Weakly Supervised Named Entity Tagging with Learnable Logical Rules

Jiacheng Li[1]*, **Haibo Ding**[2], **Jingbo Shang**[1], **Julian McAuley**[1], **Zhe Feng**[2]
University of California, San Diego[1]
Bosch Research North America[2]
{j9li,jshang,jmcauley}@eng.ucsd.edu[1]
{haibo.ding, zhe.feng2}@us.bosch.com[2]

Abstract

We study the problem of building entity tagging systems by using a few rules as weak supervision. Previous methods mostly focus on disambiguating entity types based on contexts and expert-provided rules, while assuming entity spans are given. In this work, we propose a novel method TALLOR that bootstraps high-quality logical rules to train a neural tagger in a fully automated manner. Specifically, we introduce compound rules that are composed from simple rules to increase the precision of boundary detection and generate more diverse pseudo labels. We further design a dynamic label selection strategy to ensure pseudo label quality and therefore avoid overfitting the neural tagger. Experiments on three datasets demonstrate that our method outperforms other weakly supervised methods and even rivals a state-of-the-art distantly supervised tagger with a lexicon of over 2,000 terms when starting from only 20 simple rules. Our method can serve as a tool for rapidly building taggers in emerging domains and tasks. Case studies show that learned rules can potentially explain the predicted entities.

1 Introduction

Entity tagging systems that follow supervised training, while accurate, often require a large amount of manual, domain-specific labels, making them difficult to apply to emerging domains and tasks. To reduce manual effort, previous works resort to manual lexicons (Shang et al., 2018b; Peng et al., 2019) or heuristic rules provided by domain experts (Fries et al., 2017; Safranchik et al., 2020; Lison et al., 2020b) as weak supervision. For example, LinkedHMM (Safranchik et al., 2020) can achieve performance close to supervised models using 186 heuristic rules in addition to a lexicon of over two million terms. However, it is challenging for experts to write complete and accurate rules or lexicons in emerging domains, which requires both a significant amount of manual effort and a deep understanding of the target data. How to build accurate entity tagging systems using less manual effort is still an open problem.

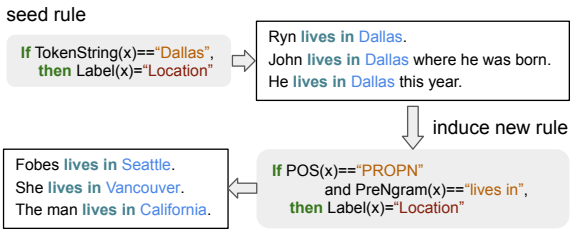


Figure 1: Examples of a seed logical rule and a newly induced rule from labeled data for recognizing locations. 'x' denotes a token span from a given sentence.

In this work, we explore methods that can automatically learn new rules from unlabeled data and a small set of seed rules (e.g. 20 rules). Such methods are desirable in real-world applications not only because they can be rapidly deployed to new domains or customized entity types, but also because the learned rules are often effective, interpretable, and simple for non-experts to "debug" incorrect predictions. As explained in Figure 1, new rules can be learned from seed rules. Specifically, we propose a novel iterative learning method TALLOR that can learn accurate rules to train a neural tagger in an automated manner, with goal to address two key issues during learning process: (1) how to detect entity boundaries and predict their types simultaneously with rules, (2) how to generate accurate and diverse pseudo labels from rules.

With such a small set of seed rules as supervision, previous works (Niu et al., 2003; Huang and Riloff, 2010; Gupta and Manning, 2014) only focus on disambiguating entity types assuming entity spans are given or just syntactic chunks (e.g., noun phrases).

*Work done during an internship at Bosch Research.

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 4568–4581
August 1–6, 2021.

	Noun phrase			TALLOR		
	P	R	F_1	P	R	F_1
BC5CDR	17.1	50.1	25.5	69.8	67.8	68.7
CHEM	3.2	35.6	5.8	63.0	60.2	61.6
CoNLL	4.1	47.3	7.5	86.9	86.7	86.8

Table 1: Boundary detection performance from our method and parsing based noun phrases.

However, we find that syntactic chunks often do not align well with target entity spans. For example, given a sentence from CoNLL2003: "*Germany's representative to the European Union's veterinary committee...*", the noun phrases[1] are "*Germany's representative*" and "*the European Union's veterinary committee*", but gold entities in the sentence are "*Germany*" and "*European Union*". We used noun phrases extracted from spaCy as predicted entity boundaries and compared them with ground truth entity boundaries, which are extracted based on the results from syntactic parsing. This setting of using noun phrases as entity candidates is similar to previous work (Niu et al., 2003; Huang and Riloff, 2010). The results are shown in Table 1, a majority of target entities are missed if we use noun phrases as entity candidates, which will not be recognized correctly later.

To address both entity boundary detection and type classification simultaneously, we first define five types of simple logical rules considering the lexical, local context, and syntax information of entities. We notice that simple logical rules are often inaccurate when detecting entity boundaries. Therefore, we propose to learn compound logical rules, which are composed from multiple simple rules and logical connectives (e.g. "and"). For example, given the sentence "*John lives in Dallas where he was born*", the simple rule "lives in _", which is a preceding context clue, will match multiple token spans such as "*Dallas*", "*Dallas where*", "*Dallas where he*" etc. In contrast, compound logical rules can both detect entity boundaries and classify their types accurately. For example, using both the preceding context and the part-of-speech (POS) tag rule (e.g. "lives in _" and POS is a proper noun) can correctly identify the Location entity "Dallas".

Though we aim to learn accurate rules, automatically acquired rules can be noisy. To ensure the quality of generated pseudo labels, we design a dynamic label selection strategy to select highly accurate labels so that the neural tagger can learn new entities instead of overfitting to the seed rules. Specifically, we maintain a high-precision label set during our learning process. For each learning iteration, we first automatically estimate a filtering threshold based on the high-precision set. Then, we filter out low-confidence pseudo labels by considering both their maximum and average distances to the high-precision set. Highly confident labels are added into the high-precision set for the next iteration of learning. Our dynamic selection strategy enables our framework to maintain the precision of recognized entities while increasing recall during the learning process, as shown in our experiments.

We evaluate our method on three datasets. Experimental results show that TALLOR outperforms existing weakly supervised methods and can increase the average F_1 score by 60% across three datasets over methods using seed rules. Further analysis shows that TALLOR can achieve similar performance with a state-of-the-art distantly supervised method trained using 1% of the human effort[2]. We also conduct a user study concerning the explainability of learned logical rules. In our study, annotators agree that 79% (on average over three annotators) of the matched logical rules can be used to explain why a span is predicted as a target entity.

In summary, our main contributions are:

- We define five types of logical rules and introduce compound logical rules that can accurately detect entity boundaries and classify their types. Automatically learned rules can significantly reduce manual effort and provide explanations for entity predictions.
- To effectively learn rules, we propose a novel weakly supervised method with a dynamic label selection strategy that can ensure the quality of pseudo labels.
- We conduct experiments on both general and domain-specific datasets and demonstrate the effectiveness of our method.

2 Tagging with Learned Logical Rules

We study named entity tagging under a weakly supervised setting, and propose TALLOR (**Ta**gging with **L**earnable **Lo**gical **R**ules) to build a tagger with only a small set of rules. Compared with previous work, our framework requires less human

[1]Noun phrases are extracted using spaCy noun chunks.

[2]In experiments, our method used 20 rules, the other system used a manually constructed lexicon of over 2000 terms.

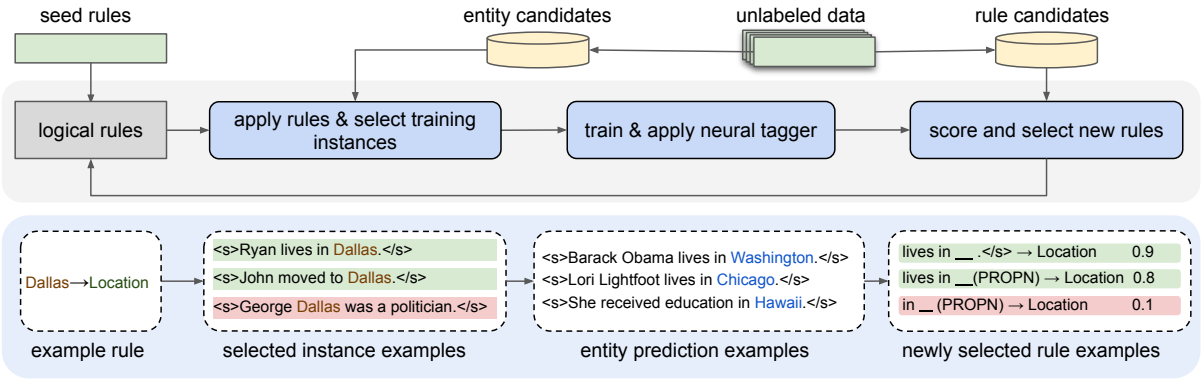


Figure 2: Overview of our tagging framework with logical rules and examples for each step.

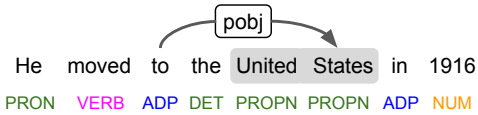
effort via the use of learned rules; we also show that these rules can be used to explain tagging results. Instead of treating tagging as a sequence labeling task, we formulate tagging as a span labeling task, in which named entities are modeled as spans over one or more tokens. With this setting, logical rules can easily be used for labeling entities.

Overview Figure 2 shows the flow of our iterative learning framework, which consists of the following components. First, we generate all entity candidates and rule candidates from unlabeled data. Then, for each iteration, we apply logical rules to the unlabeled data and select a set of high-quality weak training examples. Next, we train a neural tagger with the selected training examples and predict the labels of unlabeled data using the trained model. Finally, we select new accurate logical rules from candidate rules using the predictions. The newly learned rules will further be used to obtain weak training labels for the next iteration.

2.1 Logical Rule Extraction

In our work, a logical rule is defined in the form of "*if* p *then* q" (or "$p \rightarrow q$").[3] For entity tagging, q is one of the target entity classes, and p can be any matching logic. For example, "if a span's preceding tokens are '*lives in*', then it is a Location". We design the following five types of simple logical rules to consider the lexical, local context, and syntax information of an entity candidate.

Simple Logical Rules. A simple logical rule is defined as a logical rule that contains a single condition predicate. We design the following five predicates to represent common logical conditions. Given a candidate entity, (1) `TokenString` matches its lexical string; (2) `PreNgram` matches its preceding context tokens; (3) `PostNgram` matches its succeeding context tokens; (4) `POSTag` matches its part-of-speech tags; (5) `DependencyRel` matches the dependency relations of its head word.



Given a candidate entity "*United States*" in the above example, we can extract the following example logical rules for recognizing Locations:[4]

`TokenString`=="*united state*" → Location,
`PreNgram`=="*move to the*" → Location,
`PostNgram`=="*in 1916*" → Location,
`POSTag`=="PROPN PROPN" → Location,
`DependencyRel`=="*to*" (via pobj) → Location.

More details about extraction of each condition predicate are included in Appendix A.1.

Compound Logical Rules. A compound logical rule is formed with multiple condition predicates and logical connectives including `and` ($\wedge$), `or` ($\vee$), and `negation` ($\neg$). In this work, we focus on learning compound logical rules connected with conjunctions ($\wedge$) to recognize entities precisely, because simple logical rules are often insufficient to identify entity boundaries. In the above example, the rule `PreNgram`=="*move to the*" can match multiple candidates such as "*United*", "*United States*", and "*United States in*" etc., of which many are inaccurate. However, with a compound rule, e.g. `PreNgram`=="*move to the*" $\wedge$ `POSTag`=="PROPN PROPN", we can correctly recognize that "*Unitied States*" is a Location.

[3] "heuristic rules" and "labeling rules" can also be converted to logical rules, so they can be used interchangeably.

[4] All words in rules are lower-case and lemmatized.

We enumerate and extract all possible logical rules from unlabeled data based on our pre-defined rule types before the training process.

2.2 Applying Logical Rules

At each iteration, we apply both seed and learned logical rules to unlabeled entity candidates to obtain a set of weakly labeled instances. In case an entity candidate is matched by multiple rules (potentially conflicting), we use the majority vote as the final weak label.

Entity Candidates. In this work, we treat tagging as a span labeling task as described earlier. Before our learning process, we enumerate all token spans up to a maximum length from unlabeled data as entity candidates. We also notice that common phrases (e.g., "*United States*") are rarely split into different entities (e.g. "*United*", "*States*"). Therefore, we generate a list of common phrases using the unsupervised AutoPhrase method (Shang et al., 2018a) and merge two continuous spans together as a single entity candidate if they can form a common phrase.

2.3 Dynamic Training Label Selection

After applying the learned rules to unlabeled data, some of the weakly generated labels can be incorrect, which will lead to poor performance of our neural tagger in the next step. To filter out noisy labels, we propose to maintain a high-precision entity set to keep the accurately labeled training examples from each iteration.

Inspired by Zhang et al. (2020), we design a method to select high-quality labels from weakly generated labels by seed logical rules into the high-precision set. Specifically, given an entity category i, its corresponding high-precision set H_i, and a weakly labeled instance e_q, we first compute a confidence score of e_q belonging to category i by considering both its maximum pair similarity to the high-precision set H_i (called **local score**) and its average similarity to H_i (called **global score**). Then, the weakly labeled instance e_q will be selected into the high-precision set if its confidence score is larger than a threshold that is also estimated based on the high-precision set.

Instance Embedding. We compute the embedding of an entity instance as the mean of the embeddings of its tokens. A token's embedding is computed as the average of the first three layers' outputs from a pre-trained language model [5].

Local Score. Given a weakly labeled instance e_q and an example e_i from the high-precision set, we first compute their similarity as the cosine score between their embeddings. Then, we compute the local confidence score of e_q belonging to category i as the maximum of its similarities between all examples in the high-precision set.

Global Score. The local score is estimated based on a single instance in the high-precision set. Though it can help explore new entities, it can also be inaccurate in some cases. Therefore, we propose to compute a more reliable score to estimate the accuracy of an instance e_q belonging to a category i, which is called the global score. Specifically, we first sample a small set E_s from the high precision set H_i and then compute the prototypical embedding $\mathbf{x}_{E_s}$ of E_s as the average of embeddings of all instances in E_s. In our work, we sample N times and compute the global score as:

$$\text{score}_i^{glb} = \frac{1}{N} \sum_{1 \le j \le N} \cos(\mathbf{x}_{E_s}^j, \mathbf{x}_{e_q}) \quad (1)$$

To balance the exploration ability and reliability, we compute the final confidence score of a weakly labeled instance belonging to a category as the geometric mean of its local and global scores.

Dynamic Threshold Estimation. We hypothesize that different categories of entities may have different thresholds for selecting high-quality weak labels. We may also need to use different thresholds at different iterations to dynamically balance exploration and reliability. For example, we may expect our learning process to be reliable at earlier iterations and be exploratory at later stages. Motivated by this hypothesis, we propose to use a dynamic threshold to select high-quality weak labels. Specifically, we hold out one entity instance in the high precision set and compute its confidence score with respect to the rest of the examples in the high-precision set. We randomly repeat T times and use the minimum value as the threshold. For category i, it is calculated as:

$$\text{threshold} = \tau \cdot \min_{k \le T, e_k \in H_i} \text{score}_i(e_k) \quad (2)$$

where e_k is the held-out entity instance and $\tau \in [0, 1]$ is a temperature to control the final threshold.

[5] We used different pre-trained language models for different domains. Details are in Section 3.1.

2.4 Neural Tagging Model

Following Jiang et al. (2020), we treat tagging as a span labeling problem. The key idea is to represent each span as a fixed-length embedding and make predictions based on its embedding. Briefly, given a span and its corresponding sentence, we first initialize all tokens in a sentence using a pre-trained language model, and then apply a Bi-LSTM and Self-Attention layer, and obtain the contextual embedding of the sentence. Finally, we compute the span embedding by concatenating two components: a *content representation* calculated as the weighted average across all token embeddings in the span, and a *boundary representation* that concatenates the embeddings at the start and end positions of the span. Then, we predict the label of a span using a multilayer perceptron (MLP). For our detailed formulation please refer to Appendix A.2.

2.5 Logical Rule Scoring and Selection

Every iteration, we first predict the labels of all text spans using our neural tagging model. Then, we rank and select the 70%[6] most confident spans per category based on their prediction probabilities from the tagging model as weak labels for computing rule scores. We select new rules from rule candidates based on their confidence scores. We adopt the *RlogF* method (Thelen and Riloff, 2002) to compute the confidence score of a rule r:

$$F(r) = \frac{F_i}{N_i} \log_2(F_i) \tag{3}$$

where F_i is the number of spans predicted with category label i and matched by rule r, and N_i is the total number of spans matched by rule r. Intuitively, this method considers both the accuracy and coverage of rules because $\frac{F_i}{N_i}$ is the accuracy of the rule and $\log_2(F_i)$ represents the rule's ability to cover more spans.

In our experiments, we select the top K rules for each entity class per iteration. We increase K by η per iteration to be more exploratory in later iterations. We also use a threshold θ of rule accuracy (i.e. $\frac{F_i}{N_i}$) to filter out noisy rules. This method allows a variety of logical rules to be considered, yet is precise enough that all logical rules are strongly associated with the category.

[6]Different categories and datasets may require different thresholds to select high-quality labels. Setting a percentage means we will have dynamic thresholds for different categories so that the model will be robust to different categories and domains.

3 Experiments

We first compare our method with baselines on three datasets and further analyze the importance of each component in an ablation study. We also report the performance of our method with different numbers of seed rules and at different iterations. Finally, we show an error analysis and present a user study to analyze how many logical rules can be used as understandable explanations.

3.1 Experimental Setting

We evaluate our method on the following three datasets. Note that we use each training set **without** labels as our unlabeled data.

BC5CDR (Li et al., 2016) is the BioCreative V CDR task corpus. It contains 500 train, 500 dev, and 500 test PubMed articles, with 15,953 chemical and 13,318 disease entities.

CHEMDNER (Krallinger et al., 2015) contains 10,000 PubMed abstracts with 84,355 chemical entities, in which the training/dev/test set contain 14,522/14,572/12,434 sentences respectively.

CoNLL2003 (Sang and Meulder, 2003) consists of 14,041/3,250/3,453 sentences in the training/dev/test set extracted from Reuters news articles. We use Person, Location, and Organization entities in our experiments.[7]

Seed Rules and Parameters. In our experiments, we set the maximum length of spans to 5, and select the top $\mathrm{K} = 20$ rules in the first iteration for BC5CDR and CoNLL2003, and $\mathrm{K} = 60$ for the CHEMDNER dataset. Since it is relatively easy for users to manually give some highly accurate `TokenString` rules (i.e., entity examples), we use `TokenString` as seed rules for all experiments. To be specific, we manually select 20 highly frequent `TokenString` rules as seeds for BC5CDR and CoNLL2003 and 40 for CHEMDNER because of its large number of entities. The manual seeds for each dataset are shown in Appendix A.7. For pre-trained language models, we use BERT (Devlin et al., 2019) for CoNLL2003, and SciBert (Beltagy et al., 2019) for BC5CDR and CHEMDNER. All our hyperparameters are selected on dev sets. More setting details are in Appendix A.4.

[7]We do not evaluate on Misc category because it does not represent a single semantic category, which cannot be represented with a small set of seed rules.

Methods	BC5CDR			CHEMDNER			CONLL2003		
	Precision	Recall	F_1	Precision	Recall	F_1	Precision	Recall	F_1
Seed Rules	**94.09**	3.81	7.33	**91.60**	13.19	23.07	**95.77**	2.76	5.36
LinkedHMM	10.18	15.60	12.32	23.99	10.77	14.86	19.78	31.51	24.30
HMM-agg.	43.70	21.60	29.00	49.60	18.40	26.80	52.00	8.50	14.60
CGExpan	40.96	24.75	30.86	45.70	25.58	32.80	55.97	28.7	37.95
AutoNER	42.22	30.66	35.52	66.83	27.59	39.05	32.07	5.98	10.08
Seed Rules + Neural Tagger	78.33	21.60	33.86	84.18	21.91	34.78	72.57	24.68	36.83
Self-training	73.69	29.55	42.19	85.06	20.03	32.42	72.80	24.83	37.03
Our Learned Rules	79.29	18.46	29.94	69.86	21.97	33.43	65.51	21.12	31.94
Ours w/o Autophrase	74.56	32.93	45.68	67.74	55.99	61.31	71.37	25.50	37.57
Ours w/o Instance Selection	58.70	63.37	60.95	42.64	48.25	45.27	58.51	58.8	58.65
TALLOR	66.53	**66.94**	**66.73**	63.01	**60.18**	**61.56**	64.29	**64.14**	**64.22**

Table 2: Performance of baselines (in upper section), our method and its sub-components (in lower section).

3.2 Compared Baseline Methods

Seed Rules. We apply only seed rules to each test set directly and evaluate their performance.

CGExpan (Zhang et al., 2020) is a state-of-the-art lexicon expansion method by probing a language model. Since `TokenString` seed rules can be viewed as a seed lexicon, we expand its size to 1,000 using this method and use them as `TokenString` rules. We apply the top 200, 500, 800, and 1,000 rules to test sets and report the best performance.

AutoNER (Shang et al., 2018b) takes lexicons of typed terms and untyped mined phrases as input. We use the best expanded lexicon from CGExpan as typed terms, and both of the expanded lexicon and the mined phrases from AutoPhrase (Shang et al., 2018a) as untyped mined phrases. For detailed information on the AutoNER dictionary, refer to Appendix A.6

LinkedHMM (Safranchik et al., 2020) introduces a new generative model to incorporate noisy rules as supervision and predict entities using a neural NER model. In our experiments, we use the expanded lexicon by CGExpan as tagging rules and AutoPhrase mined phrases as linking rules.

HMM-agg. (Lison et al., 2020a) proposes a hidden Markov model to first generate weak labels from labeling functions and train a sequence tagging model. We convert the expanded lexicon by CGExpan to labeling functions and report results of the tagging model.

Seed Rule + Neural Tagger. This method is our framework without iteration learning. After applying seed rules, we use the weakly generated labels to train our neural tagger and report the result of the tagger.

Self-training. We first obtain weak labels by applying seed rules. Then, we build a self-training system using the weak labels as initial supervision and our neural tagger as the base model.

Methods (Fries et al., 2017; Ratner et al., 2017; Huang and Riloff, 2010) which use noun phrases as entity candidates are not included here because noun phrases have poor recall on the three datasets as shown in Table 1. CGExpan outperforms other entity set expansion methods (e.g., Yan et al. (2019)) so we use CGExpan as our baseline for automatic lexicon expansion.

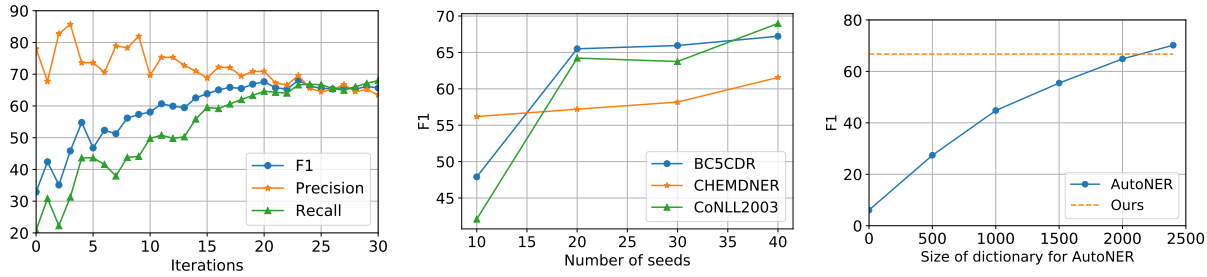
3.3 Performance of Compared Methods

We present the precision, recall, and micro-averaged F_1 scores on three datasets in Table 2. Results show that our method significantly outperforms baseline methods obtaining an average of 24-point F_1 improvement across three datasets over the best baseline.

We see that the precision of our seed rules is high, but the recall is lower. The lexicon expansion method (CGExpan) can recognize more entities but also introduces errors resulting in an improvement to recall but a dramatic decrease in precision.

Existing weakly supervised methods (i.e., AutoNER, LinkedHMM and HMM-agg.) cannot recognize entities effectively with either seed rules or expanded rules by CGExpan. These methods require a high-precision lexicon as input; however, the precision of the automatically expanded lexicon is not sufficient to meet this requirement. Though seed rules are very accurate, they lack coverage of various entities.

Our method without iteration (*Seed Rules + Neural Tagger*) and self-training can achieve high precision because of the accurate pseudo labels generated from seed rules. It is interesting to note that



(a) (b) (c)

Figure 3: (a) Iterations vs. performance of our method on BC5CDR. (b) Performance with different numbers of seed rules. (c) Performance of AutoNER with different sizes of manual lexicon and our method on BC5CDR.

the self-training method based on our neural tagger also achieved low recall. We hypothesize that this is mainly due to the neural tagger overfitting the small set of labels from seed rules.

Ablation Study. We also performed an ablation study to analyze the importance of some components in our framework, and report the performance in Table 2 (the lower section). Results show that our learned rules are accurate but lack coverage. Without using common phrases mined by Autophrase (i.e., *Ours w/o Autophrase*), our method achieves dramatically lower recall demonstrating the effectiveness of common phrases for improving coverage. Without high-quality training instance selection (*Ours w/o Instance Selection*), the precision is lower than our best method indicating the importance of the instance selection step.

3.4 Analysis of Learning Iterations and Seeds

Performance vs. Iterations. Figure 3a shows the performance of our method at different iterations. We see that our method improves the recall from 20% to over 60% during the learning process with a slight decease in precision, and achieves the best F_1 score after 25 iterations. Results on other two datasets show the same trend (in Appendix A.8).

Performance with Different Numbers of Seeds. Figure 3b shows the performance of our method using different numbers of manually selected seed rules on three datasets. We see that our method can achieve continuous improvement using more seeds. We also notice that our method can achieve over 55% F_1 on CHEMDNER with only 10 seeds demonstrating the effectiveness of our framework under minimal supervision setting. Our method obtains significantly better results (around 65% F_1) when using 20 seeds than 10 seeds on BC5CDR and CoNLL indicating that 20 seeds is a reasonable starting point for building a tagging system without much manual effort.

3.5 Comparison with Distant Supervision

AutoNER (Shang et al., 2018b) is a distantly supervised method using a manually created lexicon as supervision. We also compared our method to this method to figure out how many terms we need to manually created for AutoNER to achieve similar performance with our method. We conducted experiments on BC5CDR and used only 20 seeds for our method. For AutoNER, we used additional M terms from a manually created lexicon (Shang et al., 2018b)[8]. Figure 3c shows the performance with different values of M. Results show that AutoNER needs an additional $\sim$ 2000 terms to achieve similar performance (around 66% F_1) with our method, which demonstrates that our method is effective under minimal supervision without access to a large manual lexicon.

3.6 Analysis of Rule Selection Strategies

In our work, we designed three rule selection strategies: (1) *entity type* selects the top K rules for each entity category; (2) *rule type* selects the top K rules for each logical rule type; (3) *entity&rule type* selects the top K rules for each entity category and logical rule type. Results in Table 4 show that entity type based selection achieves the best performance.

3.7 Error Analysis of Learned Logical Rules

We show the statistics of different types of rules learned after all iterations in Table 5.[9] We see that TokenString rule is the most rule type for domain-specific datasets (BC5CDR and CHEMDNER). For

[8] AutoNER authors compiled the lexicon from MeSH database and CTD Chemical and Disease vocabularies, which are manually created by experts.

[9] TokenStr, Pre, Post, POSTag, and Dep are short for TokenString, PreNgram, PostNgram, POSTag, and DependencyRel.

Examples	Predicted Labels	Gold Label
Error Type: Similar Semantic Concepts (56%)		
The aim of this work is to call attention to the risk of tacrolimus use in patients with SSc.	Disease	NotEntity
We recorded time to first dysrhythmia occurrence , respective times to 25 % and 50 % reduction of the heart rate (HR) and mean arterial pressure , and time to asystole and total amount of bupivacaine consumption.	Disease	NotEntity
The severity of pain due to etomidate injection , mean arterial pressure , heart rate , and adverse effects were also evaluated.	Disease	NotEntity
Error Type: Inaccurate Boundary (20%)		
Furthermore ameliorating effect of crocin on diazinon induced disturbed cholesterol homeostasis was studied.	Disease	Disease
Pretreatment with S. virgaurea extract for 5 weeks at a dose of 250 mg / kg followed by isoproterenol injection significantly prevented the observed alterations.	Chemical	Chemical
This depressive -like profile induced by METH was accompanied by a marked depletion of frontostriatal dopaminergic and serotonergic neurotransmission , indicated by a reduction in the levels of dopamine , DOPAC and HVA , tyrosine hydroxylase and serotonin , observed at both 3 and 49 days post - administration.	Chemical	Chemical
Error Type: Nested Entity (20%)		
Early postoperative delirium incidence risk factors were then assessed through three different multiple regression models.	Disease	Disease
The impact of immune - mediated heparin -induced thrombocytopenia type II (HIT type II) as a cause of thrombocytopenia.	Disease	Disease
Extensive literature search revealed multiple cases of coronary artery vasospasm secondary to zolmitriptan , but none of the cases were associated with TS.	Disease	Disease
Error Type: Others (4%)		
It is characterized by its intense urotoxic action , leading to hemorrhagic cystitis.	Disease	NotEntity
Famotidine is a histamine H2-receptor antagonist used in inpatient settings for prevention of stress ulcers and is showing increasing popularity because of its low cost .	Chemical	NotEntity
It is characterized by its intense urotoxic action , leading to hemorrhagic cystitis.	Disease	NotEntity

Table 3: Gold entities are underlined, predicted entities are in red. **Error type "similar semantic concepts"** means that our rules cannot distinguish two closely related semantic concepts. **Error type "inaccurate boundary"** means our rules label incorrectly about the boundaries of entities. **Error type "nested entity"** means the error is due to multiple possible entities are nested. **NotEntity** means the predicted span is not an entity.

Rule Selection Strategy	Precision	Recall	F_1
Rule Type	57.14	63.00	59.93
Entity&Rule Type	61.73	64.97	63.31
Entity Type	**66.53**	**66.94**	**66.73**

Table 4: Performance on the BC5CDR dataset with three different rule selection strategies.

Rule Type	BC5CDR	CHEMDNER	CoNLL
TokenStr	503 (41%)	1667 (44%)	779 (25%)
Pre ∧ Post	203 (17%)	629 (17%)	956 (31%)
Pre ∧ POS	288 (24%)	585 (16%)	455 (15%)
POS ∧ Post	149 (12%)	418 (11%)	438 (14%)
Dep ∧ POS	79 (6%)	432 (12%)	469 (15%)

Table 5: Number and ratio of different type rules.

the general domain task, PreNgram∧PostNgram is the most rule type learned by our model.

We also performed an error analysis on the BC5CDR dataset. Specifically, we sampled 100 entities predicted incorrectly by our learned rules and analyzed their error types. Analysis results show that 56% of errors are caused by an inability to distinguish closely related entity categories (chemicals vs medications), and another 20% are due to incorrect detection of entity boundaries. We also notice that some spans (e.g. "HIT type II") and their sub-spans (e.g. "HIT") are both disease entities (i.e., nested entities), but only the longer ones are annotated with gold labels. Our rules sometimes only predict the sub-spans as diseases, which contributes to 20% of the errors. We put examples of each error type in Table 3.

3.8 User Study of Explainable Logical Rules

Since our logical rules are intuitive clues for recognizing entities, we hypothesize that automatically learned rules can be used as understandable explanations for the predictions of entities. Therefore, we conducted a user study to find out how many logical rules are explainable. Specifically, we applied the learned rules in BC5CDR and sampled 100 entities labeled by at least one logical rule other

Labeled Entities and Sentences	Learned Logical Rules	Entity type
This occlusion occurred after EACA therapy in a patient with SAH and histopathological documentation of recurrent SAH.	PreNgram="a patient with" ∧ PostNgram="and"	Disease
We also analyzed published and unpublished follow-up data to determine the risk of ICH (PROPN) in antithrombotic users with MB.	PreNgram="the risk of" ∧ POStag=PROPN	Disease
3 weeks after initiation of amiodarone therapy for atrial fibrillation (ADJ NOUN).	PreNgram="therapy for" ∧ POStag=ADJ NOUN	Disease
Although 25 mg of lamivudine (NOUN) was slightly less effective than 100mg (P=.011) and 300 mg (P=.005).	PreNgram="mg of" ∧ POStag=NOUN	Chemical
These results suggest that the renal protective effects of misoprostol (NOUN) is dose - dependent.	PreNgram="protective effect of" ∧ POStag=NOUN	Chemical

Table 6: Examples of learned rules and correctly labeled entities (in red) by the learned rules in BC5CDR dataset.

than TokenString[10] for our user study. Some examples are shown in Table 6. We asked two annotators without domain knowledge and one biological expert to annotate whether our learned logical rules can be understood and used as explanations for why a span is predicted as a disease or chemical. Manual annotation results show that the two annotators and the biological expert agree that 81%, 87%, and 70% of the predicted entities can be explained by logical rules, respectively.

4 Related Work

Different types of methods have been proposed to build named entity tagging systems using indirect or limited supervision. Distant supervision (Mintz et al., 2009) is one kind of methods that have been proposed to alleviate human effort by training models using existing lexicons or knowledge bases. Recently, there have been attempts to build NER systems with distant supervision (Ren et al., 2015; Fries et al., 2017; Giannakopoulos et al., 2017). AutoNER (Shang et al., 2018b) trained a NER system by using both typed lexicons and untyped mined phrases as supervision. Peng et al. (2019) proposed an AdaPU algorithm to incorporate an incomplete dictionary as supervision. However, lexicons or knowledge bases are not always available for new domains and tasks, especially in specific domains and low-resource settings. Manually constructing these lexicons is often very expensive.

Bootstrapping is a technique to learn models from a small set of seeds, which has been proposed for word sense disambiguation (Yarowsky, 1995) and product attribute extraction (Putthividhya and Hu, 2011). Bootstrapping methods (Niu et al., 2003; Huang and Riloff, 2010) have been proposed for building entity tagging systems by assuming target entities are just proper names or noun phrases. Gupta and Manning (2014) used an improved pattern scoring method to bootstrap domain-specific terminologies with restricted part-of-speech patterns. However, previous works only focused on disambiguating entity types by assuming target entities are given or just syntactic chunks. But, as we shown earlier, target entities often do not align well with simple syntactic chunks. Bootstrapping methods that can automatically detect entity boundaries and predict their types simultaneously are desirable in real-world applications.

Recently, methods have been proposed to obtain weak labels by manually writing labeling functions (Bach et al., 2017). Based on this idea, several methods (Safranchik et al., 2020; Lison et al., 2020a) have been proposed for NER by assuming the availability of a sufficient amount of handcrafted labeling functions and lexicons. However, manually designing labeling rules is challenging, which requires a significant amount of manual effort and domain expertise. Our work aims to learn logical rules automatically to reduce human effort.

5 Conclusion

In this work, we explored how to build a tagger from a small set of seed logical rules and unlabeled data. We defined five types of simple logical rules and introduced compound logical rules that are composed from simple rules to detect entity boundaries and classify their types simultaneously. We also design a dynamic label selection method to select accurate pseudo labels generated from learned rules for training a discriminative tagging model. Experimental results demonstrate that our method is effective and outperforms existing weakly supervised methods.

[10]We exclude TokenString rules because they are self-explainable.

References

Stephen H. Bach, B. He, A. Ratner, and C. Ré. 2017. Learning the structure of generative models without labeled data. *Proceedings of machine learning research*, 70:273–82.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. Scibert: A pretrained language model for scientific text. In *EMNLP/IJCNLP*.

J. Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Jason Alan Fries, Sen Wu, A. Ratner, and Christopher Ré. 2017. Swellshark: A generative model for biomedical named entity recognition without labeled data. *ArXiv*, abs/1704.06360.

Athanasios Giannakopoulos, C. Musat, Andreea Hossmann, and Michael Baeriswyl. 2017. Unsupervised aspect term extraction with b-lstm crf using automatically labelled datasets. In *WASSA@EMNLP*.

S. Gupta and Christopher D. Manning. 2014. Improved pattern learning for bootstrapped entity extraction. In *CoNLL*.

Ruihong Huang and E. Riloff. 2010. Inducing domain-specific semantic class taggers from (almost) nothing. In *ACL*.

Zhengbao Jiang, W. Xu, J. Araki, and Graham Neubig. 2020. Generalizing natural language analysis through span-relation representations. In *ACL*.

Martin Krallinger, O. Rabal, Florian Leitner, Miguel Vazquez, David Salgado, Zhiyong Lu, Robert Leaman, Yanan Lu, Dong-Hong Ji, D. M. Lowe, R. Sayle, R. Batista-Navarro, R. Rak, Torsten Huber, Tim Rocktäschel, Sérgio Matos, D. Campos, Buzhou Tang, H. Xu, Tsendsuren Munkhdalai, K. Ryu, S. V. Ramanan, P. S. Nathan, S. Zitnik, M. Bajec, L. Weber, Matthias Irmer, S. Akhondi, J. Kors, S. Xu, X. An, Utpal Kumar Sikdar, A. Ekbal, M. Yoshioka, Thaer M. Dieb, Miji Choi, Karin M. Verspoor, Madian Khabsa, C. Lee Giles, H. Liu, K. Ravikumar, Andre Lamurias, F. Couto, Hong-Jie Dai, R. Tsai, C. Ata, T. Can, Anabel Usie, Rui Alves, Isabel Segura-Bedmar, Paloma Martínez, J. Oyarzábal, and A. Valencia. 2015. The chemdner corpus of chemicals and drugs and its annotation principles. *Journal of Cheminformatics*, 7:S2 – S2.

Kenton Lee, Luheng He, M. Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *EMNLP*.

J. Li, Yueping Sun, Robin J. Johnson, Daniela Sciaky, Chih-Hsuan Wei, Robert Leaman, A. P. Davis, C. Mattingly, Thomas C. Wiegers, and Zhiyong Lu. 2016. Biocreative v cdr task corpus: a resource for chemical disease relation extraction. *Database: The Journal of Biological Databases and Curation*, 2016.

P. Lison, A. Hubin, Jeremy Barnes, and Samia Touileb. 2020a. Named entity recognition without labelled data: A weak supervision approach. *ArXiv*, abs/2004.14723.

Pierre Lison, Jeremy Barnes, Aliaksandr Hubin, and Samia Touileb. 2020b. Named entity recognition without labelled data: A weak supervision approach. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1518–1533, Online. Association for Computational Linguistics.

M. Mintz, Steven Bills, R. Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL/IJCNLP*.

Cheng Niu, Wei Li, Jihong Ding, and Rohini K Srihari. 2003. A bootstrapping approach to named entity classification using successive learners. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 335–342.

Minlong Peng, Xiaoyu Xing, Qi Zhang, Jinlan Fu, and X. Huang. 2019. Distantly supervised named entity recognition using positive-unlabeled learning. *ArXiv*, abs/1906.01378.

Jeffrey Pennington, R. Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Duangmanee Putthividhya and Junling Hu. 2011. Bootstrapped named entity recognition for product attribute extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1557–1567, Edinburgh, Scotland, UK. Association for Computational Linguistics.

A. Ratner, Stephen H. Bach, Henry R. Ehrenberg, Jason Alan Fries, Sen Wu, and C. Ré. 2017. Snorkel: Rapid training data creation with weak supervision. *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, 11 3:269–282.

Xiang Ren, Ahmed El-Kishky, C. Wang, Fangbo Tao, Clare R. Voss, and Jiawei Han. 2015. Clustype: Effective entity recognition and typing by relation phrase-based clustering. *KDD : proceedings. International Conference on Knowledge Discovery Data Mining*, 2015:995–1004.

Esteban Safranchik, Shiying Luo, and Stephen H. Bach. 2020. Weakly supervised sequence tagging from noisy rules. In *AAAI*.

E. T. K. Sang and F. D. Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *ArXiv*, cs.CL/0306050.

Jingbo Shang, Jialu Liu, Meng Jiang, X. Ren, Clare R. Voss, and Jiawei Han. 2018a. Automated phrase mining from massive text corpora. *IEEE Transactions on Knowledge and Data Engineering*, 30:1825–1837.

Jingbo Shang, Liyuan Liu, X. Ren, X. Gu, Teng Ren, and Jiawei Han. 2018b. Learning named entity tagger using domain-specific dictionary. *ArXiv*, abs/1809.03599.

M. Thelen and E. Riloff. 2002. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *EMNLP*.

Lingyong Yan, Xianpei Han, L. Sun, and B. He. 2019. Learning to bootstrap for entity set expansion. In *EMNLP/IJCNLP*.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *ACL*.

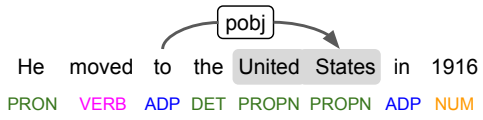
Yunyi Zhang, Jiaming Shen, Jingbo Shang, and Jiawei Han. 2020. Empower entity set expansion via language model probing. In *ACL*.

A Appendices

A.1 Details of Logical Rule Extraction

In this section, we present details of the extraction and the matching logic of our designed logical rules, using the following sentence with a location entity **United States** as an example.

Example 1.



We first obtain a parsed dependency tree of the sentence using the spaCy pipeline (`en_core_web_sm` model). Then our framework will generate all candidate rules for each candidate entity. Here, we use the token span `United States` as the target candidate entity to show how these rules are extracted.

TokenString. We use the lower-case and lemmatized tokens of an entity candidate as a TokenString rule. Given the above example, we will extract a `TokenString`="united state" rule.

PreNgram. It matches preceding N tokens. All tokens in rules will be lower cased and lemmatized. In our experiments, we set N to 3. In Example 1, we extract `PreNgram`="the", `PreNgram`="to the", and `PreNgram`="move to the" as candidate rules.

PostNgram. It matches the succeeding N tokens, which are also lower cased and lemmatized. N is set to 3 in our experiments. In Example 1, we can extract `PostNgram`="in", `PostNgram`="in 1916", and `PostNgram`="in 1916 ." as candidate rules.

POSTag. We extract the part-of-speech tags of tokens in a span text using the spaCy pipeline. In Example 1, we can extract `POSTag`="PROPN PROPN" as a candidate rule.

DependencyRel. We first find the head word[11] in the text span. Then, we extract the governor (i.e. head) of the head word as a dependency rule with depth 1. In Example 1, `state` is the head word of text span `United States`. `to` is the governor of head word `state`, so `DependencyRel`="to" is the DependencyRel rule with depth 1. Next, all tokens dependent on the head word are considered as DependencyRel rules with depth 2. In Example 1, word `move` is logical rule with depth 2. We use $\|$ to connect token with depth 1 and token with depth 2. Finally, in Example 1, we have logical rule `DependencyRel`="to" and `DependencyRel`="move$\|$to".

The numbers of rule candidates for each dataset are: BC5CDR (108,756), CHEMDNER (441, 595), CONLL2003 (142, 976).

[11]For simplicity, we just used the last token as the head word of a token span.

A.2 Details of Neural Tagger

In this section, we present details of span representation and prediction in our neural tagger.

Span Representation. Given a sentence $\mathbf{x} = [w_1, w_2, \ldots, w_n]$ of n tokens, a span $s_i = [w_{b_i}, w_{b_{i+1}}, \ldots, w_{e_i}]$, where b_i, e_i are the start and end indices respectively. The representation of spans contains two components: a content representation $\mathbf{z}_i^c$ calculated as the weighted average across all token embeddings in the span, and a *boundary representation* $\mathbf{z}_i^u$ that concatenates the embeddings at the start and end positions of the span. Specifically,

$$\begin{aligned}
\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_n &= \text{TokenRepr}(w_1, w_2, \ldots, w_n), \\
\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_n &= \text{BiLSTM}(\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_n), \\
\mathbf{z}_i^c &= \text{SelfAttn}(\mathbf{c}_{b_i}, \mathbf{c}_{b_i+1}, \ldots, \mathbf{c}_{e_i}), \\
\mathbf{z}_i^u &= [\mathbf{u}_{b_i}; \mathbf{u}_{e_i}], \mathbf{z}_i = [\mathbf{z}_i^c; \mathbf{z}_i^u],
\end{aligned}$$

where TokenRepr could be non-contextualized, such as Glove (Pennington et al., 2014), or contextualized, such as BERT (Devlin et al., 2019). BiLSTM is a bi-directional LSTM layer and SelfAttn is a self-attention layer. For further details please refer to Lee et al. (2017).

Span Prediction. We predict labels for *all* spans up to a fixed length of l words using a multilayer perceptron (MLP):

$$\mathbf{o}_i = \text{softmax}(\mathbf{MLP}^{\text{span}}(\mathbf{z}_i)) \quad (4)$$

where $\mathbf{o}_i$ is prediction for the span. We introduce one negative label `NEG` as an additional label which indicates invalid spans (i.e., spans that are not named entities in the corpus).

A.3 Negative Instances for Training

To provide negative supervision for neural network training, we pre-process unlabeled data and collect all noun phrases. Token spans outside noun phrases are used as initial negative supervision. Compared with previous works (Ratner et al., 2017; Fries et al., 2017) that directly use noun phrases as entity candidates, in our work, noun phrases only provide negative supervision. In the following iterations, these negative instances still have a chance to be recognized correctly.

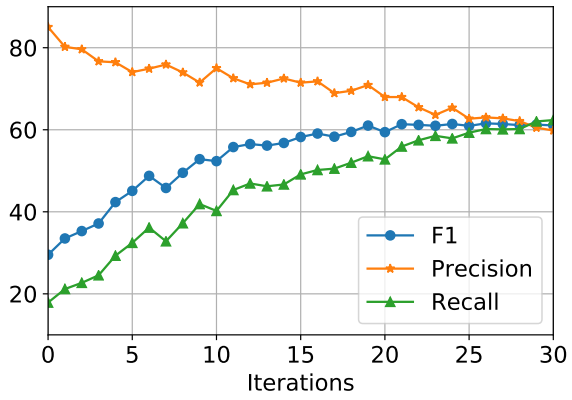


Figure 4: Iterations vs. performance of the neural NER tagger on CHEMDNER datasets.

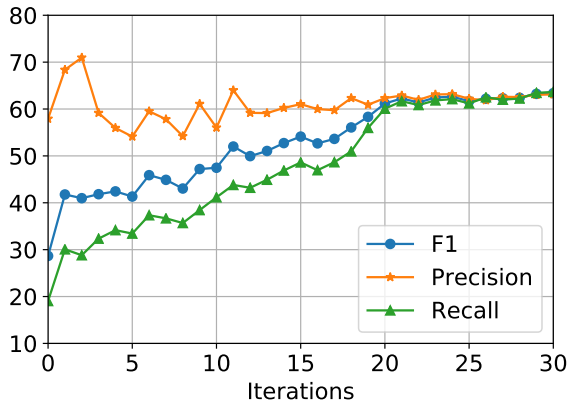


Figure 5: Iterations vs. performance of the neural NER tagger on CoNLL2003 datasets.

A.4 Parameters

In our neural NER tagger, we use the Adam optimizer with learning rate $2e^{-5}$, a dropout ratio 0.5, and a batch size of 32 for all experiments. For better stability, we use gradient clipping of 5.0. In addition, the maximum length of spans is 5, and precision thresholds for rules are 0.9 for all experiments.

In the dynamic label selection step, we set the temperature of thresholds to 0.8, sample times $N = 50$, $E_s = 3$, and the temperature $\tau = 0.8$ to control threshold. In logical rule scoring and selection step, we set $\eta = 1$, and threshold $\theta = 0.9$.

In our experiments, we use SciBert for two biomedical datasets and Bert for CoNLL2003 dataset. During training, we run the framework for 32 iterations for all datasets and select the best model based on development sets.

Condition	Label
TokenString(x)==“nicotine”	Chemical
TokenString(x)==“morphine”	
TokenString(x)==“haloperidol”	
TokenString(x)==“warfarin”	
TokenString(x)==“clonidine”	
TokenString(x)==“creatinine”	
TokenString(x)==“isoproterenol”	
TokenString(x)==“cyclophosphamide”	
TokenString(x)==“sirolimus”	
TokenString(x)==“tacrolimus”	
TokenString(x)==“proteinuria”	Disease
TokenString(x)==“esrd”	
TokenString(x)==“thrombosis”	
TokenString(x)==“tremor”	
TokenString(x)==“hepatotoxicity”	
TokenString(x)==“hypertensive”	
TokenString(x)==“thrombotic”	
TokenString(x)==“microangiopathy”	
TokenString(x)==“thrombocytopenia”	
TokenString(x)==“akathisia”	

Table 7: Seed logical rules for BC5CDR dataset.

A.5 Implementation

We implement our framework with Pytorch 1.4.0[12] and our rule labeling is based on Snorkel 0.9.5[13]. We train our framework on NVIDIA Quadro RTX 8000 GPU. Our neural NER module has 114,537,220 parameters. It takes about 30 minutes to complete a whole iteration.

A.6 Dictionary for AutoNER

In Table 2, we used the same manual seed rules as supervision for all experiments. For AutoNER, all phrases generated from AutoPhrase are used as untyped phrases (i.e., full dictionary in AutoNER), the sizes are: BC5CDR (6,619), CHEMDNER (15,995), CONLL2003 (4,137). We expanded seeds with CGExpan and used the expansion as the typed terms for AutoNER (i.e., the core dictionary in AutoNER). We experimented with different sizes of dictionaries and reported the best results. The sizes for the best performance are: BC5CDR (800), CHEMDNER (500), CONLL2003 (1000). We found that the performance will be lower when we try to use larger automatically expanded dictionaries.

A.7 Seed Logical Rules

In this section, we show the seeds used in experiments of Table 2.

Seed logical rules for BC5DCR, CoNLL2003 and CHEMDNER is shown in Table 7, 8 and 9

[12]https://pytorch.org/

[13]https://www.snorkel.org/

Condition	Label
TokenString(x)==“britain”	Location
TokenString(x)==“italy”	
TokenString(x)==“russia”	
TokenString(x)==“sweden“	
TokenString(x)==“belgium”	
TokenString(x)==“iraq”	
TokenString(x)==“south africa”	
TokenString(x)==“united states”	
TokenString(x)==“wasim akram”	Person
TokenString(x)==“waqar younis”	
TokenString(x)==“mushtaq ahmed”	
TokenString(x)==“mother teresa”	
TokenString(x)==“aamir sohail”	
TokenString(x)==“bill clinton”	
TokenString(x)==“saeed anwar”	
TokenString(x)==“osce”	Organization
TokenString(x)==“nato”	
TokenString(x)==“honda”	
TokenString(x)==“interfax”	
TokenString(x)==“marseille”	

Table 8: Seed logical rules for CoNLL2003 dataset.

respectively.

A.8 Iterations vs. Performance

Figure 4 and Figure 5 show the performance vs. iterations on CHEMDNER and CoNLL 2003 dataset.

Condition	Label
TokenString(x)==“glucose”	Chemical
TokenString(x)==“oxygen”	
TokenString(x)==“cholesterol”	
TokenString(x)==“glutathione”	
TokenString(x)==“ethanol”	
TokenString(x)==“ca (2 +)”	
TokenString(x)==“calcium”	
TokenString(x)==“androgen”	
TokenString(x)==“copper”	
TokenString(x)==“graphene”	
TokenString(x)==“glutamate”	
TokenString(x)==“dopamine”	
TokenString(x)==“cocaine”	
TokenString(x)==“cadmium”	
TokenString(x)==“serotonin”	
TokenString(x)==“estrogen”	
TokenString(x)==“nicotine”	
TokenString(x)==“tyrosine”	
TokenString(x)==“resveratrol”	
TokenString(x)==“nitric oxide”	
TokenString(x)==“cisplatin”	
TokenString(x)==“alcohol”	
TokenString(x)==“superoxide”	
TokenString(x)==“curcumin”	
TokenString(x)==“(1) h”	
TokenString(x)==“metformin”	
TokenString(x)==“amino acid”	
TokenString(x)==“arsenic”	
TokenString(x)==“zinc”	
TokenString(x)==“testosterone”	
TokenString(x)==“flavonoids”	
TokenString(x)==“camp”	
TokenString(x)==“methanol”	
TokenString(x)==“amino acids”	
TokenString(x)==“mercury”	
TokenString(x)==“fatty acids”	
TokenString(x)==“polyphenols”	
TokenString(x)==“nmda”	
TokenString(x)==“silica”	
TokenString(x)==“5 - ht”	

Table 9: Seed logical rules for CHEMDNER dataset.