AutoTriggER: Label-Efficient and Robust Named Entity Recognition with Auxiliary Trigger Extraction

Anonymous EMNLP submission

Abstract

Deep neural models for named entity recognition (NER) have shown impressive results in overcoming label scarcity and generalizing to unseen entities by leveraging distant supervision and auxiliary information such as explanations. However, the costs of acquiring such additional information are generally prohibitive. In this paper, we present a novel two-stage framework (AUTOTRIGGER) to improve NER performance by automatically generating and leveraging "*entity triggers*" which are human-readable cues in the text that help guide the model to make better decisions. Our framework leverages post-hoc explanation to generate rationales and strengthens a model's prior knowledge using an embedding interpolation technique. This approach allows models to exploit triggers to infer entity boundaries and types instead of solely memorizing the entity words themselves. Through experiments on three well-studied NER datasets, AUTOTRIGGER shows strong label-efficiency, is capable of generalizing to unseen entities, and outperforms the RoBERTa-CRF baseline by nearly 0.5 F1 points on average. [1]

1 Introduction

Named Entity Recognition (NER) serves as a key building block in information extraction systems. Recent advances in deep neural models for NER have yielded state-of-the-art performance when sufficient human annotations are available (Lample et al., 2016; Liu et al., 2018; Peters et al., 2017; Ma and Hovy, 2016). However, such success cannot easily transfer to practitioners developing NER systems in specific domains (*e.g.*, biomedical papers, financial reports, legal documents), where domain-expert annotations are expensive and slow to obtain. Moreover, NER systems face challenges in real-world applications where novel entities, unseen in training data, are often encountered (Lin et al., 2021). Recent attempts addressing label scarcity and improving generalization to unseen entities have explored various types of human-curated resources as auxiliary supervision, such as entity dictionaries (Peng et al., 2019; Shang et al., 2018; Yang et al., 2018; Liu et al., 2019a), labeling rules (Safranchik et al., 2020; Jiang et al., 2020), prompts (Ding et al., 2021; Cui et al., 2021), demonstrations (Lee et al., 2022), retrieved context (Wang et al., 2021), and labeling explanations (Hancock et al., 2018; Wang et al., 2020; Ye et al., 2020; Lin et al., 2020; Lee et al., 2020a). In particular, human-provided labeling explanations as auxiliary supervision signals are cost-effective compared to collecting label-only annotations for larger number of instances. For the NER task, Lin et al. (2020) introduced the concept of an *entity trigger*, an effective way to represent explanations for the labeling decisions (See Figure 1 (a) vs. (b)).

However, prior works have the following limita-

[1]Code and data have been uploaded and will be published:

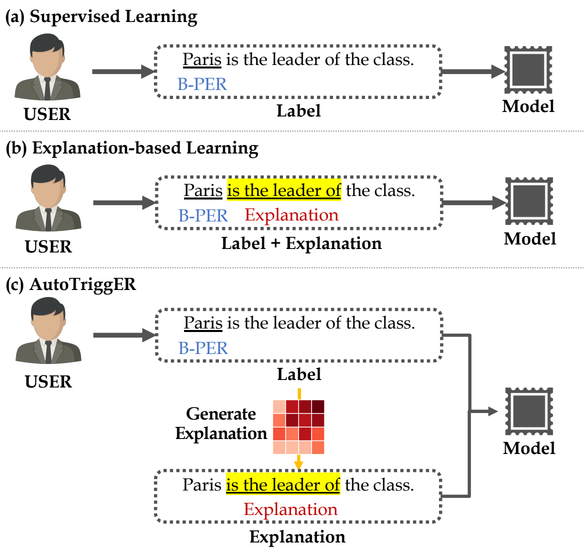


Figure 1: Existing explanation-based learning frameworks mostly rely on humans provided labeling explanations while our framework automatically generates and leverages explanations to NER.

tions: (1) *Expense of collecting rationales*: Prior works primarily use a limited number of *crowd-soured* explanations (e.g., *entity trigger*) for improving data (label) efficiency of model training. While such human-curated auxiliary supervision is of high quality, the crowd-sourcing procedure can be very expensive and time-consuming. This largely limits the scale and domains of the collected rationales; (2) *Unseen entity generalization*: The biggest advantage of leveraging such explanations for NER is a generalization ability towards unseen entities. Reinforcing this prior knowledge in the model rather than memorizing the entity words themselves can make the model robust against unseen entities during training. However, prior works do not evaluate such generalization ability.

To address these limitations, we propose a two-stage NER framework, named AUTOTRIGGER , which improves label efficiency and model robustness without human efforts (see Figure 1 (c)). First, it *automatically* generates entity triggers as explainable features to reduce human effort on collecting rationales. It then strengthens the prior knowledge in the model by interpolating model embeddings of entity-masked sentence which force the model to rely more on triggers and trigger-masked sentence which force the model to rely more on the entity.

The *first* stage of our framework (Sec. 3.1) aims to *automatically extract entity triggers* using a post-hoc explanation. We propose to exploit the syntactic features of a sentence for assigning importance scores to a group of input tokens such that we can extract entity triggers as auxiliary supervision. Specifically, we form a collection of trigger candidates for an entity in a sentence using the phrases extracted from its *constituency parsing tree* (Joshi et al., 2018). Then we score each trigger candidate based on its ability to predict the target entity, while varying the surrounding context to ensure trigger robustness.

The *second* stage (Sec. 3.2) focuses on how to use triggers, which are useful contextual clues, in making the prediction. We propose *Trigger Interpolation Network* (TIN), a novel architecture that effectively uses trigger-labeled NER data to train a model. Here, we employ two separate masking passes when learning our model's embeddings: one masking the entity words (forcing the model to rely more on the triggers) and one masking the triggers (forcing the model to rely more on the entity words). We then interpolate the embeddings of both entity-masked and trigger-masked sentences to learn a mixed sentence representation, which is provided as input to a standard sequence labeling task. In this manner, the TIN can effectively learn to focus on useful contextual clues to infer entity boundaries and types with contextualized embeddings from pre-trained language models such as BERT (Devlin et al., 2019) while it may not miss the entities whose type can be determined by itself. (e.g., inputs that contain only the entities).

Extensive experimental results on several domains show that AUTOTRIGGER framework consistently outperforms baseline methods by 0.5 F1 points on average in fully supervised setting. Our work shows strong performance especially in label scarcity setting and unseen entity generalization. In the label scarcity setting ranging from 50 to 200 number of train instances, assuming a task that needs annotation from scratch, our model gains more than 3-4 F1 score on average. Also, for the filtered test set where the entities did not appear in the train and development sets, assuming a real-world applications in which entities can be out of the distribution of the train data, our model gains more than 2-3 F1 score on average.

2 Problem Formulation

Named Entity Recognition. We let $\mathbf{x} = [x^{(1)}, x^{(2)}, \dots x^{(n)}]$ denote the sentence consisting of a sequence of n words and $\mathbf{y} = [y^{(1)}, y^{(2)}, \dots y^{(n)}]$ denote the NER-tag sequence. The task is to predict the entity tag $y^{(i)} \in \mathcal{Y}$ for each word $x^{(i)}$, where $\mathcal{Y}$ is a pre-defined set of tags such as $\{$B-PER, I-PER, …, O$\}$. We let $\mathcal{D}_L$ denote the labeled dataset consisting of the set of instances $\{(\mathbf{x_i}, \mathbf{y_i})\}$, where $\mathbf{x_i}$ is the i-th input sentence and $\mathbf{y_i}$ is its output tag sequence. Here, we use BIOES scheme (Chiu and Nichols, 2016).

Entity Trigger Extending from extractive rationales for text classification (DeYoung et al., 2020; Jain et al., 2020), an *entity trigger* is a form of extractive explanatory annotation for NER, defined as *a group of words that help explain the recognition process of an entity mentioned in the sentence* (Lin et al., 2020). For example, in Figure 2, "*had ... dinner at*" and "*where the food*" are two distinct triggers associated with the RESTAURANT entity "*Sunnongdan*". Formally, given a labeled NER instance $(\mathbf{x}, \mathbf{y})$, we use T to denote the set of entity triggers for this instance. Each trigger $t_i \in T$ is

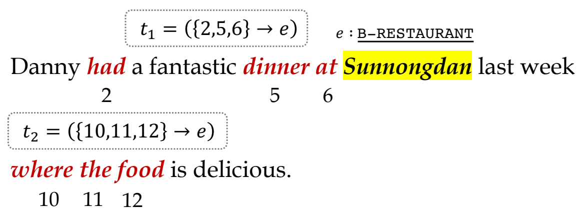

$t_1 = (\{2,5,6\} \rightarrow e)$
e : B-RESTAURANT
Danny had a fantastic dinner at Sunnongdan last week
2
5
6
$t_2 = (\{10,11,12\} \rightarrow e)$
where the food is delicious.
10
11
12

Figure 2: **Example of entity trigger.** Entity trigger t_i is a cue phrase toward the entity e in the sentence, which is represented by a set of corresponding word indices. Both entity triggers (t_1,t_2) are associated to the same entity e ("*Sunnongdan*") typed as restaurant.

associated with an entity e and a set of word indices $\{w_i\}$. That is, $t = (\{w_1, w_2, \dots\} \rightarrow e)$ represents an entity trigger. For example, in Figure 2, $t_1 = \{2, 5, 6\}$ is an entity trigger identified for entity "*Sunnongdan*" in the sentence. A *trigger-labeled NER dataset*, $\mathcal{D}_T = \{(\mathbf{x_i}, \mathbf{y_i}, T(\mathbf{x_i}, \mathbf{y_i}))\}$, consists of examples in a labeled NER dataset $\mathcal{D}_L$ with their associated entity triggers.

Problem Definition We focus on the problem of how to *automatically* extract entity triggers to create a trigger-labeled dataset $\mathcal{D}_T$ from $\mathcal{D}_L$ without manual effort and to cost-effectively train a NER model using $\mathcal{D}_T$. Here, we aim to achieve better performance and robustness over existing NER models using the same amount of training instances. Moreover, to check the quality of the automatically extracted triggers, we compare with the model using human-labeled triggers.

3 Approach

AutoTriggER is a two-stage architecture that begins with an *automatic trigger extraction* stage followed by a *trigger interpolation network* (`TIN`). It automatically extracts and scores the importance of entity trigger phrases in the first stage (Sec. 3.1) and uses them in the later stage to train the NER model (Sec. 3.2).

3.1 Automatic Trigger Extraction

Automatic trigger extraction is the first stage of our AutoTriggER framework. Different from prior works on extracting rationales and explanations for sentence classification (Ribeiro et al., 2016; Li et al., 2016b), we look to extract triggers for explaining the detection (occurrence) of an entity in the sentence. In our study, we extend the *sampling and occlusion* (`SOC`) algorithm (Jin et al., 2020), a feature attribution algorithm for post-hoc model explanation, to extract entity triggers. Here, we

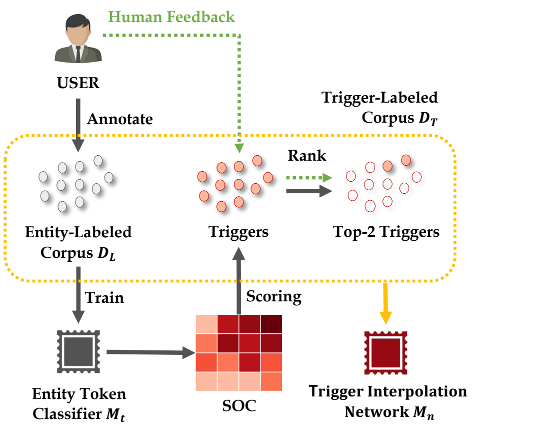

Human Feedback
USER
Annotate
Trigger-Labeled Corpus D_T
Rank
Entity-Labeled Corpus D_L
Triggers
Top-2 Triggers
Train
Scoring
Entity Token Classifier M_t
SOC
Trigger Interpolation Network M_n

Figure 3: **Overview of the proposed AutoTrigger.** It trains an entity-token classifier $\mathcal{M}_t$ with entity-labeled corpus $\mathcal{D}_L$ and uses the sampling-and-occlusion (SOC) algorithm to extract triggers. There is a provision for leveraging human feedback in the framework for refining automatically generated triggers. Trigger Interpolation Network (TIN) learns the NER model from the trigger-labeled corpus.

exploit a token classifier to model the score function of the target entity in the sentence and limit the search space to a set of phrases from the constituent parse tree. Specifically, given a labeled NER corpus $(\mathbf{x_i}, \mathbf{y_i}) \in \mathcal{D}_L$, we consider four main steps for entity trigger extraction: 1) generating a set of candidate phrases $\mathcal{P}$, 2) constructing entity token classifier $\mathcal{M}_t$, 3) trigger phrase scoring, and 4) trigger phase selection.

Candidate Generation Given a labeled sentence $(\mathbf{x_i}, \mathbf{y_i})$, we obtain its constituency parse tree and consider the set of phrase nodes $\mathcal{P}$ from the tree as trigger candidates (see Figure. 4 for an illustration with examples). `SOC` aims to compute context-independent phrase-level importance for sequence classification tasks such as sentiment analysis and relation extraction (Jin et al., 2020). It uses agglomerative clustering (Singh et al., 2019) to effectively compute the phrase-level importance without evaluating all the possible phrases in the sentence. Here, we provide pre-defined hierarchy for limiting the search space to a set of phrases from the constituent parse tree. Specifically, given a sentence instance $\mathbf{x_i} \in \mathcal{D}_L$ and a target entity mentioned in this sentence $e \in \mathbf{x_i}$, we generate a set of phrase candidate $\mathcal{P} = \{\mathbf{p_i}\}$, where $\mathbf{p_i} = (w_s, w_e)$. Here (w_s, w_e) denote the start and end index of the phrase span $\mathbf{p_i}$. To generate $\mathcal{P}$, we parse the input sentence $\mathbf{x_i}$ using an off-the-shelf constituency parser [2] and

[2]https://stanfordnlp.github.io/CoreNLP/parse.html

collect phrase nodes in the parse tree as the set of candidate phrases $\mathcal{P}$. To avoid including target entity as an entity trigger, we discard phrases that contain the target entity, i.e., $\{\mathbf{p_j} | e \in \mathbf{p_j}\}$.

Entity Token Classifier Construction In order to apply SOC to the sequence labeling task, we propose a way to derive the importance score of the phrase $\mathbf{p}$ specific to the input sentence x and target entity e. To achieve this, we use a token classification model $\mathcal{M}_t$ to generate prediction scores in the label space (e.g., probability for target class of a token). Given an input sentence $\mathbf{x_i} = [x_i^{(1)}, x_i^{(2)}, \dots x_i^{(n)}]$, $\mathcal{M}_t$ classifies each token $x_i^{(j)}$ to the named entity tag $y_i^{(j)} \in \mathcal{Y}$ where $\mathcal{Y}$ is a predefined set of named entity tags such as B-PER, I-PER and O. We learn $\mathcal{M}_t$ with labeled corpus $\mathcal{D}_L$ to fit a probability distribution $\mathbb{P}(\mathbf{y}|\mathbf{x})$. Then, we can derive the prediction score function s towards the target entity e which is computed as the average conditional probability over tokens of the target entity $x^{(j)} \in e$ as follows:

$$s(\mathbf{x}, e) = \frac{1}{|e|} \sum_{x^{(j)} \in e} \mathbb{P}(\mathbf{y^{(j)}} | x^{(j)}). \tag{1}$$

Phrase Scoring. Once we have the set of candidate phrases $\mathcal{P}$ for the given input instance (x_i, y_i), we aim to measure the score of each of the candidate phrase $\mathbf{p}$ w.r.t the target entity e. We use the score function s of the $\mathcal{M}_t$ to measure the importance score of each phrase $\mathbf{p}$. Here, we have two steps: (1) *input occlusion*, (2) *context sampling*.

Input occlusion (Li et al., 2016b) computes the importance of $\mathbf{p}$ specific to the entity e in the input $\mathbf{x}$ by measuring the prediction difference caused by replacing the phrase $\mathbf{p}$ with padding tokens $0_\mathbf{p}$. We use $s\left(\mathbf{x_{-p}}, e; \mathbf{0_p}\right)$ to denote the model prediction score after replacing the masked-out context $\mathbf{x_{-p}}$ with padding tokens $0_\mathbf{p}$:

$$\phi(\mathbf{p}, \mathbf{x}, e) = s(\mathbf{x}, e) - s\left(\mathbf{x_{-p}}, e; \mathbf{0_p}\right). \tag{2}$$

However, the importance score $\phi(\mathbf{p}, \mathbf{x}, e)$ from equation 2 has a challenge that it is difficult to model the direct impact of $\mathbf{p}$ towards e since the score is dependent on the context words around $\mathbf{p}$. For example, in Figure. 4, $\mathbf{p}$ ("*the next mayor*") is replaced by pad tokens to compute its importance towards the entity e ("*Cary Moon*"). However, the importance score of $\mathbf{p}$ is dependent on context words around $\mathbf{p}$ ("*won't be ... of Seattle*"). To compute the context independent importance score,

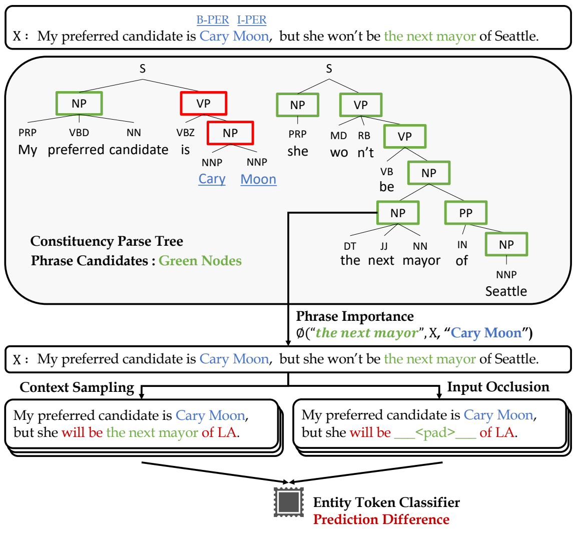

B-PER I-PER
X : My preferred candidate is Cary Moon, but she won't be the next mayor of Seattle.
S
NP
VP
PRP VBD NN VBZ NP
My preferred candidate is
NNP NNP
Cary Moon
NP VP
PRP she
MD RB VP
wo n't
VB be NP
NP PP
DT JJ NN IN NP
the next mayor of
NNP Seattle
Constituency Parse Tree
Phrase Candidates : Green Nodes
Phrase Importance
ϕ("the next mayor", X, "Cary Moon")
X : My preferred candidate is Cary Moon, but she won't be the next mayor of Seattle.
Context Sampling
Input Occlusion
My preferred candidate is Cary Moon, but she will be the next mayor of LA.
My preferred candidate is Cary Moon, but she will be ___<pad>___ of LA.
Entity Token Classifier
Prediction Difference

Figure 4: **Overview of the Sampling and Occlusion (SOC).** It creates a set of phrase candidates with phrase nodes of the constituency parse tree, and then computes the phrase importance by average prediction difference between context sampled sentences and its phrase-masked sentences. Note that the entity mention "*Cary Moon*" is not included as candidate.

SOC proposes *context sampling* that samples the context words around the phrase p and computes the average prediction differences over the samples. Specifically, it samples the context words $\hat{x}_\delta$ from a language model $p(\hat{x}_\delta | x_{-\delta})$ that is trained on $\mathcal{D}_L$, and obtains a set of context word replacements $\mathcal{S}$. Here, we use 20 replacements. For each replacement $\hat{x}_\delta \in \mathcal{S}$, we measure the prediction difference caused by replacing the phrase $\mathbf{p}$ with padding tokens. We take the average of these prediction differences to be the context-independent score $\phi(\mathbf{p}, \mathbf{x}, e)$ of the phrase $\mathbf{p}$, as expressed in equation 3:

$$\frac{1}{|\mathcal{S}|} \sum_{\hat{\mathbf{x}}_\delta \in \mathcal{S}} \left[s\left(\mathbf{x}_{-\delta}, e; \hat{\mathbf{x}}_\delta\right) - s\left(\mathbf{x}_{-\{\delta, \mathbf{p}\}}, e; \hat{\mathbf{x}}_\delta; \mathbf{0_p}\right) \right] \tag{3}$$

In Figure. 4, context words "*won't be*" and "*of Seattle*" around the phrase "*the next mayor*" are replaced into "*will be*" and "*of LA*" which are sampled from the language model. Then, the classifier computes the prediction difference between the sampled sentences with and without the phrase.

Phrase Selection The importance score $\phi(\mathbf{p}, \mathbf{x}, e)$ for phrase $\mathbf{p}$ is the degree to which $\mathbf{p}$ determines the correct entity type of target entity e. After obtaining $\phi(\mathbf{p}, \mathbf{x}, e)$ for all phrase candidates $\mathcal{P} = \{\mathbf{p_i}\}$, we pick the top k candidate phrases with the highest importance score as the entity

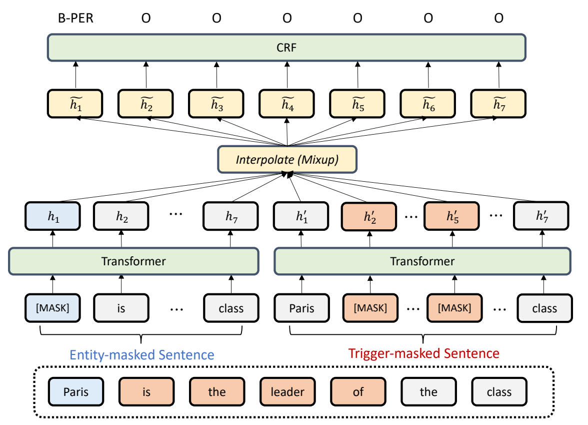


Figure 5: **Overview of the Trigger Interpolation Network (TIN).** Given an input sentence we create an Entity-masked sentence and a Trigger-masked Sentence. Then we interpolate token level representations h_i and h'_i to create new hidden state representation, $\tilde{h}_i$. Interpolated hidden representations are fed to a CRF.

triggers, where k is a hyperparameter. Specifically, for each input instance $(\mathbf{x_i}, \mathbf{y_i}) \in \mathcal{D}_L$, we pick the top k candidate phrases as entity triggers $T(\mathbf{x_i}, \mathbf{y_i})$ per each entity $e \in \mathbf{x_i}$ to create a form $(\mathbf{x_i}, \mathbf{y_i}, T(\mathbf{x_i}, \mathbf{y_i})) \in \mathcal{D}_T$. Here, we search for the best k out of [1, 2, 3, 5, 7, 10] (See Figure 12).

3.2 Trigger Interpolation Network (TIN)

The second stage of AUTOTRIGGER is the trigger interpolation network (TIN), which we define as a neural network that learns from a trigger-labeled dataset $\mathcal{D}_T$ consisting of a set of instances of the form $\{(\mathbf{x}, \mathbf{y}, T(\mathbf{x}, \mathbf{y}))\}$. The main idea behind the model is to combine two representations obtained from the sentence with target entity tokens and automatically extracted entity triggers. In one representation, the entity tokens are masked and we hypothesize that it forces the model to infer the entity boundaries and types by leveraging the contextual clues. However, some of the entity tokens itself may already have enough information to infer the entity type without any contextual clues. Thus, in another representation, the entity tokens become the predominant source of information for classifying the entity as we mask out the triggers.

TIN encodes the input sequence $\mathbf{x}$ with a transformer encoder $\mathbf{F}(.;\theta)$ and feeds the hidden representations $\mathbf{h}$ to a CRF tagger. Our proposal is to create two different representations of a token in a sequence and interpolate (mixup) them. By doing this, we expect the model to predict less confidently on interpolations of hidden representations, which eventually make model be robust to both entity-perturbed and trigger-perturbed sentences. As shown in Figure 5, we first create entity-masked sentence $\mathbf{x}_{-e}$ and trigger-masked sentence $\mathbf{x}_{-t}$ for a given input instance $\{(\mathbf{x}, \mathbf{y}, T(\mathbf{x}, \mathbf{y}))\}$, and then compute the interpolations in the output space of transformer encoder $\mathbf{F}(.;\theta)$ as follows:

$$\begin{aligned} \mathbf{h} &= \mathbf{F}(\mathbf{x}_{-e};\theta), \mathbf{h}' = \mathbf{F}(\mathbf{x}_{-t};\theta) \\ \tilde{\mathbf{h}} &= \lambda\mathbf{h} + (1-\lambda)\mathbf{h}'. \end{aligned} \tag{4}$$

Here, the transformer encoder $\mathbf{F}(.;\theta)$ for both $\mathbf{x}_{-e}$ and $\mathbf{x}_{-t}$ is sharing the weights. Then we use interpolated hidden representations $\tilde{\mathbf{h}}$ as the input to the final CRF tagger.

When inferring tags on unlabeled sentences without entity triggers, we expect the trained $\mathbf{F}(.;\theta)$ to generate representation from the input $\mathbf{x} \in \mathcal{D}_u$, which can generalize well to unseen entities given the seen contextual clues, and seen entities given new surrounding context. We then use it as an input to the final CRF tagger to get tag predictions.

Computational Cost. The computation overhead of training TIN is caused by forwarding input two times into the transformers to get two different representations, which is 1.5X larger than vanilla models (e.g. BERT+TIN+CRF vs. BERT+CRF). However, the overhead of inference is same as vanilla models since the inference steps are same.

4 Experimental Setup

In this section we describe datasets along with the baseline methods and experiment settings.

4.1 Datasets

We consider three NER datasets as target tasks. We consider two datasets for a bio-medical domain: **BC5CDR** (Li et al., 2016a), **JNLPBA** (Collier and Kim, 2004) and one dataset for a general domain: **CoNLL03** (Tjong Kim Sang, 2002). For BC5CDR and CoNLL03, we also have crowd-sourced entity trigger dataset $\mathcal{D}_{HT}$ (Lin et al., 2020) to compare the quality of our automatically extracted triggers with. They randomly sample 20% of the data from each of the train sets and ask crowd-workers to select triggers for entities in those train sets. Details on datasets are discussed in Appendix A.3.

4.2 Compared Methods

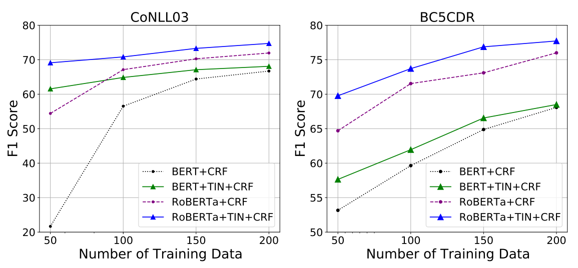
To show the effectiveness of entity triggers, we compare baseline models that learn from entity-labeled dataset $\mathcal{D}_L$ and trigger-labeled dataset $\mathcal{D}_T$ respectively. Here, we compare models under three different embedders: GloVE (Pennington et al., 2014), BERT (Devlin et al., 2019), and RoBERTa (Liu et al., 2019b). We don't include few-shot learning and semi-supervised learning due to following reasons: (1) Few-shot learning (Yang and Katiyar, 2020) is a N-way k-shot task that needs N classes, each with k training examples. This setup is restrictive because it requires a special sampling strategy since the number of training examples for the class must be fixed. (2) Semi-supervised learning (Peng et al., 2019; Shang et al., 2018; Yang et al., 2018; Liu et al., 2019a) uses additional supervisions and in-domain unlabeled data to augment train data by weakly labeling, which is different from our setup that only uses limited in-domain labeled data.

Baselines without triggers We apply the following standard models on $\mathcal{D}_L$: (1) **BLSTM+CRF** first adopts bidirectional LSTM to encode word embeddings into the representations. Then, it feeds the representations into a linear layer to get label scores, and put the scores into a CRF tagger to predict the optimal path of entity tags; (2) **CRF** considers word embeddings directly as the representations and conducts the same process of (1).

Methods with Triggers We apply the following models on $\mathcal{D}_T$: (1) **BLSTM+TMN+CRF** (Lin et al., 2020) adopts the structured self-attention layer (Lin et al., 2017) above the bidirectional LSTM to encode the sentence and entity trigger into vector representation respectively. It then learns trigger representations that can be generalized to unseen sentences to tag the named entity; (2) **TMN+CRF** adopts the structured self-attention layer directly above the embeddings, and conducts the same process of (1); (3) **TIN+CRF** is the *trigger interpolation network* with CRF layer.

5 Results and Analysis

Here we look into a series of analysis questions as follows: (1) Can `TIN` with automatically extracted triggers improve the performance of existing transformer-based models? (2) Can `TIN` with automatically extracted triggers generalize well on the unseen entities? (3) Are automatically extracted triggers work better than human-provided triggers? (4) Can we improve the performance by asking human to refine automatically extracted triggers? We present experiment settings and details in Appendix A.1- A.2.



(a) CoNLL03 (b) BC5CDR

Figure 6: Performance Comparison (F1-score) on CoNLL03 and BC5CDR by small numbers of train data instances (50, 100, 150, 200).

5.1 Performance Comparison

Here, we test all models by varying the amount of training data from 20% to 100% to show the impact of train data size. In Table 1, we report the performance of the baseline approaches and our model variants on three different datasets. Here are findings: (1) `BLSTM` harms the performance of transformer-based models while it works well with GloVE embeddings; (2) Models that receive both entities and triggers as input generally outperform the *entity-only* baselines (See `CRF` vs. `TIN+CRF`, `TMN+CRF`); (3) `TIN` outperforms `TMN`, which shows that `TIN` is doing better leveraging entity triggers for NER, and `RoBERTa+TIN+CRF` outperforms all the baselines regardless of the amount of data that is used to train it; (4) Comparing `BERT+TIN+CRF` to `BERT+CRF`, we observe a performance drop in CoNLL03. We further investigate this phenomenon and find a large drop in F1 score (from 0.82 to 0.79) for the MISC class from the `BERT+TIN+CRF` (See Table 3), which shows that triggers may provide a precision decreasing signal for the MISC type.

Performance in extreme label scarcity settings We hypothesize that our models will have larger performance gains in extreme label scarcity settings, because of their ability to leverage additional information from triggers which enables them to reap more benefits from given training data. To investigate this we observe the performance of our models and baselines starting with only 50-200 sentences to train them. Figure 6 shows the performance under the extreme label scarcity setting. Here, `TIN+CRF` achieves large performance gain in extremely low-resource setting. Specifically, we observe over 50% relative gain compared to the baseline for 50 training sentences.

Unseen entity generalization We hypothesize that our framework can generalize well to unseen

Embedder	**Method** / Train set %	**BC5CDR**					**JNLPBA**					**CoNLL03**				
		20%	40%	60%	80%	100%	20%	40%	60%	80%	100%	20%	40%	60%	80%	100%
GloVE	+ BLSTM+CRF	71.92	76.29	79.04	80.72	81.07	66.36	69.31	71.25	71.90	72.79	85.06	88.33	88.98	89.84	90.72
	+ BLSTM+TMN+CRF	**74.70**	**78.15**	**80.57**	**82.77**	**83.37**	**66.78**	**70.23**	**71.41**	**71.7**	**72.55**	**87.46**	**88.88**	**89.39**	**90.16**	**90.24**
BERT	+ BLSTM+CRF	44.51	65.88	74.23	80.65	82.56	59.26	69.39	72.04	73.24	73.26	68.60	87.09	89.42	90.20	90.86
	+ CRF	75.30	80.52	82.94	84.00	85.02	69.02	70.84	72.58	73.06	73.18	**88.61**	**90.20**	**91.10**	**91.37**	**91.48**
	+ BLSTM+TMN+CRF	74.98	78.24	82.52	82.89	84.38	69.56	71.36	72.26	72.92	73.34	86.88	88.78	90.05	90.35	90.99
	+ TMN+CRF	75.07	80.14	82.57	84.03	85.49	69.25	71.34	71.70	72.98	73.38	88.39	89.34	90.24	90.83	91.06
	+ TIN+CRF (Ours)	**77.37**	**81.40**	**83.23**	**85.25**	**85.74**	**70.48**	**72.10**	**72.81**	**73.91**	**74.83**	87.84	89.64	89.71	90.39	90.75
RoBERTa	+ CRF	82.85	84.63	86.08	86.44	87.10	71.07	72.19	73.32	73.50	75.37	90.53	91.63	91.90	92.06	93.09
	+ BLSTM+TMN+CRF	83.00	84.89	86.16	86.69	87.78	71.51	72.91	73.43	74.65	75.52	90.76	91.50	91.84	92.22	92.47
	+ TMN+CRF	83.89	85.56	86.65	87.31	87.78	71.79	72.87	73.10	74.19	75.13	90.82	91.47	92.32	92.42	92.57
	+ TIN+CRF (Ours)	**84.45**	**86.09**	**87.5**	**87.84**	**88.09**	**73.12**	**74.23**	**74.45**	**74.96**	**76.98**	**91.37**	**92.03**	**92.63**	**92.51**	**93.24**

Table 1: Performance comparison (F1-score) by different percentage usage of the train data. For TMN and TIN baselines, we use the top 2 candidate phrases from SOC with constituency parsing as triggers. Best models under each embedder are **bold**.

Encoder	**Method** / Train set %	**BC5CDR**					**JNLPBA**					**CoNLL03**				
		20%	40%	60%	80%	100%	20%	40%	60%	80%	100%	20%	40%	60%	80%	100%
BERT	+ CRF	63.93	67.95	70.36	73.43	73.66	56.81	57.00	58.43	59.49	60.11	78.65	82.32	82.24	82.90	83.36
	+ TMN+CRF	65.37	68.51	68.80	72.17	73.46	56.39	56.98	58.14	59.01	59.98	80.18	82.39	82.16	82.46	83.93
	+ TIN+CRF (Ours)	**67.10**	**69.21**	**72.32**	**73.81**	**74.65**	**57.19**	**58.73**	**59.67**	**59.90**	**60.36**	**81.52**	**82.82**	**83.01**	**83.72**	**84.66**
RoBERTa	+ CRF	72.16	74.55	75.92	76.39	76.17	58.88	60.05	61.22	62.32	63.71	82.39	84.06	84.50	84.98	85.61
	+ TMN+CRF	73.11	74.27	76.13	76.18	76.29	59.62	60.73	61.17	62.44	62.98	83.30	**85.30**	85.71	**85.47**	85.97
	+ TIN+CRF (Ours)	**73.65**	**75.78**	**77.14**	**77.34**	**76.90**	**60.85**	**61.50**	**62.37**	**63.55**	**64.11**	**84.11**	84.82	**85.84**	85.31	**87.38**

Table 2: Performance comparison (F1-score) on filtered test set by different percentage usage of the train data. Filtered test set is the set which the entities did not appear in the train and development sets.For TMN and TIN baselines, we use the top 2 candidate phrases from SOC with constituency parsing as triggers. Best models under each embedder are **bold**.

Type	BERT+CRF			BERT+TIN+CRF		
	Precision	Recall	F1-score	Precision	Recall	F1-score
LOC	0.92	0.94	0.93	0.91	0.93	0.92
MISC	0.81	0.82	**0.82**	0.75	0.84	<u>0.79</u>
ORG	0.88	0.90	0.89	0.86	0.90	0.88
PER	0.97	0.96	0.96	0.96	0.96	0.96

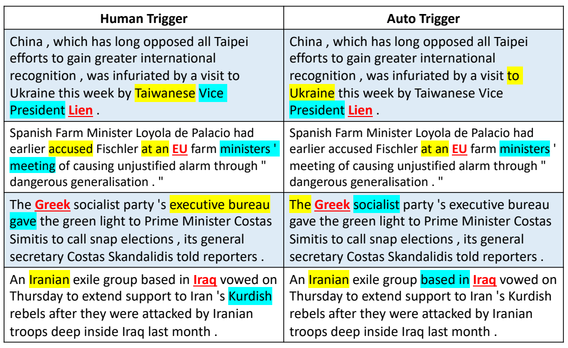
Table 3: Classification Report (F1-score) of BERT+CRF and BERT+TIN+CRF on CoNLL03.

entities not presented in the training data, attribute to prior knowledge of contextual clues. To evaluate the generalizability of the model, we create a filtered test set which contains instances with only entities that do not appear in the train data. Table 6 shows the performance of model variants on the filtered test set by different amount of training data. Here, TIN+CRF shows its effectiveness on predicting unseen entities, and much more when we compare under the small amount of training data.

5.2 Human-in-the-loop Trigger Extraction

We aim to study whether human participation in trigger creation could be helpful. First, we compare the performance of model variants trained with automatically extracted triggers (auto) and human-provided triggers (human), and check its label efficiency. Then, we conduct a small-scale experiment of trigger refinement by human annotators.

Human-curated vs. Auto Triggers. We use $\mathcal{D}_{HT}$ as the source of human triggers and use the same dataset to extract auto triggers with SOC algorithm (see Appendix Table 6). We then sample 25%, 50%, and 75% of the instances from both to construct 5%, 10%, 15% percent of our experimentation dataset (since $\mathcal{D}_{HT}$ is a 20% random sample from $\mathcal{D}_L$). One big difference between human and auto is whether the triggers are contiguous token spans or not. For example, humans are asked to annotate a group of word tokens that represent *"general"* phrase like *"had dinner at"* from the sentence *"We <u>had</u> a fantastic <u>dinner at</u> Sunnongdan."*, while a set of phrase candidates $\mathcal{P}$ from the constituency parse tree can only contain the contiguous token spans. Figure. 7 shows examples of human and auto. These examples are from CoNLL03, and auto are extracted from the entity token classifier which is trained on 20% of the train data. Tab. 4 shows that auto are comparable or even stronger than human even though created with no human labeling. The success of auto triggers might be that their impact on the entity labeling is directly at the model level, while human triggers, even if they are meaningful on the surface level, might have less impact in determining the entity label as they do not mimic what the model thinks. We manually inspect the auto and human and found that auto are consecutive while human are usually non-consecutive. Even though there could be many reasons for the sub-optimal performance of human selected triggers available in the dataset (Lin et al., 2020), we do not rule out the possibility of leveraging human expertise to help. Further case examples are presented in Appendix A.4.



Human Trigger	Auto Trigger
China , which has long opposed all Taipei efforts to gain greater international recognition , was infuriated by a visit to Ukraine this week by Taiwanese Vice President **Lien** .	China , which has long opposed all Taipei efforts to gain greater international recognition , was infuriated by a visit to Ukraine this week by Taiwanese Vice President **Lien** .
Spanish Farm Minister Loyola de Palacio had earlier accused Fischler at an **EU** farm ministers ' meeting of causing unjustified alarm through " dangerous generalisation . "	Spanish Farm Minister Loyola de Palacio had earlier accused Fischler at an **EU** farm ministers ' meeting of causing unjustified alarm through " dangerous generalisation . "
The **Greek** socialist party 's executive bureau gave the green light to Prime Minister Costas Simitis to call snap elections , its general secretary Costas Skandalidis told reporters .	The **Greek** socialist party 's executive bureau gave the green light to Prime Minister Costas Simitis to call snap elections , its general secretary Costas Skandalidis told reporters .
An Iranian exile group based in **Iraq** vowed on Thursday to extend support to Iran 's Kurdish rebels after they were attacked by Iranian troops deep inside Iraq last month .	An Iranian exile group based in **Iraq** vowed on Thursday to extend support to Iran 's Kurdish rebels after they were attacked by Iranian troops deep inside Iraq last month .

Figure 7: Top 2 highlighted auto and human triggers corresponding to the boldfaced and underlined entity.

BC5CDR	BLSTM+TMN+CRF		BERT+TIN+CRF		RoBERTa+TIN+CRF	
Percentage / **Model**	human	auto	human	auto	human	auto
5%	**26.96**	24.70	66.20	**66.50**	75.79	**76.92**
10%	**46.24**	43.54	71.25	**71.84**	80.92	**81.63**
15%	**51.29**	50.44	73.88	**74.11**	83.54	**83.87**
20%	**56.28**	54.91	75.97	**76.58**	83.88	**84.17**
CoNLL03	BLSTM+TMN+CRF		BERT+TIN+CRF		RoBERTa+TIN+CRF	
Percentage / **Model**	human	auto	human	auto	human	auto
5%	56.39	**57.95**	78.17	**78.56**	84.72	**85.71**
10%	61.89	**66.58**	81.67	**82.19**	87.80	**88.12**
15%	67.48	**69.41**	83.67	**85.13**	88.40	**89.68**
20%	71.11	**74.43**	84.88	**85.58**	89.68	**90.21**

Table 4: Performance comparison (F1-score) of TMN and TIN with human and auto triggers.

Efficiency of Human-curated vs. Auto Triggers. We conduct a study to measure how trigger extraction by human affects labeling efficiency. First, we found that labeling triggers along with the entities is 1.5 times slower than labeling the entities only. Given this observation, we compare the performance between TIN models with human and auto by holding annotation time constant. We present the study in Figure. 8. Each marker on the x-axis of the plots indicate a certain annotation time, which is represented by approximate time. First, we could find that if we ask human to get triggers from scratch, it may not work better (See RoBERTa+CRF vs. RoBERTa+TIN+CRF+human). However, if we use auto triggers, we could achieve better performance and label efficiency (See RoBERTa+TIN+CRF+auto).

Human-in-the-loop Trigger Refinement. In previous study, we could find that asking human to annotate triggers from the scratch is not efficient. Here, we further think about asking human to refine auto triggers to reflect human decisions in a less labor intensive way. For all our previous experiments, we use the top two auto triggers, which limits our capacity to make the best use of them. In this experiment, given a training set with labeled entities, we extract five auto triggers (Sec. 3.1), show them to a human in a minimal interface, and ask for relevance judgments (relevant/non-relevant). We

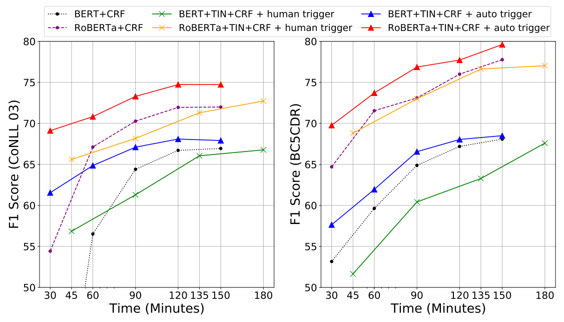


Figure 8: Performance Comparison (F1-score) by annotators' labeling time cost.

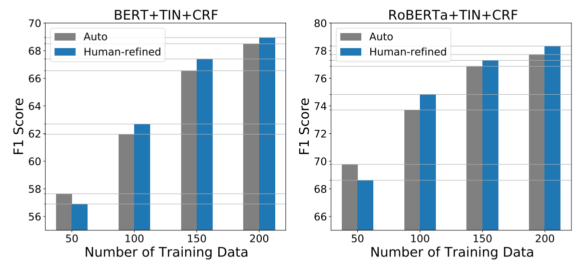


Figure 9: Performance Comparison (F1-score) on BC5CDR by different numbers of train data (50, 100, 150, 200) with auto and human-refined auto triggers.

judge relevance of the automatically extracted triggers for entities in 50 - 200 sentences. Figure. 9 shows that we get an additional performance boost with more than 50 training sentences, when human-refined auto triggers are used. It shows promise for blending human expertise with auto triggers.

6 Conclusion

In this paper, we proposed a novel two-stage framework to generate and leverage explanations for named entity recognition. It automatically extracts essentially human-readable clues in the text, which is called entity triggers, by sampling and occlusion algorithm and leverages these triggers with trigger interpolation network. The model effective learns the prior knowledge to infer the entity boundaries and types by leveraging the contextual clues. We showed that our framework, named AUTOTRIGGER, successfully generates entity triggers and effectively leverages them to improve the overall performance, especially in the label scarcity setting for technical domains where domain-expert annotations are very limited due to the high cost. Moreover, it shows better generalizability on unseen entities that do not appear in training data. We believe that this work opens up future works that can be extended to semi-supervised learning or distant supervised learning which can effectively use automatically extracted triggers to weakly label the unlabeled corpus.

Limitations

AutoTriggER could reduce human efforts on collecting rationales for NER to improve the label efficiency and unseen entity generalizability of the model. However, it has a limitation that the time cost of extracting entity triggers is pretty expensive (i.e., (1) train entity token classifier; (2) run constituency parsing to retrieve trigger candidates; and (3) run post-hoc explanation to assign importance score to each candidate). This limitation raises open questions about whether reducing total training time cost or human effort is more efficient and important.

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. In *Proc. of ICLR*.

Jason P.C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Nigel Collier and Jin-Dong Kim. 2004. Introduction to the bio-entity recognition task at JNLPBA. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA/BioNLP)*.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proc. of ACL*.

Leyang Cui, Yu Wu, Jian Liu, Sen Yang, and Yue Zhang. 2021. Template-based named entity recognition using BART. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1835–1845, Online. Association for Computational Linguistics.

G. DeJong and R. Mooney. 2004. Explanation-based learning: An alternative view. *Machine Learning*.

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

Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C. Wallace. 2020. ERASER: A benchmark to evaluate rationalized NLP models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4443–4458, Online. Association for Computational Linguistics.

Ning Ding, Yulin Chen, Xu Han, Guangwei Xu, Pengjun Xie, Hai-Tao Zheng, Zhiyuan Liu, Juanzi Li, and Hong-Gee Kim. 2021. Prompt-learning for fine-grained entity typing. *ArXiv preprint*, abs/2108.10604.

Jingfei Du, Edouard Grave, Beliz Gunel, Vishrav Chaudhary, Onur Celebi, Michael Auli, Ves Stoyanov, and Alexis Conneau. 2020. Self-training improves pre-training for natural language understanding. *arXiv preprint arXiv:2010.02194*.

John Foley, Sheikh Muhammad Sarwar, and James Allan. 2018. Named entity recognition with extremely limited data. *arXiv preprint arXiv:1806.04411*.

Matt Gardner, Yoav Artzi, Victoria Basmov, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, Nitish Gupta, Hannaneh Hajishirzi, Gabriel Ilharco, Daniel Khashabi, Kevin Lin, Jiangming Liu, Nelson F. Liu, Phoebe Mulcaire, Qiang Ning, Sameer Singh, Noah A. Smith, Sanjay Subramanian, Reut Tsarfaty, Eric Wallace, Ally Zhang, and Ben Zhou. 2020. Evaluating models' local decision boundaries via contrast sets. In *Findings of the Association for Computational Linguistics: EMNLP 2020*.

Braden Hancock, Paroma Varma, Stephanie Wang, Martin Bringmann, Percy Liang, and Christopher Ré. 2018. Training classifiers with natural language explanations. In *Proc. of ACL*.

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. 2016. Harnessing deep neural networks with logic rules. In *Proc. of ACL*.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, Proceedings of Machine Learning Research.

Sarthak Jain, Sarah Wiegreffe, Yuval Pinter, and Byron C. Wallace. 2020. Learning to faithfully rationalize by construction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4459–4473, Online. Association for Computational Linguistics.

Chengyue Jiang, Yinggong Zhao, Shanbo Chu, Libin Shen, and Kewei Tu. 2020. Cold-start and interpretability: Turning regular expressions into trainable recurrent neural networks. In *Proc. of EMNLP*.

Xisen Jin, Zhongyu Wei, Junyi Du, Xiangyang Xue, and Xiang Ren. 2020. Towards hierarchical importance attribution: Explaining compositional semantics for neural sequence models. In *Proc. of ICLR*.

Vidur Joshi, Matthew Peters, and Mark Hopkins. 2018. Extending a parser to distant domains using a few dozen partially annotated examples. In *Proc. of ACL*.

Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, Proceedings of Machine Learning Research.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proc. of NAACL-HLT*.

Dong-Ho Lee, Akshen Kadakia, Kangmin Tan, Mahak Agarwal, Xinyu Feng, Takashi Shibuya, Ryosuke Mitani, Toshiyuki Sekiya, Jay Pujara, and Xiang Ren. 2022. Good examples make a faster learner: Simple demonstration-based learning for low-resource NER. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2687–2700, Dublin, Ireland. Association for Computational Linguistics.

Dong-Ho Lee, Rahul Khanna, Bill Yuchen Lin, Seyeon Lee, Qinyuan Ye, Elizabeth Boschee, Leonardo Neves, and Xiang Ren. 2020a. LEAN-LIFE: A label-efficient annotation framework towards learning from explanation. In *Proc. of ACL*.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020b. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Jiao Li, Yueping Sun, Robin J. Johnson, Daniela Sciaky, Chih-Hsuan Wei, Robert Leaman, Allan Peter Davis, Carolyn J. Mattingly, Thomas C. Wiegers, and Zhiyong Lu. 2016a. Biocreative v cdr task corpus: a resource for chemical disease relation extraction. *Database : the journal of biological databases and curation*.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016b. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*.

Bill Yuchen Lin, Wenyang Gao, Jun Yan, Ryan Moreno, and Xiang Ren. 2021. RockNER: A simple method to create adversarial examples for evaluating the robustness of named entity recognition models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3728–3737, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Bill Yuchen Lin, Dong-Ho Lee, Ming Shen, Ryan Moreno, Xiao Huang, Prashant Shiralkar, and Xiang Ren. 2020. TriggerNER: Learning with entity triggers as explanations for named entity recognition. In *Proc. of ACL*.

Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *Proc. of ICLR*.

Liyuan Liu, Jingbo Shang, Xiang Ren, Frank Fangzheng Xu, Huan Gui, Jian Peng, and Jiawei Han. 2018. Empower sequence labeling with task-aware neural language model. In *Proc. of AAAI*.

Tianyu Liu, Jin-Ge Yao, and Chin-Yew Lin. 2019a. Towards improving neural named entity recognition with gazetteers. In *Proc. of ACL*.

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

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proc. of ACL*.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proc. of ACL*.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proc. of EMNLP*.

Yuyang Nie, Yuanhe Tian, Yan Song, Xiang Ao, and Xiang Wan. 2020. Improving named entity recognition with attentive ensemble of syntactic information. In *Findings of the Association for Computational Linguistics: EMNLP 2020*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*.

Minlong Peng, Xiaoyu Xing, Qi Zhang, Jinlan Fu, and Xuanjing Huang. 2019. Distantly supervised named entity recognition using positive-unlabeled learning. In *Proc. of ACL*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proc. of EMNLP*.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proc. of ACL*.

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. *VLDB*.

Marco Túlio 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, San Francisco, CA, USA, August 13-17, 2016*.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proc. of ACL*.

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

Sheikh Muhammad Sarwar, John Foley, and James Allan. 2018. Term relevance feedback for contextual named entity retrieval. In *CHIIR*.

Jingbo Shang, Liyuan Liu, Xiaotao Gu, Xiang Ren, Teng Ren, and Jiawei Han. 2018. Learning named entity tagger using domain-specific dictionary. In *Proc. of EMNLP*.

Chandan Singh, W. James Murdoch, and Bin Yu. 2019. Hierarchical interpretations for neural network predictions. In *International Conference on Learning Representations*.

Yuanhe Tian, W. Shen, Yan Song, Fei Xia, Min He, and Kenli Li. 2020. Improving biomedical named entity recognition with syntactic information. *BMC Bioinformatics*.

Erik F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*.

Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. 2021. Improving named entity recognition by external context retrieving and cooperative learning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1800–1812, Online. Association for Computational Linguistics.

Ziqi Wang, Yujia Qin, Wenxuan Zhou, Jun Yan, Qinyuan Ye, Leonardo Neves, Zhiyuan Liu, and Xiang Ren. 2020. Learning from explanations with neural execution tree. In *Proc. of ICLR*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proc. of EMNLP*.

Yaosheng Yang, Wenliang Chen, Zhenghua Li, Zhengqiu He, and Min Zhang. 2018. Distantly supervised NER with partial annotation learning and reinforcement learning. In *Proceedings of the 27th International Conference on Computational Linguistics*.

Yi Yang and Arzoo Katiyar. 2020. Simple and effective few-shot named entity recognition with structured nearest neighbor learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6365–6375, Online. Association for Computational Linguistics.

Qinyuan Ye, Xiao Huang, Elizabeth Boschee, and Xiang Ren. 2020. Teaching machine comprehension with compositional explanations. In *Findings of the Association for Computational Linguistics: EMNLP 2020*.

Omar Zaidan and Jason Eisner. 2008. Modeling annotators: A generative approach to learning from annotator rationales. In *Proc. of EMNLP*.

A Appendix

A.1 Experiment Settings

We implement all the baselines using PyTorch (Paszke et al., 2019) and HuggingFace (Wolf et al., 2020). We set the batch size and learning rate to 10 and 0.01 for GloVE embedder models (i.e., `GloVE+BLSTM+CRF`, `GloVE+TMN+CRF`) while we set 30 and 2e-5 for all other transformer embedder models (See Table 5). Note that for experiments in extreme low resource setting (Sec. 5.1), we set the batch size to 4 for training the models due to the extremely limited training data.

Trigger Matching Network For transformer-based `TMN` (i.e., `BERT+BLSTM+TMN+CRF`, `BERT+TMN+CRF`, etc.) we re-implement since the original repository does not support transformer embedders.

Trigger Interpolation Network For our `TIN`, we set the interpolation λ to 0.5.

Automatic Trigger Extraction For automatic trigger extraction stage, we build the entity token classifier with cased BERT-base encoder for `BERT+TIN+CRF` and RoBERTa-large for `RoBERTa+TIN+CRF`. The entity token classifier consists of the transformer encoder to encode each word token followed by a token-level linear layer that classifies each token to an entity tag. We use a batch size of 16 and learning rate of 1e-4 for training. For experiments under extreme low resource setting, we set batch size to 4 similar to the `TIN` models. To run context sampling in the SOC algorithm, we use a LSTM language model which is pre-trained on the training data.

A.2 Evaluation Metrics

We evaluate our framework by recall (R), precision (P), and F1-score (F1), though only report F1 in these experiments. Recall (R) is the number of correctly recognized named entities divided by the total number of named entities in the corpus, and precision (P) is the number of correctly recognized named entities divided by the total number of named entities recognized by the framework. A recognized entity is correct if both its boundary and its entity type are exact matches to the annotations in the test data. F1-score is the harmonic mean of precision and recall.

Embedder	GloVE	Transformer	
		BERT	RoBERTa
batch size	10	30	30
learning rate	0.01	2e-5	2e-5
epochs	10	10	10
LSTM hidden dimension	200	-	-

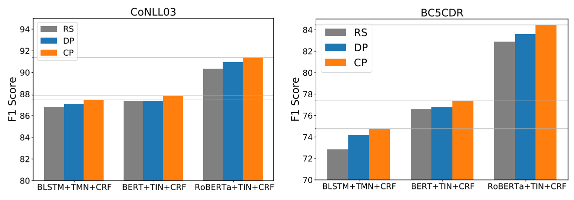
Table 5: Experimental setting details.

A.3 Data Statistics

BC5CDR (Li et al., 2016a) is a bio-medical domain NER dataset from BioCreative V Chemical and Disease Mention Recognition task. It has 1,500 articles containing 15,935 CHEMICAL and 12,852 DISEASE mentions. **JNLPBA** (Collier and Kim, 2004) is a bio-medical domain NER dataset for the Joint Workshop on NLP in Biomedicine and its Application Shared task. It is widely used for evaluating multiclass biomedical entity taggers and it has 14.6K sentences containing PROTEIN, DNA, RNA, CELL LINE and CELL TYPE. **CoNLL03** (Tjong Kim Sang, 2002) is a general domain NER dataset that has 22K sentences containing four types of general named entities: LOCATION, PERSON, ORGANIZATION, and MISCELLANEOUS entities that do not belong in any of the three categories.

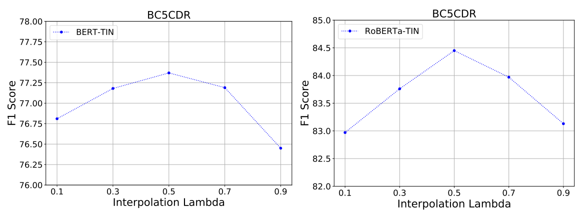
A.4 Performance Analysis

Trigger Candidate Variants. In Sec 3.1, we first constructed a set of phrase candidates $\mathcal{P}$ for which the importance score is computed. To show the efficacy of constituency parsing for constructing trigger candidates, we conduct an ablation study on different variants of it. For the construction, we compare three variants: (1) `RS` is random selection. It randomly chooses n contiguous tokens to be grouped as a phrase for k times. Consequently, $\mathcal{P}$ is composed of k random spans. (2) `DP` is dependency parsing. Here, to generate $\mathcal{P}$, we first parse the input sentence using dependency parsing. Then, we traverse from the position of entity mention in the input sentence using depth-first-traversal and get a list of tokens visited for each hop up to 2-hops. Finally, for each hop, we convert the list of tokens to a list of phrases by merging the tokens that are contiguous into a single phrase. (3) `CP` is constituency parsing, which is our current method (see Sec. 3.1). We expect each variant to provide different syntactic signals to our framework. Figure 10 shows the model's performance with triggers that have been selected from different sets of phrase candidates. As we can see, constituency parsing



(a) CoNLL03 (b) BC5CDR

Figure 10: Performance comparison (F1-score) of entity+trigger baselines on 20% training dataset of CoNLL03 and BC5CDR with different trigger candidate variants.



(a) BERT+TIN+CRF (b) RoBERTa+TIN+CRF

Figure 11: Performance comparison (F1-score) of entity+trigger baselines on 20% training dataset of BC5CDR with different interpolation weight λ.

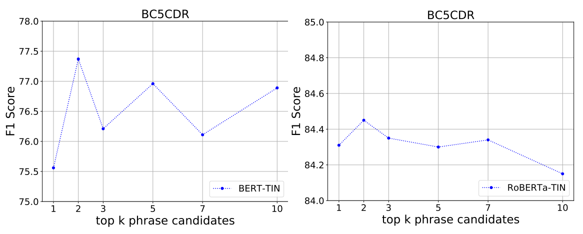
yields consistently better performance by providing better quality of syntactic signals than others.

Sensitivity Analysis of interpolation hyper-parameter (λ). In Sec 3.2, we linearly interpolated two different sources of knowledge by weight λ 0.5. To show how the weight λ affects the performance, we conduct an ablation study on different λ distribution. As we can see from Figure. 11, the framework achieves the highest performance when λ is set to 0.5. It supports that the model achieves the best when we interpolate the entity and trigger knowledge in equal.

Effect of number of triggers In Sec. 3.1, we pick the top k candidate phrases with the highest importance score as the entity triggers after obtaining the importance score for all phrase candidates. For our main experiment, we use top 2 candidate phrases (see Table 1). To show how the number of triggers affects the performance, we conduct an ablation study on model performance by different k. As we can see from Figure. 12, the framework achieves the highest performance when we use top 2 phrase candidates as triggers.

A.5 Related Works

NER with Additional Supervision Previous and recent research has shown that encoding syntactic information into NER models compensate for the

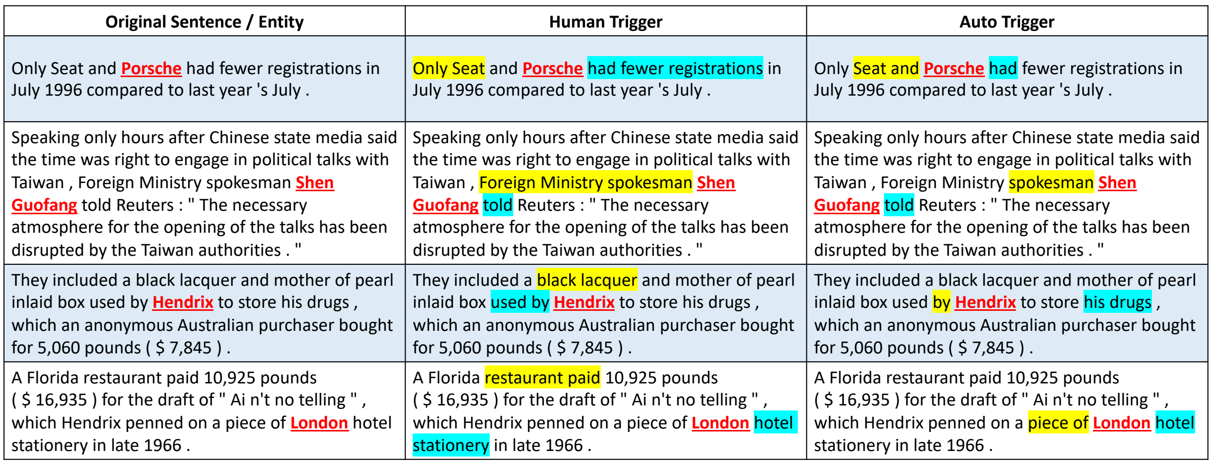


(a) BERT+TIN+CRF (b) RoBERTa+TIN+CRF

Figure 12: Performance comparison (F1-score) of entity+trigger baselines on 20% training dataset of BC5CDR with different number of triggers k.

lack of labeled data (Tian et al., 2020). The improvement is consistent across word embedding based encoding (e.g. biLSTM) as well as unsupervised language model based encoding (e.g. BioBERT (Lee et al., 2020b)) of the given text. Typically, the external information that is encoded include POS labels, syntactic constituents, and dependency relations (Nie et al., 2020; Tian et al., 2020). The general mechanism to include linguistic information into NER model is to represent them using word vectors and then concatenate those representations with the original text representation. This approach fails to identify the importance of different types of syntactic information. Recently, Tian et al. (2020) and Nie et al. (2020) both showed that key-value memory network (KVMN) (Miller et al., 2016) are effective in capturing importance of linguistic information arising from different sources. KVMN has been shown to be effective in leveraging extra information, such as knowledge base entities, to improve question answering tasks. Before applying KVMN, contextual information about a token is encoded as the key and syntactic information are encoded as values. Finally, weights over the values are computed using the keys to obtain a representation of the values and concatenate it with the context features. Our approach uses token level features extracted by an explanation generation model, but later train to be able to pick-up those explanations directly from the text at inference time.

Limited Training Data for NER. The simplest way to approach the problem of limited data for NER is to use dictionary based weak supervision. An entity dictionary is used to retrieves unlabeled sentences from a corpus and weakly label them to create additional noisy data. This approach suffers from low recall as the training data covers a limited number of entities. The models tend to



Original Sentence / Entity	Human Trigger	Auto Trigger
Only Seat and **Porsche** had fewer registrations in July 1996 compared to last year 's July .	Only Seat and **Porsche** had fewer registrations in July 1996 compared to last year 's July .	Only Seat and **Porsche** had fewer registrations in July 1996 compared to last year 's July .
Speaking only hours after Chinese state media said the time was right to engage in political talks with Taiwan , Foreign Ministry spokesman **Shen Guofang** told Reuters : " The necessary atmosphere for the opening of the talks has been disrupted by the Taiwan authorities . "	Speaking only hours after Chinese state media said the time was right to engage in political talks with Taiwan , Foreign Ministry spokesman **Shen Guofang** told Reuters : " The necessary atmosphere for the opening of the talks has been disrupted by the Taiwan authorities . "	Speaking only hours after Chinese state media said the time was right to engage in political talks with Taiwan , Foreign Ministry spokesman **Shen Guofang** told Reuters : " The necessary atmosphere for the opening of the talks has been disrupted by the Taiwan authorities . "
They included a black lacquer and mother of pearl inlaid box used by **Hendrix** to store his drugs , which an anonymous Australian purchaser bought for 5,060 pounds ($ 7,845) .	They included a black lacquer and mother of pearl inlaid box used by **Hendrix** to store his drugs , which an anonymous Australian purchaser bought for 5,060 pounds ($ 7,845) .	They included a black lacquer and mother of pearl inlaid box used by **Hendrix** to store his drugs , which an anonymous Australian purchaser bought for 5,060 pounds ($ 7,845) .
A Florida restaurant paid 10,925 pounds ($ 16,935) for the draft of " Ai n't no telling " , which Hendrix penned on a piece of **London** hotel stationery in late 1966 .	A Florida restaurant paid 10,925 pounds ($ 16,935) for the draft of " Ai n't no telling " , which Hendrix penned on a piece of **London** hotel stationery in late 1966 .	A Florida restaurant paid 10,925 pounds ($ 16,935) for the draft of " Ai n't no telling " , which Hendrix penned on a piece of **London** hotel stationery in late 1966 .

Figure 13: Case examples of auto trigger and human trigger. Entities are **bold** and underlined with red color, and its triggers are highlighted. Different triggers are color-coded.

bias towards the surface form of the entities it has observed in the dictionary. There has also been approaches to retrieve sentences from a large corpus that are similar to sentences in the low-resource corpus to enrich it. These self-training approaches have been shown to be effective both in extremely limited data (Foley et al., 2018; Sarwar et al., 2018) as well as limited data scenario (Du et al., 2020). Even though these data enhancement approaches explore a corpus to find related data cases, they do not exploit the explanation-based signals that is available within the limited data.

Learning from Explanations. Recent works on Explainable AI are primarily focused on debugging the black box models by probing internal representations (Adi et al., 2017; Conneau et al., 2018), testing model behavior using challenge sets (McCoy et al., 2019; Gardner et al., 2020; Ribeiro et al., 2020), or analyzing an impact of input examples by input perturbations or influence function looking at input examples (Ribeiro et al., 2016; Koh and Liang, 2017). However, for an explanation of the model to be effective, it must provide not only the reasons for the model's prediction but also suggestions for corresponding actions in order to achieve an objective. Efforts to cope with this issue by incorporating human explanations into the model are called Explanation-based learning (DeJong and Mooney, 2004). These works are aiming to exploit generalized explanations for drawing inferences from unlabeled data while maintaining model transparency. Most prior works on explanation-based learning are mainly focused on facilitating logical rules as an explanation. They use such rules to create weak supervision (Ratner et al., 2017) and regularize posterior (Hu et al., 2016, 2017). Another form of explanations can be specific words in the sentence which aligns to our work. Notable work in this line asks annotators to highlight important words, then learn a generative model over parameters given these rationales (Zaidan and Eisner, 2008).

Dataset	Entity Type	Original $\mathcal{D}_L$	Crowd-sourced trigger $\mathcal{D}_{HT}$	
		# of Entities	# of Entities	# of Human Triggers
CONLL 2003	PER	6,599	1,608	3,445
	ORG	6,320	958	1,970
	MISC	3,437	787	2,057
	LOC	7,139	1,781	3,456
	Total	23,495	5,134	10,938
BC5CDR	DISEASE	4,181	906	2,130
	CHEMICAL	5,202	1,085	1,640
	Total	9,383	1,991	3,770
JNLPBA	PROTEIN	27,802	-	-
	DNA	8,480	-	-
	RNA	843	-	-
	CELL LINE	3,429	-	-
	CELL TYPE	6,191	-	-
	Total	46,745	-	-

Table 6: Train data statistics.