AutoTriggER: Named Entity Recognition with Auxiliary Trigger Extraction

Anonymous ACL submission

Abstract

Deep neural models for low-resource named entity recognition (NER) have shown impressive results by leveraging distant supervision or other meta-level information (e.g. explanation). However, the costs of acquiring such additional information are generally prohibitive, especially in domains where existing resources (e.g. databases to be used for distant supervision) may not exist. In this paper, we present a novel two-stage framework (AutoTriggER) to improve NER performance by automatically generating and leveraging "*entity triggers*" which are essentially human-readable clues in the text that can help guide the model to make better decisions. Thus, the framework is able to both create and leverage auxiliary supervision by itself. Through experiments on three well-studied NER datasets, we show that our automatically extracted triggers are well-matched to human triggers, and AutoTriggER improves performance over a RoBERTa-CRF architecture by nearly 0.5 F1 points on average and much more in a low resource setting.[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. Recent attempts addressing label scarcity 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), and labeling explanations (Hancock et al., 2018; Wang et al., 2020; Ye et al., 2020; Lin et al., 2020; Lee et al., 2020).

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

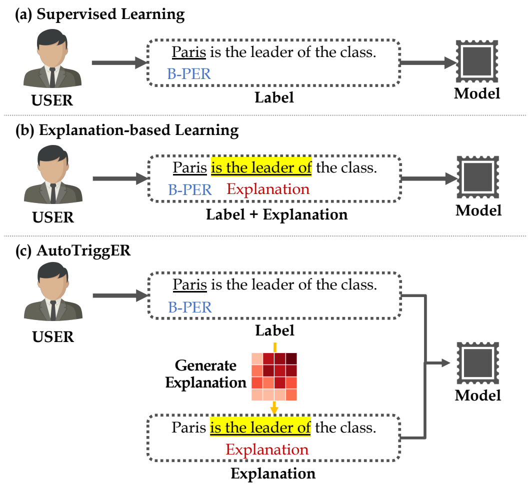


Figure 1: Existing explanation-based learning frameworks mostly rely on humans provided labeling explanations while our framework automatically generates and leverages explanations to NER.

In particular, prior works on label-efficient learning for classification (e.g., relation extraction) (Hancock et al., 2018; Wang et al., 2020; Zhou et al., 2020) and question answering (Ye et al., 2020) with explanations show that human provided explanations as auxiliary supervision signals are more cost-effective than collecting label-only annotations for larger number of instances. For the NER task, Lin et al. (2020) introducted the concept of an *entity trigger*, an effective way to represent explanations for the labeling decisions. An entity trigger is defined as a group of words in a sentence that helps to *explain* why humans would assign a type to an entity in a sentence, and it serves as an effective proxy of rationale, as shown in Figrue 1 (a) vs. (b).

Prior works primarily use a limited number of *crowd-soured* triggers for improving data (label) efficiency of model training. While such human-curated auxiliary supervision are of high quality, the crowd-sourcing procedure can be very expensive and time-consuming. This largely limits the scale and domains of the collected entity triggers. In addition, trigger-aware NER models (e.g., Trigger Matching Networks (Lin et al., 2020)) are built on conventional sequence tagging architectures, e.g., BLSTM-CRFs (Lample et al., 2016), while recent NER models are incorporating pre-trained language models as contextualized embedding, which can be highly beneficial for low-resource languages. In this paper, we propose a novel two-stage NER framework, named AUTOTRIGGER, that *automatically* generates and exploits entity triggers as explainable inductive bias to enhance NER models with little human effort (see Figure 1 (c)).

The *first* stage of our framework (Sec. 3.2) aims to *automatically extract entity triggers* using a saliency map technique based on input perturbations. Here, we propose to exploit the syntactic features of sentences for assigning importance scores to a group of input tokens such that we can extract useful entity triggers as auxiliary supervision. Specifically, for a given sentence and a target entity in it, we first extract phrases from its *constituency parsing tree* (Joshi et al., 2018) to form a collection of trigger candidates. Then, we score each trigger candidate by testing its ability to predict the target entity in a variety of sampled contexts. The rationale here is the intuition that a better trigger should be robust and help recognize the target entity in many different contexts. Here, we compare the system's ability to identify the target entity in versions of the sentence *with* and *without* the candidate trigger; if a trigger is indeed a meaningful clue, then removing it should cause a noticeable drop in score.

The *second* stage (Sec. 3.3) focuses on how to use our triggers as structured priors to reinforce the model to focus on 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 as the input to learn a mixed sentence representation as the input to standard sequence labeling. 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).

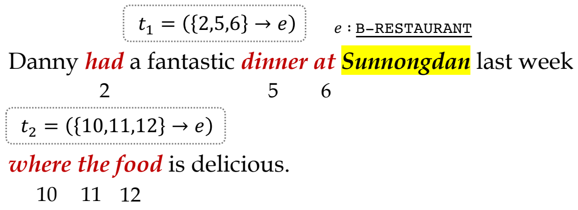


Figure 2: **Example of entity trigger.** Entity trigger t_i is a cue phrase toward the entity e in the sentence, which is represented by a set of corresponding word indices. Both entity triggers (t_1,t_2) are associated to the same entity e ("*Sunnongdan*") typed as restaurant.

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 the strong performance especially in low-resource setting for technical domains where expert annotations are limited due to the high cost. In the low-resource setting ranging from extreme to moderate, assuming a task that needs to be annotated from scratch, our model gains more than 3-4 F1 score on average.

2 Background and Formulation

We consider the problem of automatically extracting cue phrases as *entity triggers* (Lin et al., 2020) and using them to improve NER models. In this section, we introduce basic concepts about named entity recognition, entity triggers and trigger-labeled datasets. We then formally introduce our goal — creating trigger-labeled NER datasets without human annotation and then developing a learning framework that uses them to improve NER models.

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.

Entity Trigger. Lin et al. (2020) introduce the concept of "entity trigger," a novel form of explanatory annotation for NER, which is defined as a group of words that can help explain the recognition process of an entity in the sentence. For example, in Figure 2, "*had ... dinner at*" and "*where the food*" are two distinct triggers associated with the RESTAURANT entity "Sunnongdan." These explanatory cue phrases enable NER models to interpret a particular prediction and help them to generalize in a low-resource learning setting. Formally, given a particular NER example $(\mathbf{x}, \mathbf{y})$, we have T denoting the set of entity triggers for that example. Each trigger $t_i \in T$ is associated with an entity e and a set of word indices $\{w_i\}$. That is, $t = (\{w_1, w_2, \dots\} \to e)$ represents an entity trigger, e.g., $t_1 = \{2, 5, 6\} \to e$ in Figure 2. 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.

Our goal. Prior works mainly focus on creating $\mathcal{D}_T$ via manual annotation. Although trigger-labeled human annotations are cost-effective than entity-only annotations, they are still expensive and need domain experts for specialized domains. Therefore, in this work, we focus on how to *automatically* create such a trigger-labeled dataset $\mathcal{D}_T$ from $\mathcal{D}_L$ without manual effort, and then we propose a more label-efficient learning framework to use such $\mathcal{D}_T$ to improve NER models.

3 Approach

This section introduces the concepts in AUTOTRIGGER, and provides details of the framework design. We first present an overview of our AUTOTRIGGER framework (Sec. 3.1) and then discuss each of its components in detail (Sec. 3.2- 3.3).

3.1 Framework Overview

AUTOTRIGGER is a two-stage architecture that begins with a *automatic trigger extraction* stage followed by a *trigger interpolation network* (TIN). It automatically extracts and scores entity trigger phrases in the first stage (Sec. 3.2) and uses them in the later stage to learn the NER model (Sec. 3.3). Prior work (Lin et al., 2020) on incorporating such entity triggers focused on encoding human-provided entity triggers. In contrast, AUTOTRIGGER automatically generates triggers and directly uses them for learning (Figure. 3). Note that once we train the NER model, it is able to tag an entity token sequence without trigger extraction.

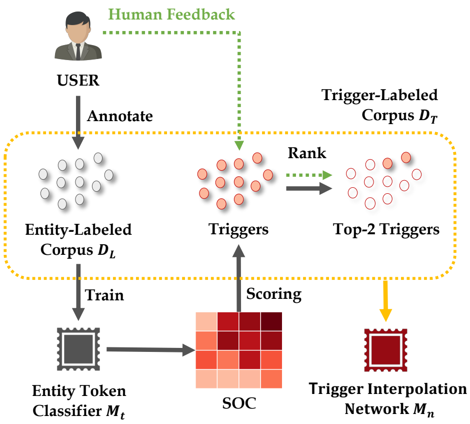


Figure 3: **Overview of AUTOTRIGGER.** It trains an entity-token classifier $\mathcal{M}_t$ with entity-labeled corpus $\mathcal{D}_L$ and uses the sampling-and-occlusion (SOC) algorithm to extract triggers. There is a provision for leveraging human feedback in the framework for refining automatically generated triggers. Trigger Interpolation Network (TIN) learns the NER model from the trigger-labeled corpus. At inference time we do not need to extract triggers and only use the NER model.

Thus we do not have the additional complexity for trigger extraction at inference time.

3.2 Automatic Trigger Extraction

Automatic trigger extraction is the first stage of our AUTOTRIGGER framework. To extract triggers, here we adopt the *sampling and occlusion* (SOC) algorithm (Jin et al., 2020), which is a saliency map technique for model interpretation. Previous works on such input analysis techniques primarily focus on modeling the relative importance of each input token based on its 1) attention intensity (Li et al., 2016b), 2) gradients (Ribeiro et al., 2016) or 3) the changes of the output by excluding it from the input (Koh and Liang, 2017). These methods can indeed produce useful explanations for some sentence classification tasks such as sentiment analysis, however, they are not well-aligned to our desired entity triggers — a group of input tokens that often poses structural constraints to a target entity.

In contrast, SOC aims to compute context-independent phrase-level importance for sequence classification tasks such as sentiment analysis and relation extraction (Jin et al., 2020). We reformulate and apply this technique for a sequence tagging task and retrieve important phrases as entity triggers. Given an input instance of the labeled corpus $(\mathbf{x_i}, \mathbf{y_i}) \in \mathcal{D}_L$, we consider four primary steps to generate entity triggers: 1) phrase candidate $\mathcal{P}$, 2) entity token classifier $\mathcal{M}_t$, 3) phrase scoring, and 4) phrase selection.

Phrase Candidate. Given a training sentence, we construct a constituency parse tree and consider the set of phrase nodes $\mathcal{P}$ from the tree as auto trigger candidates. Figure. 4 shows auto trigger candidates generated from constituency parsing of a sentence. The target entity mention "*Cary Moon*" is not included as a candidate. Note that the original SOC computes the word-level scores and extends to phrases by agglomerative clustering. Since clustering creates a large number of combinations of words to construct phrases, output phrases can be incomplete and noisy. By limiting the search space to a set of complete phrases, we could avoid such noisy triggers. Mathematically, given an input instance $(\mathbf{x_i}, \mathbf{y_i}) \in \mathcal{D}_L$ and a target entity $e \in \mathbf{x_i}$, we generate a set of phrase candidate $\mathcal{P} = \{\mathbf{p_i}\}$ where $\mathbf{p_i} = (w_s, w_e)$ and (w_s, w_e) is denoting 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 constituency parsing and collect $\mathbf{p_i}$ corresponding to phrase nodes of the constituency-based parse tree. To avoid considering target entity as part of an entity trigger, we discard a set of entity-overlapped phrases $\{\mathbf{p_j} | e \in \mathbf{p_j}\}$.

Entity Token Classifier. The second component is entity token classifier $\mathcal{M}_t$, which is a neural network for modeling the scoring module. Given an input sentence $\mathbf{x_i} = [x_i^{(1)}, x_i^{(2)}, \ldots x_i^{(n)}]$, $\mathcal{M}_t$ 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. After training $\mathcal{M}_t$ with labeled corpus $\mathcal{D}_L$, we can derive the prediction score function s of the target entity e in the input sentence $\mathbf{x_i} \in \mathcal{D}_L$. Let the conditional probability $\mathbb{P}(\mathbf{y}|\mathbf{x})$ denote the output of $\mathcal{M}_t$. Then, the prediction score function s of the target entity e is computed as the average conditional probability over tokens of the target entity e as follows:

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

Phrase Scoring. We use the phrase candidate $\mathcal{P}$ and prediction score function s of the $\mathcal{M}_t$ to measure the importance score of each phrase $\mathbf{p}$ towards target entity e by sampling and occlusion (SOC) algorithm. SOC is composed of two core methods: (1) *input occlusion*, (2) *context sampling*.

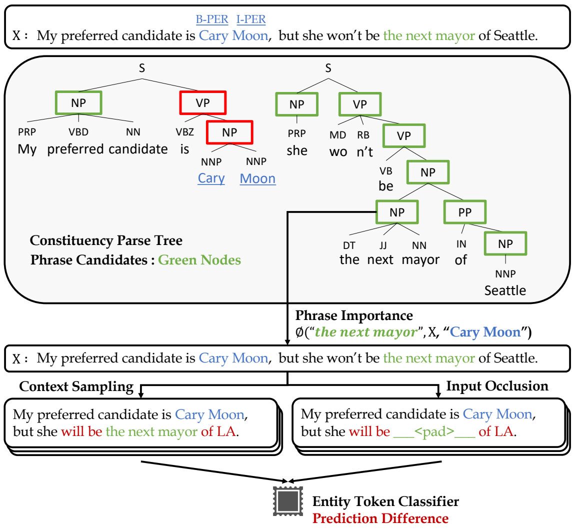


Figure 4: **Overview of the Sampling and Occlusion (SOC).** It creates a set of phrase candidates with phrase nodes of the constituency parse tree, and then computes the phrase importance by average prediction difference between context sampled sentences and its phrase-masked sentences.

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 $0_\mathbf{p}$:

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

For example, in Figure. 4, "*the next mayor*" is replaced by pad tokens to compute its importance towards the entity "*Cary Moon*". However, the importance score $\phi(\mathbf{p}, \mathbf{x}, e)$ from equation 2 has a drawback that the $\mathbf{p}$ is dependent on context words around $\mathbf{p}$. It may neglect the fact that the importance score of $\mathbf{p}$ can vary depending on which context words are around the $\mathbf{p}$.

To eliminate the dependence, *context sampling* 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 trained language model $p(\hat{x}_\delta | x_{-\delta})$ and obtains a set of context word replacements $\mathcal{S}$. 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:

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

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. After obtaining the importance score $\phi(\mathbf{p}, \mathbf{x}, e)$ for all phrase candidates $\mathcal{P} = \{\mathbf{p_i}\}$, we pick the top k candidate phrases with the highest importance score as the entity 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 $T(\mathbf{x_i}, \mathbf{y_i})$ to create a form $\{(\mathbf{x_i}, \mathbf{y_i}, T(\mathbf{x_i}, \mathbf{y_i}))\}$.

3.3 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 $\{(\mathbf{x}, \mathbf{y}, T(\mathbf{x}, \mathbf{y}))\}$. Since triggers are the most important non-entity words in an input sentence, we want to strengthen such prior knowledge in a neural network model, instead of solely memorizing the entity words themselves. However, in many training instances, the entity words themselves are sufficient to learn an entity type, diluting the model's need to understand the surrounding context (including any triggers). To force the model to learn both words typically involved in entities as well as these "most important" trigger phrases, we employ two separate masking passes when learning our model's embeddings, one masking the entity words and one masking the triggers. We then linearly interpolate the entity-masked representation and the trigger-masked representation to force the model to understand the impact of each representation for predicting the entity type.

TIN encodes the input with a transformer encoder $\mathbf{F}(.;\theta)$ and feeds the output to a CRF tagger. This part is similar to a standard transformer-based architecture for NER. Our proposal is to create two different representations of a token in a sequence and interpolate them. Figure 5 shows how a transformer architecture is used to serve this purpose. Specifically, for a given input instance $\{(\mathbf{x}, \mathbf{y}, T(\mathbf{x}, \mathbf{y}))\}$, we first create entity-masked sentence $\mathbf{x}_{-e}$ and trigger-masked sentence $\mathbf{x}_{-t}$, and then compute the interpolations in the output space of transformer encoder $\mathbf{F}(.;\theta)$ as follows:

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

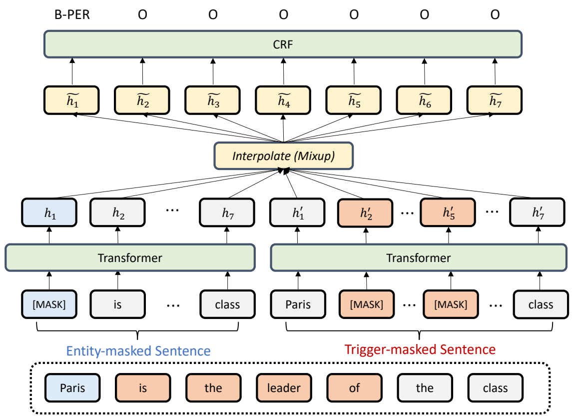


Figure 5: **Overview of the Trigger Interpolation Network (`TIN`).** Given an input sentence we create an Entity-masked sentence and a Trigger-masked Sentence. Then we interpolate token level representations h_i and h'_i to create new hidden state representation, $\tilde{h}_i$. Interpolated hidden representations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

Here, the transformer encoder $\mathbf{F}(.;\theta)$ for both $\mathbf{x}_{-e}$ and $\mathbf{x}_{-t}$ is sharing the weights. Then we use $\tilde{\mathbf{h}}$ as the input to the final CRF tagger. When inferencing tags on unlabeled sentences which have no entity triggers, we expect the trained $\mathbf{F}(.;\theta)$ is enforced to find the entity and trigger information from the input $\mathbf{x} \in \mathcal{D}_u$ and infuse both for generating enriched-information output. We then use it as an input to the final CRF tagger to get predictions.

4 Experimental Setup

In this section we describe datasets along with the baseline methods followed by experimental details.

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). Details are presented in Appendix A.3

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 sets. Data statistics are shown in Tab. 5.

4.2 Compared Methods

To show the effectiveness of entity triggers, we compare models that have same base model but use

Method / Percentage	BC5CDR					JNLPBA					CoNLL03				
	20%	40%	60%	80%	100%	20%	40%	60%	80%	100%	20%	40%	60%	80%	100%
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**
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
BERT+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**
RoBERTa+CRF	82.85	85.63	87.08	87.44	87.80	72.07	73.19	74.32	74.50	76.37	**91.53**	91.93	**92.90**	**92.96**	93.09
TMN	**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-TIN	**77.37**	**81.40**	**83.23**	**85.25**	**85.74**	**69.48**	**71.10**	**72.81**	**73.71**	**73.83**	87.84	89.64	89.71	90.39	90.75
RoBERTa-TIN	**84.45**	**86.09**	**87.5**	**87.84**	**88.09**	**73.12**	**74.23**	**74.45**	**74.76**	**76.98**	91.37	**92.03**	92.03	92.51	**93.24**

Table 1: Performance comparison (F1-score) of named entity recognition on BC5CDR, JNLPBA, and CoNLL03 datasets by different percentage usage of the train data. For entity+trigger baselines, we use the top 2 candidate phrases from SOC with constituency parsing as triggers. Best models for each encoder (BLSTM , BERT , RoBERTa) are **bold**.

different training data. Here, we present baseline models that learn $\mathcal{D}_L$ and $\mathcal{D}_T$ respectively.

Entity-Only Baseline Models. We apply the following models on $\mathcal{D}_L$: (1) **BLSTM+CRF** adopts bidirectional LSTM on the external word vectors from GloVE (Pennington et al., 2014) to produce token embeddings, which are fed into a CRF tagger to predict the optimal path of entity tags. (2) **BERT+BLSTM+CRF** extends the BLSTM+CRF by replacing the word vectors from GloVE with contextualized embeddings from pre-trained language model BERT (Devlin et al., 2019). (3) **BERT+CRF** adopts a token-level classifier on top of the BERT. Token-level classifier is a linear layer that takes as input the last hidden state of the sequence. Here, we feed the output of token-level classifier into a CRF tagger to make entity tag prediction. (4) **RoBERTa+CRF** replaces the BERT of BERT+CRF with RoBERTa (Liu et al., 2019b) which is a robustly improved BERT.

Entity+Trigger Baseline Models. We apply the following models on $\mathcal{D}_T$: (1) **TMN** (Lin et al., 2020) first adopts the structured self-attention layer (Lin et al., 2017) above the bidirectional LSTM, which uses GloVE for embeddings, to encode the sentence and entity trigger into vector representation respectively. Then, it jointly learns trigger representations and a soft matching module with self-attention such that can generalize to unseen sentences easily for tagging named entities. (2) **BERT-TIN** is *trigger interpolation network* where the transformer encoder $\mathbf{F}(.;\theta)$ is BERT. (3) **RoBERTa-TIN** is also *trigger interpolation network* where $\mathbf{F}(.;\theta)$ is RoBERTa.

4.3 Implementation Details

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 BLSTM encoder models (i.e., `BLSTM+CRF`, `TMN`, `BERT+BLSTM+CRF`) while we set 30 and 2e-5 for all other transformer models (i.e., `BERT+CRF`, `RoBERTa+CRF`, `BERT-TIN`, `RoBERTa-TIN`). For `TIN`, we set the interpolation λ to 0.5. For automatic trigger extraction stage, we set the batch size and learning rate to 16 and 1e-4 for training the entity token classifier model. To run context sampling in the SOC algorithm, we use a LSTM language model which is pre-trained on the training data. `TIN` takes 2X longer than baselines since it needs to extract triggers using SOC algorithm. Note that for experiments in extreme low resource setting (Sec. 5.2), we set the batch size to 4 for both training `TIN` and entity token classifier due to the extremely limited training data.

5 Results and Performance Analysis

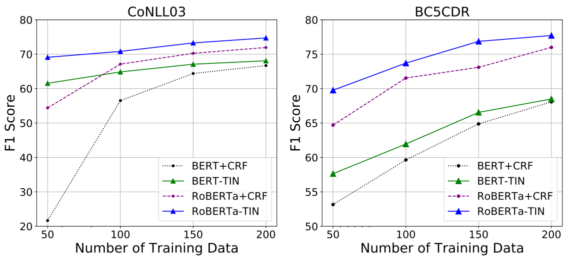
We first compare the overall performance of all baseline models and our proposed framework. Here, we test all models by varying the amount of training data from 20% to 100% to show the impact of train data size. We then discuss the effectiveness of our framework in an extremely low resource setting, assuming a task that needs to be annotated from scratch. Next, we provide a comparison of auto-triggers with human-triggers, and further show that auto-triggers can be more useful when a human judge provides binary feedback on their utility. For the ablation study, we investigate how the different variants of creating a set of trigger candidates, sensitivity of interpolation hyperparameter (λ), and number of triggers affect our framework.

5.1 Performance Comparison

In Table 1, we report the performance of the baseline approaches and our model variants on three dif-

Type	BERT-CRF			BERT-TIN		
	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	0.79
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 2: Classification Report (F1-score) of BERT-CRF and BERT-TIN on 100% CoNLL03.



(a) CoNLL03 (b) BC5CDR

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

BC5CDR	TMN		BERT-TIN		RoBERTa-TIN	
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	TMN		BERT-TIN		RoBERTa-TIN	
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 3: Performance comparison (F1-score) of entity+trigger baselines on BC5CDR and CoNLL03 with human and auto triggers.

ferent datasets. We observe that models that receive both entities and triggers as input generally outperform the *entity-only* baselines. RoBERTa-TIN outperforms all the baselines in domain-specific datasets BC5CDR and JNLPBA regardless of the amount of data that is used to train it. We only observe a performance drop in CoNLL03 when the amount of data is in the lower range. We further investigated this phenomenon and found a large drop in F1 score (from 0.82 to 0.79) for the MISC class from the RoBERTa-TIN model as shown in Table 2. Auto triggers provided a precision decreasing signal for the MISC entity type.

5.2 Performance under Low-resource Setting

We hypothesize that our models will have larger performance gains in extreme low-resource settings, because of their ability to leverage additional information from auto-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 of our models and baselines under the extreme low-resource setting. Even though our best model, RoBERTa-TIN, was on par with the baseline, RoBERTa+CRF, in the CoNLL03 dataset in the previous setting, it achieves large performance gain in extremely low-resource setting. Specifically, we observe over 50% relative gain compared to the baseline for 50 training sentences. For the BC5CDR dataset we observe persistent performance gain.

5.3 Human-in-the-loop Trigger Extraction

Human-curated vs. Auto Triggers. We compare the performance of our model variants trained with automatically extracted triggers (`auto`) and human-provided (crowd-sourced) triggers (`human`). We use $\mathcal{D}_{HT}$ as the source of human triggers and use the same dataset to extract auto triggers with `SOC` algorithm. 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 had a fantastic dinner at 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. 3 shows that auto triggers are comparable or even stronger than human-curated triggers even though created with no human labeling. The success of auto triggers can be attributed to their capacity of directly altering the entity labels. 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 lesser impact in determining the entity label as they do not mimic what the model thinks. We manually inspected the

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 underlined entity.

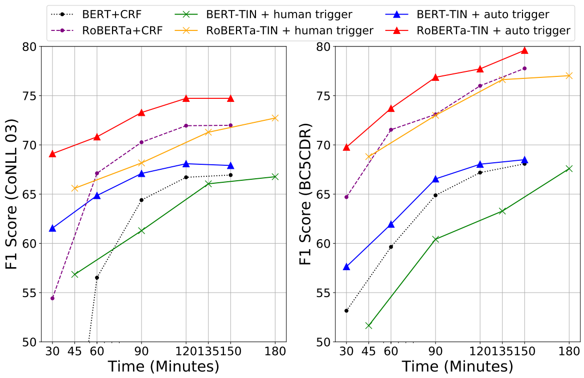
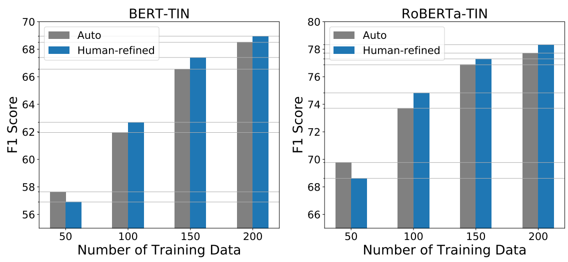


Figure 8: Performance Comparison (F1-score) by annotators' labeling time cost.

auto triggers and human triggers and found that auto triggers are consecutive while human-curated triggers 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.

Label Efficiency. We conduct experiments to demonstrate the label efficiency of our model. We found that the time for labeling on instance plus providing entity triggers takes 1.5X times more time than just simply providing a label. 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. We see that our model not only is more time and label efficient compared to both entity baselines and entity+trigger baselines with human triggers, but it also outperforms.



(a) BERT-TIN (b) RoBERTa-TIN

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.

Human-in-the-loop Trigger Refinement. We conduct a small-scale experiment of trigger refinement by human annotators. 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. 3.2), show them to a human in a minimal interface, and ask for relevance judgments (relevant/non-relevant). We judged relevance of the automatically extracted triggers for entities in 50, 100, 150, and 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 in training. This small scale annotation 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 leverage these triggers with trigger interpolation network. We show that our framework, named AUTOTRIGGER, successfully generates entity triggers and effectively leverages them to improve the overall performance, especially in the low-resource setting for technical domains where domain-expert annotations are very limited due to the high cost. Extensive experiments on three public datasets prove the effectiveness of our framework. 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.

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

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)*, pages 73–78, Geneva, Switzerland. COLING.

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 *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. 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 *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jingfei Du, Edouard Grave, Beliz Gunel, Vishrav Chaudhary, Onur Celebi, Michael Auli, Veselin Stoyanov, and Alexis Conneau. 2021. Self-training improves pre-training for natural language understanding. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5408–5418, Online. Association for Computational Linguistics.

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*, pages 1307–1323, Online. Association for Computational Linguistics.

Braden Hancock, Paroma Varma, Stephanie Wang, Martin Bringmann, Percy Liang, and Christopher Ré. 2018. Training classifiers with natural language explanations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1884–1895, Melbourne, Australia. Association for Computational Linguistics.

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. 2016. Harnessing deep neural networks with logic rules. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2410–2420, Berlin, Germany. Association for Computational Linguistics.

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*, volume 70 of *Proceedings of Machine Learning Research*, pages 1587–1596. PMLR.

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 *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3193–3207, Online. Association for Computational Linguistics.

Xisen Jin, Zhongyu Wei, Junyi Du, Xiangyang Xue, and Xiang Ren. 2020. Towards hierarchical importance attribution: Explaining compositional semantics for neural sequence models. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Vidur Joshi, Matthew Peters, and Mark Hopkins. 2018. Extending a parser to distant domains using a few dozen partially annotated examples. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1190–1199, Melbourne, Australia. Association for Computational Linguistics.

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*, volume 70 of *Proceedings of Machine Learning Research*, pages 1885–1894. PMLR.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Dong-Ho Lee, Rahul Khanna, Bill Yuchen Lin, Seyeon Lee, Qinyuan Ye, Elizabeth Boschee, Leonardo Neves, and Xiang Ren. 2020. LEAN-LIFE: A label-efficient annotation framework towards learning from explanation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 372–379, Online. Association for Computational Linguistics.

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, 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 *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8503–8511, Online. Association for Computational Linguistics.

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 *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

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 *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5253–5260. AAAI Press.

Tianyu Liu, Jin-Ge Yao, and Chin-Yew Lin. 2019a. Towards improving neural named entity recognition with gazetteers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5301–5307, Florence, Italy. Association for Computational Linguistics.

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 *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409, Austin, Texas. Association for Computational Linguistics.

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*, pages 4231–4245, Online. Association for Computational Linguistics.

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*, pages 8024–8035.

Minlong Peng, Xiaoyu Xing, Qi Zhang, Jinlan Fu, and Xuanjing Huang. 2019. Distantly supervised named entity recognition using positive-unlabeled learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2409–2419, Florence, Italy. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1756–1765, Vancouver, Canada. 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. *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*, pages 1135–1144. ACM.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.

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 *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2054–2064, Brussels, Belgium. Association for Computational Linguistics.

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)*.

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 *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

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 *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

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*, pages 2159–2169, Santa Fe, New Mexico, USA. 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*, pages 1599–1615, Online. Association for Computational Linguistics.

Omar Zaidan and Jason Eisner. 2008. Modeling annotators: A generative approach to learning from annotator rationales. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 31–40, Honolulu, Hawaii. Association for Computational Linguistics.

Wenxuan Zhou, Hongtao Lin, Bill Yuchen Lin, Ziqi Wang, Junyi Du, Leonardo Neves, and Xiang Ren. 2020. NERO: A neural rule grounding framework for label-efficient relation extraction. In *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*, pages 2166–2176. ACM / IW3C2.

A Appendix

A.1 Experimental Settings

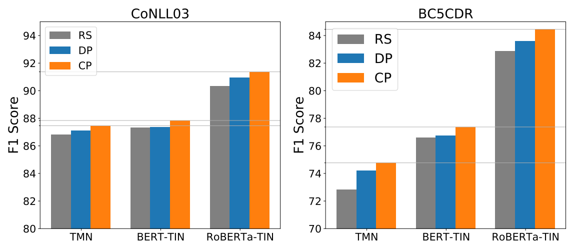
We implement all the baselines using PyTorch (Paszke et al., 2019) and HuggingFace (Wolf et al., 2020). To initialize the word embeddings, we use 100 dimension pre-trained Glove embeddings, cased BERT-base, and RoBERTa-large for each corresponding model. We set the batch size and learning rate to 10 and 0.01 for BLSTM encoder models (i.e., `BLSTM+CRF`, `TMN`, `BERT+BLSTM+CRF`) while we set 30 and 2e-5 for all other transformer models. For our `TIN`, we set the interpolation λ to 0.5. The details are present in Table 4. Also note that for experiments in extreme low resource setting (Sec. 5.2), we set the batch size to 4 for training the models due to the extremely limited training data. For automatic trigger extraction stage, we build the entity token classifier with cased BERT-base encoder for `BERT-TIN` and RoBERTa-large for `RoBERTa-TIN`. 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 the entity token classifier model. 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. `TIN` takes 2X longer than baselines since it needs to extract triggers using SOC algorithm.

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.

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.

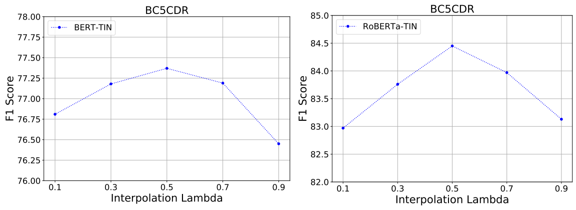


(a) CoNLL03 (b) BC5CDR

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

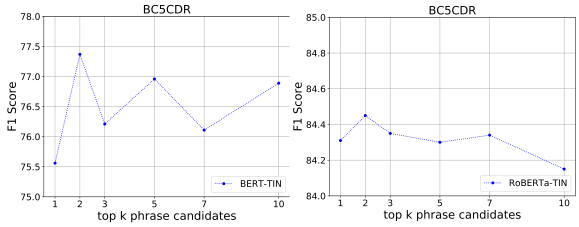
A.4 Performance Analysis

Trigger Candidate Variants. In Sec 3.2, 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. 3.2). 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



(a) BERT-TIN (b) RoBERTa-TIN

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



(a) BERT-TIN (b) RoBERTa-TIN

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

candidates. As we can see, constituency parsing yields consistently better performance by providing better quality of syntactic signals than others.

Sensitivity Analysis of interpolation hyper-parameter (λ). In Sec 3.3, 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.

Number of Triggers. In Sec. 3.2, 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 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) 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 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., 2021). 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.

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.

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).

B Related Work

NER with Additional Supervision Previous and recent research has shown that encoding syntactic information into NER models compensate for the 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) 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 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., 2021).

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 (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).

Encoder	BLSTM	Transformer	
		BERT	RoBERTa
model	`BLSTM+CRF`, `TMN`,	`BERT+CRF`, `BERT-TIN` `BERT+BLSTM+CRF`	`RoBERTa+CRF`, `RoBERTa-TIN`
batch size	10	30	30
learning rate	0.01	2e-5	2e-5
epochs	10	10	10
LSTM hidden dimension	200	-	-

Table 4: Experimental setting details.

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 5: Train data statistics.