Exploring Interpretability in Event Extraction: Multitask Learning of a Neural Event Classifier and an Explanation Decoder

[†]Zheng Tang, [‡]Gustave Hahn-Powell, [†]Mihai Surdeanu
[†]Department of Computer Science
[‡]Department of Linguistics
University of Arizona, Tucson, Arizona, USA
{zhengtang, hahnpowell, msurdeanu}@email.arizona.edu

Abstract

We propose an interpretable approach for event extraction that mitigates the tension between generalization and interpretability by jointly training for the two goals. Our approach uses an encoder-decoder architecture, which jointly trains a classifier for event extraction, and a rule decoder that generates syntactico-semantic rules that explain the decisions of the event classifier. We evaluate the proposed approach on three biomedical events and show that the decoder generates interpretable rules that serve as accurate explanations for the event classifier's decisions, and, importantly, that the joint training generally improves the performance of the event classifier. Lastly, we show that our approach can be used for semi-supervised learning, and that its performance improves when trained on automatically-labeled data generated by a rule-based system.

1 Introduction

Interpretability is a key requirement for machine learning (ML) in many domains, e.g., legal, medical, finance. In the words of (Ribeiro et al., 2016), "if users do not trust the model or a prediction, they will not use it." However, there is a tension between generalization and interpretability in deep learning, as interpretable models are often generated by "distilling" a model with good generalization, e.g., a deep learning one that relies on distributed representations, into models that are more interpretable but lose some generalization, e.g., linear models or decision trees (Craven and Shavlik, 1996; Ribeiro et al., 2016; Frosst and Hinton, 2017). Here, we argue that both generalization and interpretability are equally important. For example, in the medical space, a patient will likely reject a treatment recommended by an algorithm without an explanation. Closer to natural language processing (NLP), a statistical information extraction method that converts free text in a specific domain to structured knowledge should also provide human-understandable explanations of its extractions. This allows the subject matter expert to quality check such output without a deep knowledge of the underlying machinery, which is a necessity in successful inter-disciplinary NLP collaborations.

In this work, we propose an interpretable approach for event extraction (EE) that mitigates the tension between generalization and interpretability through multitask learning (MTL). Our approach uses an attention-based encoder to encode the input text and given entities of interest (e.g., proteins in the biomedical domain), and a decoder that jointly trains two tasks. The first task is event classification, which identifies which event applies for a given entity (e.g., phosphorylation). The second task decodes a rule in the Odin language (Valenzuela-Escárcega et al., 2018; Valenzuela-Escárcega et al., 2016), which explains the prediction of the classifier in a format that can be read and understood by human end users. An example of such a rule is shown in Figure 1. Importantly, both tasks share the same encoder, and are trained using a joint objective function.

Supporting earlier findings, we observe that joint training leads to performance improvements both within and across tasks. In our unique pairing of tasks, however, we are able to shed light on an opaque process by generating rules that provide an interpretable distillation of an event classifier's decisions.

The major contributions of this paper are:

(1) A simple neural architecture for EE that jointly learns to extract events and explain its decisions. While here we investigate event extraction, we believe this approach is applicable to many other information extraction tasks.

(2) We extend a subset of the BioNLP 2013 GENIA

```
label:  Phosphorylation
pattern: |
  trigger =
     [lemma=/phosphorylation/ & !word=/(?i)^(de|auto)/]
  theme:Protein  =
     prep_of appos? /nn|conj_(and|or)|cc/{,2}
```

🟨 Label(s) to assign to a match.
🟦 Lexical constraints on the event's predicate.
🟥 `argName:ArgType`, where `ArgType` indicates the semantic category expected for this argument.

Figure 1: An example of an event extraction rule in the Odin language that extracts phosphorylation events driven by a nominal trigger ("phosphorylation"). The event's sole argument or `theme` (the phosphorylated protein) is identified through both semantic constraints (its type must be `Protein`), and syntactic ones (it must be attached to the trigger through a certain syntactic dependency pattern: a `prep_of` followed by an optional (`?`) appositive (`appos`), followed by up to two (`{,2}`) other dependencies, e.g., `nn`). This rule would extract a `Phosphorylation(PKC)` event from the text "...which includes the phosphorylation of PKC by...".

event extraction (Kim et al., 2013) dataset with a set of rules designed to extract and explain three of the GENIA biomedical events: protein *phosphorylation*, *localization*, and *gene expression*. The result is a parallel dataset that aligns some of the GENIA event labels with rules that extract them. We release this dataset[1] for reproducibility.

(3) We train and evaluate our approach on this dataset and demonstrate that: (a) our approach achieves reasonable event classification performance, despite the fact that it uses no syntactic or part-of-speech information; (b) it decodes explanations with high accuracy, e.g., with a BLEU overlap score between the generated rules and hand-written rules of up to 93%, and (c) most importantly, we show that MTL improves performance over the individual event classification task. To our knowledge, this is the first work that demonstrates that interpretability improves classification performance.

(4) Our approach can be easily extended to a semi-supervised setting, where we use the rules associated with the events of interest to extract additional training data with "silver" labels, i.e., where we use the rule predictions as training labels for the classifier. We show that despite the inherent noise in this process, the performance of our approach improves considerably in this semi-supervised setting.

[1]https://github.com/clulab/releases/tree/master/aclsrw2020-edin/

2 Related Work

Interpretability in machine learning is an area of active research involving a multitude of approaches. In this work, we focus on *post-hoc* interpretations that explain a model's output (Lipton, 2016).

A common theme of prior research in interpretable machine learning is producing a definite decision process (e.g., a decision tree) that preserves generalization. (Craven and Shavlik, 1996) explored converting a trained network to a decision tree. Similarly, (Frosst and Hinton, 2017) trained soft binary decision trees using the predictions of a neural model. These decision trees are trained with mini-batch gradient descent using as labels a trained network's results. In the same vein, (Che et al., 2016) proposed a mimic learning framework, which trains gradient boosting trees to mimic the soft predictions of the original neural network. One unaddressed challenge with this direction, however, is that a decision tree's interpretability tends to decay as the tree increases in size.

Rather than converting a statistical model into an interpretable model such as a decision tree, other efforts have focused on *jointly* learning a statistical model with explanations for the model's output. Our work falls in this camp as well. (Hendricks et al., 2016) proposed a system for image classification that generates a natural language (NL) explanation to accompany each decision. Similarly, (Blunsom et al., 2018) learned NL explanations for the natural language inference (NLI) task, and (Ye et al., 2018) applied this idea to crime case prediction. Inspired by such approaches, here we learn to generate declarative information extraction rules that serve to explain the predictions of an event classifier.

3 Approach

Our approach jointly addresses classification and interpretability through an encoder-decoder architecture, where the decoder uses MTL for event extraction (Task 1) and rule generation (Task 2). In this paper, we apply this architecture to the extraction of unary events in the biomedical domain. The two tasks are framed as follows:

Task 1 (T1): Given a sentence and an entity in focus, it must identify which event applies to the entity, and what is its trigger, i.e., the verbal or nominal predicates that drives the lexicalization of the event (e.g., "phosphorylation").

Task 2 (T2): Decode a rule in the Odin language that explains the prediction of the event classifier. That is, the rule should identify the lexical constraints on the event trigger, e.g., its lemma, the semantic type expected of the argument, e.g., that is must be a `Protein`, and the syntactic pattern that connects the event trigger with the argument (Figure 1 shows a complete example for such a rule).

Consider this text as a walkthrough example: *which includes the phosphorylation of* **PKC** *by* . . . , where the text in bold indicates the entity that is provided in the input in this task. This follows the settings of the standard event extraction task of BioNLP 2013 (Kim et al., 2013). For Task 1, we train a series of binary event classifiers (one for each event type), which predict the position of the event's lexical predicate (i.e., trigger) that modifies each given entity (*phosphorylation* here). Drawing upon the state information from Task 1, we prime our decoder in Task 2 using a contextualized representation of the predicted event trigger to generate an information extraction rule in the Odin language that captures the same event (i.e., entity-predicate structure) identified in Task 1 (see Figure 1). We detail these two tasks next.

3.1 Task 1: Event Classifier

We train a binary event classifier for each event type, which must identify if the corresponding event type applies to the entity under consideration, and, if so, which token in the input sentence is the event's trigger.

The classifier uses an encoder with entity attention to encode its input. For each sentence with words $w_1, \ldots, w_n$ and a given entity z, we associate each word i with a representation x_i that concatenates three embeddings: $x_i = e(w_i) \circ e(p_i) \circ char(w_i)$, where $e(w_i)$ is the word embedding of token i, p_i is the word's relative position to the $entity$ under consideration, and $char(w_i)$ is the output of a bidirectional character-level LSTM (charLSTM) applied over w_i. $e(w_i)$ is initialized with the pretrained embeddings of (Hahn-Powell et al., 2016) using the word2vec Skip-gram model (Mikolov et al., 2013) trained on the full text of over 1 million biomedical papers taken from the PubMed Central Open Access Subset.[2] while $e(p_i)$ and $char(w_i)$ are initialized randomly.

[2]`https://www.ncbi.nlm.nih.gov/pmc/tools/openftlist/`

The sequence of x_is serves as input to a sentence-level bidirectional LSTM (biLSTM), whose hidden states h_is serve as input to the attention layer below.

The entity-attention layer computes a sequence of context vectors (the matrix $\boldsymbol{C}$ in the equations below), which weighs the biLSTM's hidden states by their importance to the entity z. Our attention mechanism is inspired by the transformer network (Vaswani et al., 2017). Similarly, we compute the attention function on a set of keys and values that are packed together into matrices $\boldsymbol{K}$ and $\boldsymbol{V}$. The difference is that our approach is entity-focused in its query, so we only compute the attention on a single query vector $\boldsymbol{q}$. Further, unlike the conventional encoder in a transformer network, we don't produce a single vector, but a sequence of vectors (the matrix $\boldsymbol{C}$).

$$\boldsymbol{q} = \boldsymbol{W}_q \boldsymbol{h}_z \tag{1}$$

$$\boldsymbol{K} = \boldsymbol{W}_k \boldsymbol{H}^E \tag{2}$$

$$\boldsymbol{V} = \boldsymbol{W}_v \boldsymbol{H}^E \tag{3}$$

$$\boldsymbol{s} = \boldsymbol{q}\boldsymbol{K} \tag{4}$$

$$\boldsymbol{a} = \text{softmax}(\boldsymbol{s}) \tag{5}$$

$$\boldsymbol{C} = \boldsymbol{V} \odot \boldsymbol{a} \tag{6}$$

where $\boldsymbol{W}_q$, $\boldsymbol{W}_k$, $\boldsymbol{W}_v$ are learned matrices of dimension 200×200, $\boldsymbol{H}^E$ contains the biLSTM's hidden states, and $\boldsymbol{h}_z$ is the hidden state of the entity z from $\boldsymbol{H}^E$. We concatenate each context vector ($\boldsymbol{C}_i$) with the entity vector ($\boldsymbol{H}^E_i$) and feed the concatenated vector to two feedforward layers with a softmax function, and use its output to predict if there is a trigger in this position. We calculate the classifier's loss using the binary log loss function.

3.2 Task 2: Rule Decoder

Inspired by neural machine translation (Luong et al., 2015), we use another LSTM with attention as the decoder. To center rule decoding around the trigger, which must be generated first, we first feed the trigger vector from the encoder's context as the initial state in the decoder. Then, in each timestep t, we generate the attention context vector $\boldsymbol{C}^D_t$ by using the current hidden state of the decoder, $\boldsymbol{h}^D_t$:

$$\boldsymbol{s}_t(j) = \boldsymbol{C}^E_j \boldsymbol{W}^A \boldsymbol{h}^D_t \tag{7}$$

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

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

where $\boldsymbol{W}^A$ is a learned matrix of dimensions 100 $\times$ 200, and $\boldsymbol{C}^E$ are the context vectors from the previous entity-focused attention layer. Note that the learned matrix $\boldsymbol{W}^A$ here is distinct from the matrices learned in the previous entity-attention layer. We feed this $\boldsymbol{C}_t^D$ vector to a single feed forward layer that is coupled with a softmax function. We predict the next word from a vocabulary extracted from the existing Odin rules used in our experiments (see next section for details). During training, we calculate the decoder's loss using the multiclass cross-entropy loss function.

Note that the losses corresponding to these two tasks are jointly optimized. Formally, the loss function is defined as:

$$\text{loss} = \text{loss}_c + \text{loss}_d \tag{10}$$

$$\text{loss}_c = \sum_i -(t_i^c \log(y_i) + (1 - t_i^c)\log(1 - y_i)) \tag{11}$$

$$loss_d = \sum_i -\log(p_i) \tag{12}$$

where $loss_c$ is the cross-entropy loss of the event classifier, which relies on: t^c, the target label (i.e., 1 for positive examples, 0 for negative), and y, the likelihood predicted by the model. $loss_d$ is the cross-entropy loss of the rule decoder, where i iterates over the tokens in the rule, and p_i is the decoder's probability of the correct token at position i.

4 Experiments

4.1 Dataset

We train and evaluate on three events from the BioNLP 2013 GENIA Events extraction shared task (Kim et al., 2013): Phosphorylation (P), Localization (L), and Gene Expression (GE). To facilitate comparison with previous work, we use the standard training, development, and test partitions from the original dataset. To generate data for the rule decoder, we extend this dataset with rules from the rule-based system of (Valenzuela-Escárcega et al., 2018), which reported high-precision results for Phosphorylation (92%). We manually added new rules using existing syntactic templates that cover common syntactic forms of subject-verb-object patterns to cover more events. Further, because the system of Valenzuela-Escárcega et al. (2018) did not cover L and GE events, we extended it with rules for these two events. All in all, we used: 32, 20, and 21 rules for P, L, and GE, respectively. Most of these rules rely on syntactic structures denoted in terms of dependency paths to extract event arguments (see Figure 1 for an example of such a rule). From these rules, we obtained a token-level vocabulary for the rule decoder. This poses an additional challenge on our decoder, which must now decode from raw text both the semantics necessary for these events, and the syntactic patterns needed to match event arguments. Further, note that these rules do not have perfect recall, i.e., there are events in the data that are not covered by rules. In other words, the two tasks in our MTL framework are not perfectly aligned: there are data points which are part of the training examples of T1, but not of T2 (for those training examples, the loss of decoder is set to be 0).

In addition to using these rules for explainability, we used the rule-based system to generate additional "silver" training data for these three events, by using its extractions from a collection of PubMed publications. From these papers, we extracted an additional 6592, 6321, and 2056 positive training examples for P, L, and GE, respectively. To avoid biasing the classifier to the positive classes, we also generated 3467, 3532, and 2876 negative training examples for P, L, and GE by extracting entities assign to extract evented to other event types in the BioNLP data.

4.2 Evaluation Metrics

We used precision, recall, and F1 scores to measure the performance of the event extractor (classifier), and 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. Note that the BLEU score provides an incomplete evaluation of rule quality. The more complete solution would be to evaluate these rules by executing them over free text and verifying the quality of the extracted output. However, this is not a trivial process, as some of the decoded rules break the Odin syntax, and are only executable after a manual cleanup process. We leave this evaluation for future work.

	Phosphorylation (P)			Localization (L)			Gene Expression (GE)		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Rule baseline	92.68	48.12	63.35	66.13	44.44	53.16	51.08	69.79	58.98
T1	87.78	49.38	63.20	100.00	4.04	7.77	89.32	64.30	74.77
T1 + Silver	62.75	82.50	71.28	54.55	34.34	42.15	68.43	74.31	71.25
T1 + Silver + T2	84.38	68.75	75.77	76.60	39.39	52.03	69.92	71.24	70.58
BioNLP best	83.95	85.62	84.78	86.21	53.54	66.05	91.29	82.55	86.70
BioNLP median	79.83	81.57	80.64	88.55	40.91	55.89	82.58	78.11	80.09

Table 1: Results for the three events in the BioNLP 2013 test partition. T1 and T2 indicate the two tasks in our MTL approach, i.e., the event classifier and the rule decoder, respectively. Silver indicates that that configuration used the silver data created by the rule-based system (see §4.1). BioNLP best and median indicate the best/median results during the 2013 shared task. We do not include T1 + T2 results because in this configuration we observed that there is not sufficient data to train the decoder.

4.3 Baseline

We compared our proposed methods with the rule-based baseline proposed by (Valenzuela-Escárcega et al., 2018). They used their rule-based system to extract Phosphorylation events in BioNLP 2013 Genia Events (GE) task data using 42 manually written rules (which we extended for our experiments – see Section 4.1). On the development partition, they reported a precision of 92.9%, a recall of 56.0%, and an F1 score of 69.9%. We also evaluated their system on the formal test partition and obtained a precision of 84.2%, a recall of 43.8%, and an F1 score of 57.6%. As mentioned in Section 4.1, we adjusted the grammar in this system to cover gene expression and localization events. The complete results for this system are listed in Table 1 as "Rule baseline."

4.4 Results and Discussion

Tables 1 analyzes the performance of our approach for the three events, compared against the rule-based system described in §4.1. These results highlight several important observations:

(1) T1 by itself performs generally worse than the rule baseline and the median BioNLP result. This is caused by: (a) the small size of this dataset, e.g., there are only 117 training examples for P; and (b) the fact that our approach uses no part-of-speech (POS) or syntactic information, which have been shown to be important for this BioNLP task (Kim et al., 2013). However, adding the silver data improves T1 performance considerably, e.g., 35 F1 points for Localization. This demonstrates that our approach provides a simple but effective platform for semi-supervised learning.

(2) Most importantly, jointly training for classification and explainability helps the classification task (T1) itself. As shown in the tables, combining T1 and T2 generally improves F1 scores considerably, e.g., 4 F1 points for Phosphorylation and 10 for Localization. To our knowledge, this is the first NLP work to demonstrate that aiming for interpretability also helps the main task addressed. All in all, we approach the median performance in the shared task, a respectable result considering that our approach uses only raw text as input, whereas all participants in this shared task used some form of syntactic representation. Importantly, our approach outperforms considerably the rule-based method of (Valenzuela-Escárcega et al., 2018), which served as the starting point of this work (see Section 4.3).

	BLEU	Exact Matches	Non-exact, Explainable Matches
P	93.80	86.11	2/15
L	83.78	84.33	1/9
GE	78.99	76.45	10/43

Table 2: Evaluation of decoded rules, on the BioNLP development partition. BLEU measures the overlap with hand-written rules. Exact Matches shows the percentage of decoded rules that exactly match hand-written ones. Explainable Matches shows the number of decoded rules that do not match exactly hand-written ones, but were considered good explanations by human experts.

(3) The only negative results in our experiments are the GE results in the test partition, where T1 outperforms both T1 + Silver and T1 + Silver + T2. We hypothesize that this is caused by the larger training data for this event, e.g., there are 6 times more training samples for GE than P, which allows the T1 classifier to learn by itself, without the scaffolding offered by MTL, and the additional (noisy) data in the silver dataset. This suggests that our approach is best suited for EE scenarios with minimal training data, an important subset of information extraction tasks.

But are the decoded rules actually interpretable? To answer this, we compared in Table 2 the decoded rules against the hand-written rules that matched in the BioNLP development partition.

Hand-written Rule	Decoded Rule

trigger = [lemma = /phosphorylate/ & ! word = /(?i)^(de|auto)/ & tag = /^(V|JJ)/ & ! mention = ModificationTrigger]
 	
theme : BioChemicalEntity = > nsubjpass prep_of ? /conj_(and|or|nor)|nn|cc/ { , 2 }
	trigger = [lemma = /phosphorylate/ & ! word = /(?i)^(de|auto)/ & tag = /^(V|JJ)/ & ! = ModificationTrigger]
	
	theme : BioChemicalEntity = > nsubjpass prep_of ? /conj_(and|or|nor)|nn|cc/ { , 2 }

trigger = [lemma = /detect|localiz|locat|releas|secret|translocat/ & ! word = /(?i)^de/]
 	
theme : BioChemicalEntity = prep_of ? appos ? /conj_(and|or|nor)|cc|nn/ { , 2 }
	trigger = [lemma = /detect|localiz|locat|releas|secret|translocat/ & ! word = /(?i)^de/ & ! outgoing = /prep_(by|of)/]
	
	theme : BioChemicalEntity = < /conj_(and|or|nor)/ ? /conj_(and|or|nor)|cc|nn|prep_of/ { , 2 }

trigger = [lemma = / phosphorylation / & ! word = / (? i) ^ (de | auto) / & ! outgoing = / prep_ (by | of) /]
 	
theme : BioChemicalEntity = < / conj_ (and | or | nor) / ? / conj_ (and | or | nor) | cc | nn | prep_of / { 2 } site : Site ? = nn | < dobj ? / prep_ (at | on) / num ?
	trigger = [lemma = / phosphorylate / & ! word = / (? i) ^ (de | auto) / & tag = / ^ (V | JJ) / & ! mention = ModificationTrigger]
	
	cause : BioChemicalEntity ? = < xcomp ? (nsubj | agent | < vmod) / appos | nn | conj_ (and | or | nor) | cc / { 2 }
	
	theme : BioChemicalEntity = (dobj | xcomp) / conj_ (and | or | nor) | dep | cc | nn | prep_of / { 2 } (> > [word = by]) { 2 } site : Site ? = dobj ? / prep_ (at | on) | nn | conj_ (and | or | nor) | cc / { 2 }

Table 3: Examples of mistakes in the decoded rules. The first column shows hand-written rules, while the second shows the rules decoded by our approach from sentences where the corresponding hand-written rules matched. 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. The first row lists a partial mistake, which does not affect the interpretability of the decoded rule, since it only misses one token that can be inferred by the human experts from context. The second row lists a partial mistake, which impacts the semantics of the rule. For example, the decoder missed that the path between the trigger and the theme argument starts with an optional prop_of and appos. This rule was marked as partially correct because some simple syntactic patterns, e.g., nn, can still be correctly matched by the decoded rule. The last row lists a larger decoding error that was marked as completely incorrect by the annotator. For example, in the last decoded rule, the decoder generated an incorrect cause argument, which does not exist in the data, as well as an incorrect syntactic pattern for the theme argument, i.e., the protein being phosphorylated.

That is, we performed this analysis on the subset of the development partition, where each data point is accompanied by a matching hand-written rule. This reduced this dataset to approximately 60% of the total BioNLP development set. In particular, we analyzed 108, 82, and 296 event instances with matching rules for P, L, and GE events, respectively. The table shows that our rules have high BLEU overlap with hand-written rules, e.g., 93% for P, and, by and large, they exactly match them. We believe this is an exciting result, as it shows that our approach is able to decode directly from the raw text the declarative semantics necessary for the task, as well as the syntactic patterns that match the event arguments.

Lastly, Table 3 shows examples of typical decoding errors, ranging from partial mistakes that do not affect the interpretability of rules to complete decoding mistakes. As we mentioned above, we cannot ensure the validation of the generated rules with our current approach. Table 3 shows that this indeed happens in our output. For example, the decoder generates a binary operator such "!=" without the left operand (first row in the table).

5 Conclusions

We introduced an interpretable approach for event extraction that jointly trains an event classifier with a component that translates the classifier's decisions into interpretable extraction rules. We implemented this approach using an encoder-decoder architecture, where the decoder jointly optimizes the decoding of extraction rules and event classification. We evaluated the proposed approach on three biomedical events and demonstrated that the decoder generates interpretable rules, and that the joint training improves the performance of the event classifier. We also showed that the performance of our approach further improves when trained on automatically-labeled data generated by a rule-based system.

In the longer term, we envision a decoder with constraints, which enforces that the generated rules follow correct Odin syntax. We plan to include constraints as part of decoding to aid in rule synthesis. For example, in the Odin language, brackets must be paired to produce syntactically valid rules. This can be enforced with different strategies in the decoder, ranging from constrained greedy decoding to globally optimal solutions that could be implemented with integer linear programming. We suspect that including such validity constraints will further improve the quality of the decoded rules.

Further, we plan to use this decoder in an iterative, semi-supervised learning scenario akin to co-training (Blum and Mitchell, 1998). That is, the newly decoded, executable rules can be applied over large, unannotated texts to generate new training examples for the event classifier.

Acknowledgments

This work was supported by the Defense Advanced Research Projects Agency (DARPA) under grant #W911NF1810014. Mihai Surdeanu and Gus Hahn-Powell declare a financial interest in lum.ai. This interest has been properly disclosed to the University of Arizona Institutional Review Committee and is managed in accordance with its conflict of interest policies.

References

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM.

Phil Blunsom, Oana-Maria Camburu, Thomas Lukasiewicz, and Tim Rocktäschel. 2018. e- snli: Natural language inference with natural language explanations.

Zhengping Che, Sanjay Purushotham, Robinder Khemani, and Yan Liu. 2016. Interpretable deep models for icu outcome prediction. In *AMIA Annual Symposium Proceedings*, volume 2016, page 371. American Medical Informatics Association.

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

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

Gus Hahn-Powell, Dane Bell, Marco A. Valenzuela-Escárcega, and Mihai Surdeanu. 2016. This before that: Causal precedence in the biomedical domain. In *Proceedings of the 2016 Workshop on Biomedical Natural Language Processing*, pages 146–155. Association for Computational Linguistics.

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.

Jin-Dong Kim, Yue Wang, and Yamamoto Yasunori. 2013. The genia event extraction shared task, 2013 edition-overview. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 8–15.

Zachary C Lipton. 2016. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

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.

Marco A. Valenzuela-Escárcega, Özgün 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 10th International Conference on Language Resources and Evaluation*. LREC.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Hai Ye, Xin Jiang, Zhunchen Luo, and Wenhan Chao. 2018. Interpretable charge predictions for criminal cases: Learning to generate court views from fact descriptions. *arXiv preprint arXiv:1802.08504*.