Bootstrapping Neural Relation and Explanation Classifiers

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

1 Introduction

Recently Tang and Surdeanu (2022) proposed a supervised method that jointly trains a relation classifier (e.g., which extracts the relation per:city_of_birth between *John* and *London* in the sentence *John was born in London*) with an explanation classifier that identifies context words that are important for the relation at hand (e.g., *born* and *in* in the above example). One limitation of this method is that, similar to other neural approaches, it is data hungry. This is an important drawback for real-world applications where annotated data is expensive to obtain.

In this work, we expand this approach to semisupervised scenarios where the only supervision comes from a few example rules. In particular, our method iteratively converts the explanations produced by the above method into rules, and uses these rules to generate new "silver" annotations that are added to the training data. The specific contributions of this effort are:

(1) We introduce a novel semi-supervised neurosymbolic strategy for relation extraction that is explainable and requires minimal supervision. Our approach is neuro-symbolic because it relies on rules to explain the predictions of the neural relation classifier, and also to self-label training data.

(2) We evaluate this approach on the TACRED dataset (Zhang et al., 2017) and obtain competive results in a few-shot setting, where the only supervision comes from a small number of example rules. [1] Our experiments highlight several important observations. First, our approach outperforms the model that contains the seed rules by 15 F1 points, which validates the self-training direction. Second, our method performs considerably better than a sister approach that uses the relation classifier (rather than the rules generated from explanations) for self supervision. We hypothesize that this is because the neural classifier suffers more from the "curse of dimensionality" due to its large number of parameters and the small amount of training data than our rules, which are constrained to simple syntactic patterns. Third, our approach performs comparatively with prompt-based methods (Sainz et al., 2021; Zhang et al., 2022), even though our direction is simpler as it does not require a separate natural language inference component.

2 Related Work

For brevity, we focus our related work discussion on semi-supervised directions for information extraction that are closest to the proposed work: bootstrapping/self-training and recent prompt-based zero- or few-shot methods.

2.1 Bootstrapping/Self-Training

Typical bootstrapping methods iterate through three steps: (a) annotate seed data using a small amount of human supervision (e.g., rules for information extraction); (b) train a model with the available annotations, and, finally, (c) apply the model on unlabeled texts to produce new "silver" annotations (Abney, 2002). These approaches were popular before the deep-learning revolution. For example, Yarowsky (1995) used bootstrapping for word sense disambiguation; Riloff (1996) used it for dictionary

[1] We use an average of 7 rules per relation type in our experiments.

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 2: Short Papers, pages 48–56
July 9-14, 2023

acquisition; and Collins and Singer (1999) relied on bootstrapping for named entity classification. More recently, Gupta and Manning (2015) proposed a bootstrapping algorithm for named entity extraction that expands the set of known entities using word embeddings and k-nearest neighbor clustering. Eyal et al. (2021) used a syntactic search engine (Shlain et al., 2020) to bootstrap relation extraction. They also utilized natural language generation to further augment training data, which led to improved results. To our knowledge, we are the first to apply bootstrapping to a neuro-symbolic information extraction method, providing us both generalizability and explainability.

2.2 Prompt-based Zero- or Few-shot Learning

Recent large pre-trained language models (PLMs) with huge amount of parameters have showed the ability to handle NLP tasks with only a few examples or with prompts. Sainz et al. (2021) reformatted the relation extraction task as a natural language inference (NLI) task driven by a small set of manual templates. They obtained state-of-the-art results on the TACRED relation extraction dataset (Zhang et al., 2017) in both zero- and few-shot scenarios. The main limitation of this work is that it relies on a transformer-based NLI engine, which is not available in every domain. Wei et al. (2022) show that PLMs can perform multi-hop reasoning when using chain-of-thought prompts. Zhang et al. (2022) propose a prompt-based rule discovery and model boosting. However, Webson and Pavlick (2022) showed that the PLMs do not actually understand the prompt, which makes their decisions unreliable. Unlike the prompt-based approaches, our approach does not need the specific engine, e.g., for NLI, to perform the task. This gives us more flexibility in the choice of PLM and application domain.

3 Approach

Similar to traditional bootstrapping (Abney, 2002), our approach iteratively trains its classifier with the currently annotated data and applies the resulting model to the raw data to produce new annotations.

$R_0 \leftarrow R_{manual}$;

$D_{train} \leftarrow RuleExecutor(R_0, D_{raw})$;

for $i \leftarrow 1$ to N do

  $M_i \leftarrow f_{EC-RC}(D_{train})$;

  $P_i, E_i \leftarrow M_i(D_{train})$;

  $R_i \leftarrow RuleGenerator(P_i, E_i)$;

  $D_{train} \leftarrow$

  $D_{train} + RuleExecutor(R_i, D_{raw})$; end

Algorithm 1: Pseudo code of our training procedure. R_{manual} is the small set of seed rules; D_{raw} is the collection of unlabeled sentences. f_{EC-RC} is the joint explanation-relation classifier of Tang and Surdeanu (2022). M_i is the trained neural model in ith iteration, P_i and E_i are the M_i model's outputs (labels and explanations), and R_i is the set of new rules generated from M_i's outputs.

However, unlike traditional self training, which uses the classifier to annotate data, our approach converts the current model and data into rules, and uses the generated rules to annotate data. As we discuss in Section 4 this performs better empirically. Algorithm 1 shows the overall training procedure. We discuss the three key components below.

(1) Rule Executor: We use Odin (ValenzuelaEscárcega et al., 2016) system as our rule executor. Common rules in this paper are syntactic patterns that contain a lexical trigger (or predicate) and syntactico-semantic arguments. These rules can be summarized as if-this-then-that patterns, e.g.: if predicate=*born* and nsubj is PERSON and nmod_in is CITY then relation=per:city_of_birth. [2] The rule executor efficiently matches these patterns over the syntactic trees of sentences.

(2) Neural Model: Our approach utilizes the approach of Tang and Surdeanu (2022). It contains two main classifiers: a relation classifier (RC), and an explanation classifier (EC). The RC is a multiclass classifier that distinguishes between actual relation labels seen in training. The EC is a binary word-level classifier, which labels which words in the sentence are important for the relation at hand. For example, for the sentence *"[CLS] John was born in London."*,

[2] nsubj and nmod_in are syntactic dependencies that indicate nominal subject and indirect object attached to the verb through the preposition *in*, respectively.

the RC predicts a per:city_of_birth relation between *John* and *London*, and the EC identifies which words are critical for this relation (*born* and *in*). The EC and RC are trained jointly: the RC relies only on the hidden states of the context words identified by the EC (rather than, say, the [CLS] embedding); the EC is trained in a semi-supervised way, i.e., to maximize the probability of the correct RC label.

(3) Rule Generator: The rule generator has two major components: the generator and the filter. The generator takes the model output from the neural model above and produces rules by: (a) connecting the EC output to the trigger of the rule; (b) generating subject and object arguments that are connected to the trigger through the shortest syntactic dependency path, and (c) assigning the RC output (the label) to this syntactic pattern. The filter takes the rules produced by the generator, applies them to a validation set and evaluates their precision. If a rule's performance is below a certain threshold, the filter discards it.

3.1 Training Procedure

In iteration 0, we feed the seed rules R_0 to the rule executor which applies them on the unlabeled sentence set D_{raw}. These rules are a small set of rules written by human annotators. We add the rulematched data as seed annotations to the labeled data set D_{train} and remove them from D_{raw}.

In iteration i, we train the neural model M_i with all labeled data in D_{train} and use it to labeled the current D_{raw}. Then, we generate and filter the rules that explain the sentences in D_{raw} using the rule generator. Next, we feed the newly generated rules R_{i+1} to the rule executor, apply them over D_{raw}, and produced new labeled data, i.e., sentences with labeled relations. Lastly, we add the newly labeled data to D_{train} and remove the corresponding sentences from D_{raw}. We repeat this procedure until performance converges on a validation set.

4 Experimental Results

4.1 Data Preparation

We report results on the TACRED relation extraction (RE) dataset (Zhang et al., 2017). To mimic low-resource scenarios, we hide all gold labels from the training set. We keep only 1% of the development set labeled for tuning purposes. We use as seeds (R_0) the rule set from (Tang and Surdeanu, 2022), which is a combination of the surface patterns of Angeli et al. (2015), and syntactic rules written in the Odin language (ValenzuelaEscárcega et al., 2016), which were manually created by Tang and Surdeanu (2022). Overall, we use an average of 7 rules per relation type. Tang and Surdeanu (2022) indicated that these rules did not require considerable effort, i.e., they were developed by one of the authors within a few hours.

4.2 Baselines

We compare our results with four baselines: an extended version of the rule-based approach of Angeli et al. (2015), a typical self-training approach, a prompt-based RE approach based on natural language inference (NLI) (Sainz et al., 2021), and a prompt-based rule discovery and boosting approach (Zhang et al., 2022): Rule-based extraction: This baseline uses only the two sets of rules in our seed set (R_0): (a) the surface rules from (Angeli et al., 2015), which are executed in the Stanford CoreNLP pipeline (Manning et al., 2014); and (b) the syntactic rules of Tang and Surdeanu (2022), which are executed in the Odin framework.[3]

Self-training: This baseline is similar with our full method, with the exception that, in each iteration, we use the trained RC model to label new data rather than the generated rules.

NLI-prompt: (Sainz et al., 2021) reformulated the RE task as an entailment task driven by templates. They manually generated a number of verbalization templates for each relation in TACRED, e.g., the per:city_of_birth relation is verbalized as as {subj} was bom in {obj}, where {subj} and {obj} will be replaced with the entities in the given sentence. Thus, the sentence containing the relation to classify becomes the premise and the verbalized template the hypothesis. The RE task is then reduced to finding the best entailment template for the given sentence. no_relation is generated if no entailment score over a certain threshold is observed.

PRBOOST iteratively generates rules from prompting, asks a human expert to filter the rules,

[3] The rule set from (Angeli et al., 2015) also included some syntactic rules, but we found out that they only matched the simpler per:title relation, so we did not use them.

use the rules to generate new annotations, and, lastly, use the annotations to train a new model (Zhang et al., 2022).

4.3 Implementation and Evaluation Details

For our method we follow the same implementation details and hyper parameters as Tang and Surdeanu (2022). The only difference is that instead of using the full development set, we randomly select 1% from the TACRED development set for tuning, i.e., to decide which generated rules to keep, and to decide when the bootstrapping training procedure completes. For the former, we used 0.5 as the threshold; that is, if the precision of a rule is lower than the threshold, we discard that rule.

Approach	Precision	Recall	F1
Baselines			
Rules	85.82	24.21	37.77
Self-training	65.58	38.56	48.56
NLI-prompt[7]	55.46	52.09	53.72
PRBOOST	–	–	48.1
Our Approach			
Iteration 4	67.10	45.14	53.97

Table 1: Relation extraction results on the TACRED test partition. Iteration 4 was the best iteration in development.

For a fair comparison, for the NLI-prompt approach of Sainz et al. (2021) we chose their zeroshot scenario [4] and RoBERTa (Liu et al., 2019).[5][6] Further, to guarantee the same level of supervision, we converted our seed rules to their verbalization templates (see Appendix A for the conversion procedure). Lastly, we estimate their threshold for no_relation using the same validation dataset as our approach. We iterated from 0.1 to 0.9 with a step of 0.1, and observed the best validation results for a threshold of 0.8.

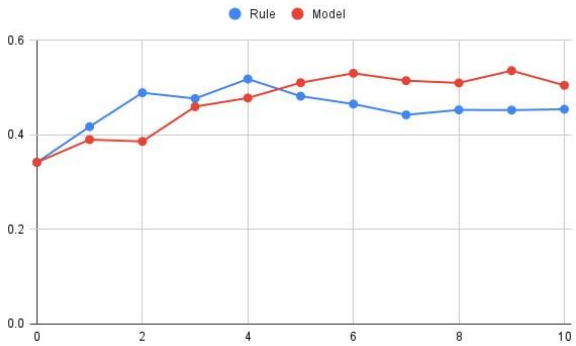

Figure 1: Learning curves of our approach (Rule) and traditional model-based self-training (Model), on validation.

4.4 Results and Discussion

Table 1 reports the overall performance of our approach and the four other methods. For PRBOOST we used the numbers reported in the corresponding paper. We draw the following observations:

(1) As expected, the rule-based baseline has high precision but suffers from low recall. In contrast, our best model that is bootstrapped from the same rules has 20% higher recall and 15% higher F1 (absolute). This indicates that the bootstrapping approaches popularized for information extraction several decades ago remain valid in the neural era.

(2) Our approach performs statistically significantly better than the traditional self-training approach that uses the relation classifier for self labeling (53.97 vs. 48.56 F1)[7]. The fact that rules perform better for self labeling than the actual neural model is somewhat surprising. Our hypothesis is that the neural model suffers more from overfitting due to its large number of parameters and the relatively small amount of training data. Rules generalize better (and thus produce better "silver" labels) because the simple syntactic patterns generated provide reduced

[4] This settings uses supervision from templates, similar to our seed rules.

[5] The NLI-prompt requires a fine-tuned NLI layer in the language model, which differs from SpanBERT, the LM used by Tang and Surdeanu (2022). We believe that RoBERTa is the closest alternative.

[6] Results using our templates and tuning. Sainz et al. (2021) reported a F1 score of 55.6 in their paper.

[7] We performed statistical significance analysis using nonparametric bootstrap resampling with 1000 iterations.

opportunities for overfitting. To validate this hypothesis we plot the learning curves of the two approaches on our validation partition in Figure 1. [8] These curves indicate that the best performance of our approach is in iteration 4, while the neural self-training continues to improve on validation until iteration 9. However, as shown in Figure 2, the performance of the modelbased self-training on test saturates after iteration 4, which suggests that, indeed, the neural self-training method suffers from overfitting.

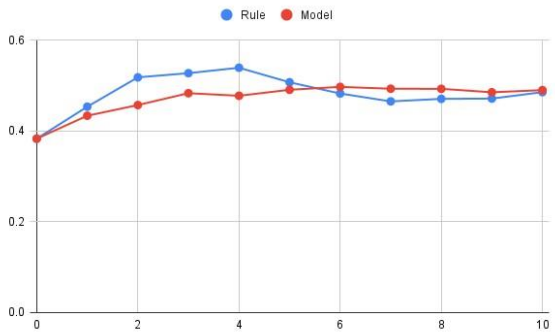

Figure 2: Learning curves of our approach (Rule) and traditional model-based self-training (Model), on test.

(3) Our method performs better than PRBOOST and similarly to the NLI-prompt method. This suggests that self-training, when carefully implemented, remains competitive with more modern alternatives such as prompt-based methods. More importantly, our approach is simpler, as it does not need the extra inference layers, e.g., the NLI classifier in the NLI-prompt approach.

4.5 Error Analysis

We conclude this section with a brief error analysis that compares our rule-based bootstrapping approach with the "traditional" neural-model-based self-training approach.

First, we conducted a comparative analysis of the annotations produced by the two methods after the first iteration. In this setting, both approaches were trained on the same seed annotations, which ensures a fair comparison. Out of all positive examples in training data (excluding the seed examples), our approach annotated 7.64% of them correctly, while self-training annotated only 3.70% correctly. Among these true positives produced by the neural bootstrapping, 75.68% of them are also annotated correctly by our approach. This indicates that our generated rules not only cover most of the neural model's annotations, but also correctly annotate more uncovered instances.

Sean Parker , a 17-year-old stu dent , was portraying a casualty clutching a head injury caused by a falling classroom fan . Gold label: per:title Rule label: per:title NN Model label: no_relation
Gilchrist teamed controversial Chris Simcox with, a Minuteman pub newspaper lisher in Tucson , Ariz. , to form the project , which drew nearly 900 volunteers to Arizona in April . Gold label: no_relation Rule label: per:title NN Model label: no_relation

Table 2: Example outputs from a per:title rule, The subject and object entities, which are provided in the task input, are highlighted in blue and orange. The important tokens for explainability identified by the various methods are highlighted in red.

Table 2 shows a case where the neural-modelbased self-training method falls short (first row in the table) and a case where bootstrapping does not seem to help (second row). These two cases are extracted by the same rule, in which the trigger words ", a" are used to connect SUBJ_PERSON and OBJ_TITLE entities through

[8] Appendix B contains a more detailed curve for our approach including precision, recall, and F1.

<punct or <punct appos syntactic dependencies. This rule matches 120 examples in the training set, 102 of which are true positive. Importantly, only 67 of the 120 examples are uncovered by the neural bootstrapping model, which highlights again the increased coverage of our rule-based method. Interestingly, while the label produced by the rule-based bootstrapping model for the second example in the table is technically wrong, in the opinion of the authors the gold label is incorrect here. This suggests that rules not only improve self-training, but have the potential to also improve the consistency of training data.

5 Conclusion

We introduced a method that self trains (or bootstraps) a neuro-symbolic approach for relation extraction that combines neural relation and explanation classifiers. Unlike traditional self-training, our approach uses rules for bootstrapping. In particular, our method iteratively converts the explainable models' outputs to rules and applies them to unlabeled text to produce new annotations.

We evaluated our approach in a low-resource scenario where there is no labeled data, and the only supervision comes from a small number of seed patterns. Our experiments showed that using rules in the bootstrapped training procedure is better than the typical self-training method that relies on neural model predictions. Further, we show that we obtain similar performance with prompt-based models for relation extraction without the additional NLI component required by such approaches.

Limitations

In this work we have tested our approach using SpanBERT, a relatively small model when compared to, say, DeBERTa_large or GPT. SpanBERT has been reported to obtain state-of-the-art performance for relation extraction (Joshi et al., 2020; Tang and Surdeanu, 2022), but it is unclear if a larger LM would improve this semi-supervised learning setting.

We use both surface patterns (in the tokensregex (Chang and Manning, 2014) format) and syntactic patterns (Odin (Valenzuela-Escárcega et al., 2016)) as training seeds, but our approach can only produce syntactic patterns as outputs. This is not ideal, since there is empirical evidence showing that the mixed representation for rules may provide better performance. For example, we can easily capture per_title relation with a surface rule such as *"{obj_title} {subj_person}"*, which simply looks for the two entities being adjacent.

Ethics Statement

This work did not involve human annotations, other than the set of rules used as seeds (Angeli et al., 2015; Tang and Surdeanu, 2022).

It is unlikely but possible that the automaticallygenerated rules we used during bootstrapping augment some unknown biases in the unlabeled data. In a brief analysis of the data we did not observe any such situations. However, this potential undesired side effect is important and should not be ignored in the eventual deployment of this method in real-world applications.

References

Steven Abney. 2002. Bootstrapping. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 360–367, USA. Association for Computational Linguistics.

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*.

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

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *1999 Joint SIGDAT conference on empirical methods in natural language processing and very large corpora*.

Matan Eyal, Asaf Amrami, Hillel Taub-Tabib, and Yoav Goldberg. 2021. Bootstrapping relation extractors using syntactic search by examples. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1491–1503, Online. Association for Computational Linguistics.

Sonal Gupta and Christopher D. Manning. 2015. Distributed representations of words to guide bootstrapped entity classifiers. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1215–1220, Denver, Colorado. Association for Computational Linguistics.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld,

Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland. Association for Computational Linguistics.

Ellen Riloff. 1996. Automatically generating extraction patterns from untagged text. In *AAAI/IAAI, Vol. 2*.

Oscar Sainz, Oier Lopez de Lacalle, Gorka Labaka, Ander Barrena, and Eneko Agirre. 2021. Label verbalization and entailment for effective zero and fewshot relation extraction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1199–1212, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Micah Shlain, Hillel Taub-Tabib, Shoval Sadde, and Yoav Goldberg. 2020. Syntactic search by example. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 17–23, Online. Association for Computational Linguistics.

Zheng Tang and Mihai Surdeanu. 2022. It Takes Two Flints to Make a Fire: Multitask Learning of Neural Relation and Explanation Classifiers. *Computational Linguistics*, pages 1–40.

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).

Albert Webson and Ellie Pavlick. 2022. Do promptbased models really understand the meaning of their prompts? In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2300–2344, Seattle, United States.
Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd annual meeting of the association for computational linguistics*, pages 189–196.

Rongzhi Zhang, Yue Yu, Pranav Shetty, Le Song, and Chao Zhang. 2022. Prompt-based rule discovery and boosting for interactive weakly-supervised learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 745–758, Dublin, Ireland. 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.

A From Rules to NLI Templates

To guarantee the same level of supervision between our approach and NLI-prompt, we converted our seed rules to their verbalization templates. To convert a rule to a verbalized template, we first apply the rule to the available texts, extract the shortest span that covers the trigger and the subject/object arguments, and extract this shortest text span as the verbalized template. The actual template is the span with entities replaced with placeholders *{subj}* and *{obj}*. For example, for the sentence *"[CLS] John was born in London."*, the shortest span is *"John was born in London"*, and the template will be *"{subj} was born in {obj}."*

B Learning Curve

Figure 3 shows the changes in precision, recall, and F1 scores over multiple iterations on the validation set. As shown, the recall and F1 are steadily increasing during this procedure. This is inspiring since it shows that our approach can help improve the generalizability of the neural model in the lowresource scenario. Further, we note that the drop in precision is the reason the F1 score stops improving after iteration 4. However, this is solvable since our annotations are from the rules and there are ways to control the quality of the rules other than just filtering out the low precision ones. We leave this analysis as future work.

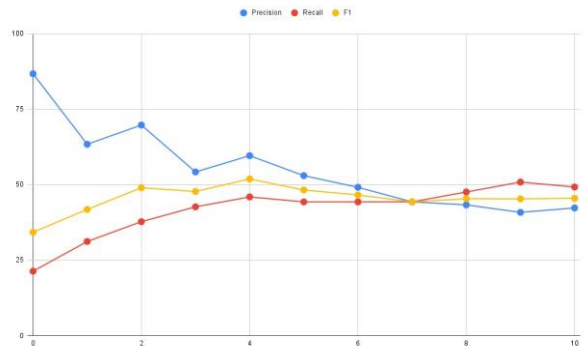

Figure 3: Learning curve of our approach in development. The X axis indicates the bootstrapping iterations.

C Experimental Details

We follow the same details from Tang and Surdeanu (2022)'s experiments. Table 3 shows the hyperparameter details for training.

Number of iterations	10
Number of epochs	20
Learning rate	1e-5
Dropout rate	0.1
Batch size	32
Max sequence length	128
Scheduler	Linear with warm-up

Table 3: Hyperparameter details for training.

ACL 2023 Responsible NLP Checklist A For every submission:

3 A1. Did you describe the limitations of your work? *At the beginning of page 5.*

3 A2. Did you discuss any potential risks of your work? *Discussed in the limitations section.*

3 A3. Do the abstract and introduction summarize the paper's main claims? *abstract and section 1*

7 A4. Have you used AI writing assistants when working on this paper? *Left blank.*

B 7 Did you use or create scientific artifacts? *Left blank.* B1. Did you cite the creators of artifacts you used?

No response. B2. Did you discuss the license or terms for use and / or distribution of any artifacts?

No response.

B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)? *No response.*

B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
No response.

B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.? *No response.*

B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be. *No response.*

C 3 Did you run computational experiments? *section 4*

3 C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used? *In appendix*

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

3 C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values? *In appendix*

3 C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run? *Section 4*

3 C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 3.

D 7 Did you use human annotators (e.g., crowdworkers) or research with human participants? *Left blank.*

D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.? *No response.*

D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)? *No response.*

D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used? *No response.* D4. Was the data collection protocol approved (or determined exempt) by an ethics review board? *No response.*

D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?

No response.