And for Rules + TACRED training data, it took 4 minutes to finish one epoch in average[7].

Encoder and classifier components	Size
Vocabulary	53953
POS embedding dimension	30
NER embedding dimension	30
LSTM hidden layers	200
Feedforward layers	200
GCN layers	200
Relation	41
Decoder component	**Size**
LSTM hidden layers	200
Pattern embedding dimension	100
Feedforward layer	200
Maximum decoding length	100
Pattern	1141

Table 4: Details of our neural architecture.

All the hyperparameters above were tuned manually. We trained our model on PyTorch 3.8.5 with CUDA version 10.0, using one NVDIA Titan RTX.

B Dataset Introduction

You can find the details of TACRED data in this link: `https://nlp.stanford.edu/projects/tacred/`.

C Rules

The rule-base system we use is the combination of Stanford's Tokensregex (Chang and Manning, 2014) and Semregex (Chambers et al., 2007). The rules we use are from the system of Angeli et al. (2015), which contains 4528 Tokensregex patterns and 169 Semgrex patterns.

We extracted the rules from CoreNLP and mapped each rule to the TACRED dataset. We provided the mapping files in our released dataset. We also generate the dataset with only datapoints matched by rules in TACRED training partition and its mapping file.

[7]The software is available at this URL: https://github.com/clulab/releases/tree/master/naacl-trustnlp2021-edin.