AutoTriggER: Named Entity Recognition with Auxiliary Trigger Extraction

Dong-Ho Lee[1,3]*, Ravi Kiran Selvam[1]*, Sheikh Muhammad Sarwar[2], Bill Yuchen Lin[1], Fred Morstatter[3], Jay Pujara[1,3], Elizabeth Boschee[3], James Allan[2], Xiang Ren[1,3]
[1]University of Southern California [2]University of Massachusetts, Amherst [3]Information Science Institute, USC
[1]{dongho.lee, rselvam, yuchen.lin, jpujara, xiangren}@usc.edu [2]{smsarwar, allan}@cs.umass.edu
[3]{fredmors, boschee}@isi.edu

ABSTRACT

While deep neural models for named entity recognition (NER) have shown impressive results, these models often require additional human annotation that is expensive and time-consuming to generate, especially for new or low-resource domains. Some success has been shown replacing conventional human annotation with distant supervision or other meta-level information (e.g. explanations). However, the costs of generating this additional information can still be 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*" for named entity recognition. These triggers—essentially human-readable "clues" in the text that can help guide the model to better decisions—are first identified automatically using a *sampling and occlusion algorithm*. Next, we propose a *trigger interpolation network* to leverage these triggers in a transformer-based NER model. By combining these stages, AutoTriggER 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 standard RoBERTa-CRF architecture by nearly 0.5 F1 points on average and much more in a low resource setting.[1]

ACM Reference Format:
Dong-Ho Lee[1,3]*, Ravi Kiran Selvam[1]*, Sheikh Muhammad Sarwar[2], Bill Yuchen Lin[1],, Fred Morstatter[3], Jay Pujara[1,3], Elizabeth Boschee[3], James Allan[2], Xiang Ren[1,3]. 2020. AutoTriggER: Named Entity Recognition with Auxiliary Trigger Extraction. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3394486.3403153

1 INTRODUCTION

Named Entity Recognition (NER) serves as a key building block in information extraction systems and thus also a necessary step for building knowledge bases such as DBPedia [2] for general topics or COVID19-KG [8] for a vertical domain. Recent advances in deep neural models for NER have yielded state-of-the-art performance when sufficient human annotations are available [19, 25, 28, 36]. However, although strong results have been obtained by these neural NER systems on academic NER benchmarks, 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 [26, 34, 42, 47], labeling rules [15, 40], and labeling explanations [12, 20, 24, 45, 48].

In particular, prior works in label-efficient learning for classification tasks (e.g., relation extraction) [12, 45, 51] and question answering [48] with explanations or human-written rule-like explanation, show that asking humans to provide explanations as auxiliary model supervision is more cost-effective than simply collecting more label-only annotations. As for the task of NER, the concept of an *entity trigger* [24] is 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

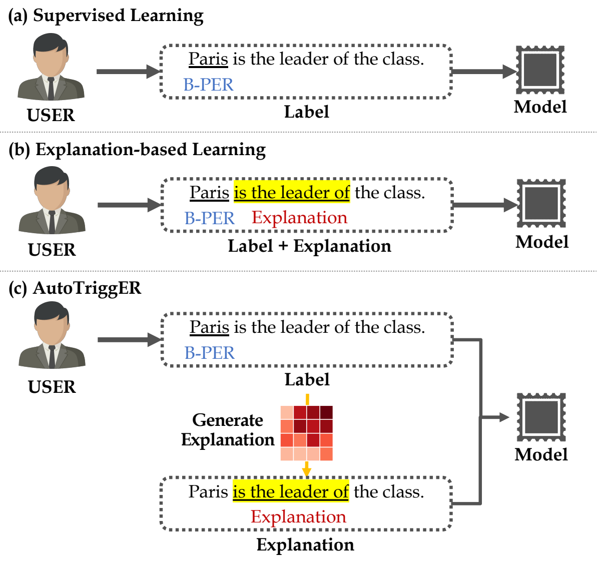

(a) Supervised Learning
USER
Paris is the leader of the class.
B-PER
Label
Model
(b) Explanation-based Learning
USER
Paris is the leader of the class.
B-PER
Explanation
Label + Explanation
Model
(c) AutoTriggER
USER
Paris is the leader of the class.
B-PER
Label
Generate Explanation
Paris is the leader of the class.
Explanation
Explanation
Model

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

[1]Code and data have been uploaded and will be published: https://anonymous.4open.science/r/3b8c759e-fc86-4cc7-b9e8-b6dc252549a1/

ACM ISBN 978-1-4503-7998-4/20/08.
https://doi.org/10.1145/3394486.3403153

would recognize an entity in the sentence and they serve as an effective proxy of rationales, as shown in Figrue 1 (a) vs. (b). For instance, a human could infer that "*Sunnongdan*" is likely to be a restaurant entity in the sentence "*Danny had a dinner at Sunnongdan last week, where the food is delicious.*" — despite never having seen the word *Sunnongdan* before. This is possible because the cue phrases "*had ... dinner at*" and "*where the food*" both suggest there should be a restaurant entity between them. We call such phrases for explaining the labeling rationale as *entity triggers* (or just *triggers* for simplicity).

Prior works on using labeling rules or explanations for improving data (label) efficiency of model training primarily use a limited number of *crowd-soured* triggers. While such human-curated auxiliary supervision are of high quality, the crowd-sourcing procedure can be very expensive and time-consuming, as annotators are asked to provide both entity labels and their associated explanations or rules in specifically designed annotation frameworks. This largely limits the scale and domains of the collected entity triggers. In addition, trigger-aware NER models (e.g., Trigger Matching Networks [24]) are built on conventional sequence tagging architectures, e.g., BLSTM-CRFs [19]. Few works connect entity triggers with contextualized embeddings from pre-trained language models (PTLMs) such as BERT [7], 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 saliency map techniques based on input perturbations. Most saliency map techniques primarily focus on modeling the relative importance of each input token based on its 1) attention intensity [22], 2) gradients [38] or 3) the changes of the output by excluding it from the input [18]. 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. We propose to exploit the syntactic features of sentences for properly 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* [17] 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 context. 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. This general *sampling-and-occlusion* method is thus able to successfully identify high-quality triggers in a broad variety of domains. For example, with our method, "*my preferred candidate*" and "*the next mayor*" are indicated as cue phrases (i.e., entity triggers) that determine "*Cary Moon*" as the person entity in the sentence "My preferred candidate is Cary Moon, but she won't be the next mayor of Seattle" (see Figure 4). Such generated entity triggers not only make trigger-aware models (and their downstream information extraction applications) more trustworthy, but also provide a human-readable, faithful explanation for further debugging in a *human-in-the-loop* way.

The *second* stage (Sec. 3.3) focuses on how to use our triggers as structured priors for better performance and generalization ability of NER, especially in a low-resource setting which is very common in novel domains for NER. We propose *Trigger Interpolation Network* (TIN), a novel architecture that effectively uses trigger-labeled NER data to train a model. 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 (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 exploit the triggers to infer entity boundaries and types with the powerful contextualized embeddings from pre-trained language models such as BERT.

Extensive experimental results on several domains show that the proposed AutoTrigger framework consistently outperforms baseline methods by 0.5 F1 points on average in fully supervised setting. Our work shows the powerful performance especially in low-resource setting for technical domains where domain-expert annotations are very limited due to the high cost. In the extreme low-resource setting, 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* [24] 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 in this paper — 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 $\mathbf{x} = [x^{(1)}, x^{(2)}, \ldots x^{(n)}]$ denote the sentence consisting of a sequence of n words and $\mathbf{y} = [y^{(1)}, y^{(2)}, \ldots y^{(n)}]$ denote the NER-tag sequence. The task of named entity recognition (NER) 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, B-LOC, I-LOC, ..., O}, representing the *B*eginning, *I*nside of entity spans of PER/LOC type and *O*utside of any entity, respectively. 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 the corresponding 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

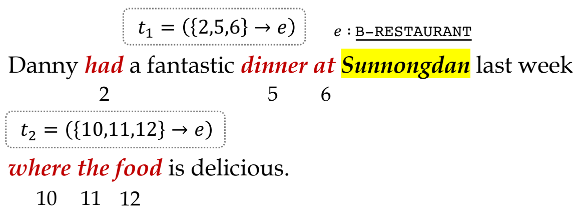

$t_1 = (\{2,5,6\} \rightarrow e)$
e : B-RESTAURANT
Danny had a fantastic dinner at Sunnongdan last week
2 5 6
$t_2 = (\{10,11,12\} \rightarrow e)$
where the food is delicious.
10 11 12

Figure 2: 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.

defined as a group of words that can help explain the recognition process of an entity in the same 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 not only can enable NER models to interpret a particular prediction but also help generalize NER models 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 $\{w_i\}$, where w_i are integers in the range $[1, |\mathbf{x}|]$. That is, $t = (\{w_1, w_2, \dots\} \longrightarrow e)$ represents an entity trigger, e.g., $t_1 = \{2, 5, 6\} \longrightarrow e$ 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 such a $\mathcal{D}_T$ via manually annotating triggers by crowd-source workers. Although trigger-labeled human annotations are more cost-effective than conventional entity-only annotations, they are still expensive and need domain experts for specialized domains, such as biomedical publications, legal reports, financial documents, etc. 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 such that we can use such $\mathcal{D}_T$ to improve NER models in low-resource settings.

3 APPROACH

This section introduces the concepts in AUTOTRIGGER, and provides details of the framework design. We first present an overview of the AUTOTRIGGER framework (Sec. 3.1) and introduce the details of each component (Sec. 3.2- 3.3).

3.1 Framework Overview

We solve the problem using a two-stage architecture called AUTOTRIGGER framework, which consists of *automatic trigger extraction* and a *trigger interpolation network* (TIN). In the first stage, the AUTOTRIGGER automatically extracts entity triggers by computing the importance score of phrases specific to the target entity in the sentence (Sec. 3.2). Then, the second stage learns the NER model with extracted entity triggers (Sec. 3.3). Prior work [24] on incorporating such entity triggers primarily focused on encoding human-provided entity triggers. In contrast, as shown in Figure. 3, our AUTOTRIGGER is automatically generating triggers and directly using them for model learning.

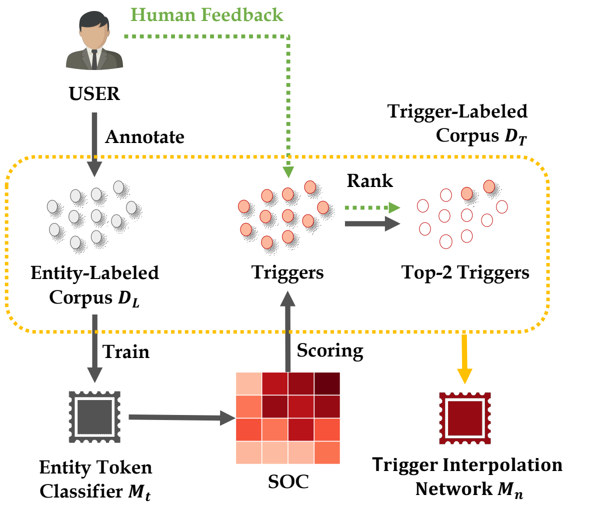

Human Feedback
USER
Annotate
Trigger-Labeled Corpus D_T
Rank
Entity-Labeled Corpus D_L
Triggers
Top-2 Triggers
Train
Scoring
Entity Token Classifier M_t
SOC
Trigger Interpolation Network M_n

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. In this process, the user can also polish triggers with simple annotations. Trigger interpolation network finally learns from the trigger-labeled corpus.

3.2 Automatic Trigger Extraction

Automatic trigger extraction is the first stage of our AUTOTRIGGER framework. To extract triggers, here we use the *sampling and occlusion* (SOC) algorithm [16], which is one of the input analysis techniques for model interpretation. Previous works [22, 38] on input analysis techniques mostly focus on word-level importance. In contrast, *sampling and occlusion* aims to compute context-independent phrase-level importance for sequence classification tasks such as sentiment analysis and relation extraction. We reformulate and apply this technique into a sequence tagging task and retrieve important phrases as entity triggers. To construct a set of phrase candidates to measure the importance, we find that exploiting phrase nodes of the constituency parsing tree can result in better retrieving qualitative entity triggers. However, this method has a limitation that it is difficult to extend to other tasks such as sequence labeling, in which the output is not single but multiple tokens. In this work, we reformulate the method to be able to handle such multiple outputs and apply the module to the sequence labeling task. 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. To compute the importance of each phrase in the sentence, we first need to construct a set of target phrase candidates $\mathcal{P}$. Here, we aim to generate $\mathcal{P}$ using the parse tree. The reason why we first create a set of phrase candidates is to avoid noisy triggers. The original SOC computes the word-level scores and extends to phrases by agglomerative clustering. Since the clustering combines words to form a phrase, the phrase can be incomplete and noisy. By limiting the search space to a set of complete phrases, we could avoid such noisy triggers. For example, in Figure. 4, the sentence $\mathbf{x}$ is parsed into the constituency parse tree. Then, we create a set of phrase candidates with all phrase nodes except the nodes that include entity "*Cary Moon*". Specifically, 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. Additionally, in our work, the target entity itself is not considered an entity trigger. To avoid such case, 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 defined as 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 one of the named entity tags $y_i^{(j)} \in \mathcal{Y}$ where $\mathcal{Y}$ consists of predefined set of named entity tags such as {B-PER, I-PER, B-LOC, 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:

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

Phrase Scoring. We use the phrase candidate $\mathcal{P}$ and prediction score function s of the entity token classifier $\mathcal{M}_t$ 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*.

Input occlusion [22] 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 $0_\mathbf{p}$:

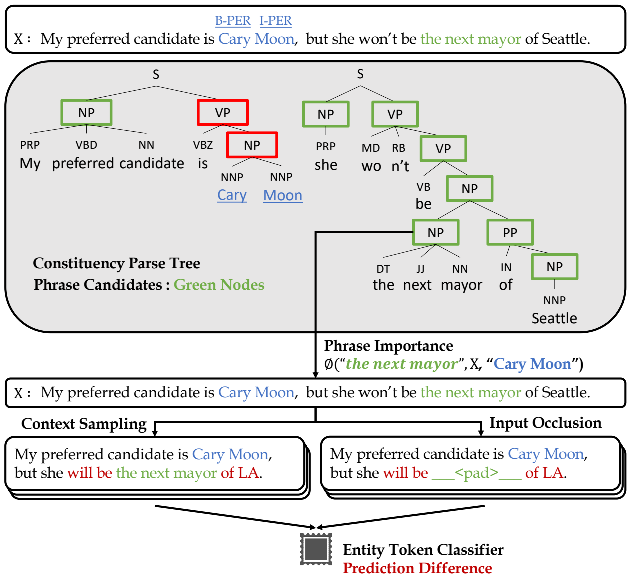
$$\phi(\mathbf{p}, \mathbf{x}, e) = s(\mathbf{x}, e) - s\left(\mathbf{x_{-p}}, e; \mathbf{0_p}\right). \qquad (2)$$

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:

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

For instance, 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



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
My preferred candidate
VBZ
is
NP
NNP NNP
Cary Moon
S
NP
PRP
she
VP
MD RB
wo n't
VP
VB
be
NP
NP
DT JJ NN
the next mayor
PP
IN
of
NP
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 compute the phrase importance by average prediction difference between context sampled sentences and its phrase-masked sentences.

classifier computes the prediction difference between the sampled sentences with and without the phrase "*the next mayor*".

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 $\{(\mathbf{x_i}, \mathbf{y_i}, T(\mathbf{x_i}, \mathbf{y_i}))\}$. Then we construct the trigger-labeled dataset $\mathcal{D}_T$.

3.3 Trigger Interpolation Network (TIN)

The second stage of AutoTrigger is the trigger interpolation network (TIN), which is defined as a neural network for learning from trigger-labeled dataset $\mathcal{D}_T$ consisting of set of instances of the form $\{(\mathbf{x}, \mathbf{y}, T(\mathbf{x}, \mathbf{y}))\}$. Prior work [24] on incorporating $\mathcal{D}_T$ primarily focused on encoding human-provided entity triggers and softly grounding them on unlabeled sentences by conventional attention mechanism. In this work, we argue that a better understanding of how entity triggers affect the predictions of the model can improve the ability to leverage entity triggers. Here, we define the impact of entity triggers by two crucial questions :

- *Are entity words themselves sufficient to learn an entity type?*
- *How entity triggers affect to learn an entity type?*

In this work, we want to strengthen our model's knowledge of triggers while we also want to strengthen the knowledge of entities that do not need the triggers. We address these questions by *mixup* [50], which is performing linear interpolation between random pairs to create virtual training data for data augmentation.

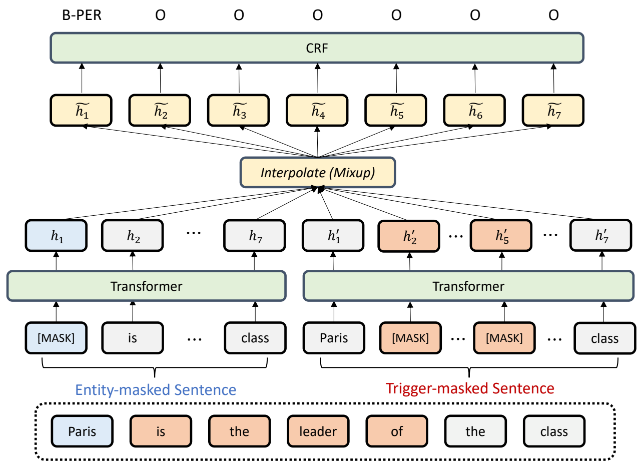


Figure 5: Overview of the Trigger Interpolation Network (TIN). Input sentence is expressed in two different sentences: 1) Entity-masked sentence 2) Trigger-masked Sentence, and then get encoded hidden states h and h'. The network linearly interpolates both into representation $\widetilde{h}$ and forward it to the final CRF tagger.

mixup is originally proposed to train an image classifier on linear interpolations of randomly sampled image data. Recent works generalized this to the textual domain by interpolating output spaces [30] or embedding spaces [3]. For example, let $\mathbf{x_1}$ and $\mathbf{x_2}$ be two text sequences and $\mathbf{y_1}$ and $\mathbf{y_2}$ be their corresponding one-hot label vectors. They apply *mixup* as follows:

$$\tilde{\mathbf{x}} = \lambda \mathbf{x_1} + (1-\lambda)\mathbf{x_2} \\ \tilde{\mathbf{y}} = \lambda \mathbf{y_1} + (1-\lambda)\mathbf{y_2} \tag{4}$$

where $\lambda \in [0,1]$ is a variable sampled from Beta distribution. However, we found that this technique can be used not only for the data augmentation, but also for learning from two different sources.

Here, we focus on linearly interpolating two different information and leverage it to make a prediction. Here, we interpolate the entity masked representation and trigger masked representation to force the model to understand the impact of each representation in predicting the entity type. We construct the architecture following the most common design of neural NER models which exploits CRF layer in the output of the encoder. Specifically, it adopts the transformer model (e.g., BERT, RoBERTa) to encode representations by capturing the semantics of the input text sequence. Then, the output is passed to a CRF layer that produces a probability distribution over the tag sequence using the dependencies among the labels of the entire sequence. To find the best sequence of labels for an input sequence, the Viterbi algorithm is used. TIN encodes the input with the transformer encoder $\mathbf{F}(.;\theta)$ and feeds into the final CRF tagger. Specifically, for a given 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:

$$\mathbf{h} = \mathbf{F}(\mathbf{x}_{-e};\theta), \mathbf{h}' = \mathbf{F}(\mathbf{x}_{-t};\theta) \\ \tilde{\mathbf{h}} = \lambda\mathbf{h} + (1-\lambda)\mathbf{h}'. \tag{5}$$

Dataset	Entity Type	Original $\mathcal{D}_L$	Crowd-sourced trigger $\mathcal{D}_{HT}$	
		# of Entities	# of Entities	# of Human Triggers
CONLL 2003	PER, ORG, MISC, LOC	23,495	5,134	10,938
BC5CDR	Disease, Chemical	9,383	1,991	3,770
JNLPBA	Protein, DNA, RNA Cell Line, Cell Type	46,745	-	-

Table 1: Train data statistics.

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 tag predictions.

4 EXPERIMENTAL SETUP

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

4.1 Dataset

We use three NER datasets in our experiments (see original $\mathcal{D}_L$ in Tab. 1). **BC5CDR** [21] 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** [4] 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** [44] 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. For BC5CDR [21] and CoNLL03 [44], we also have crowd-sourced entity trigger dataset $\mathcal{D}_{HT}$ [24] to compare the quality of our automatically extracted triggers with. For BC5CDR and CoNLL03, 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.

4.2 Compared Methods

To show the effectiveness of entity triggers, we compare models that have same base model but use different training data. Here, we present baseline models that learn $\mathcal{D}_L$ and $\mathcal{D}_T$ respectively. In addition, we present various methods to construct a set of trigger candidates for which the importance scores are computed.

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 [35] 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 [7]. (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 [27] which is a robustly improved BERT.

Entity+Trigger Baseline Models. We apply the following models on $\mathcal{D}_T$: (1) TMN [24] first adopts the structured self-attention layer [23] above the bidirectional LSTM, which uses GloVE to produce token 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. TMN is comparable to BLSTM+CRF and BERT+BLSTM+CRF which use bidirectional LSTM to encode the input sequence. (2) BERT-TIN is our *trigger interpolation network* where the transformer encoder $\mathbf{F}(.;\theta)$ is BERT. BERT-TIN is comparable to BERT+CRF which use BERT to encode the input sequence. (3) RoBERTa-TIN is also our *trigger interpolation network* where the transformer encoder $\mathbf{F}(.;\theta)$ is RoBERTa. RoBERTa-TIN is comparable to RoBERTa+CRF which use BERT to encode the input sequence.

4.3 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.

4.4 Experimental Settings

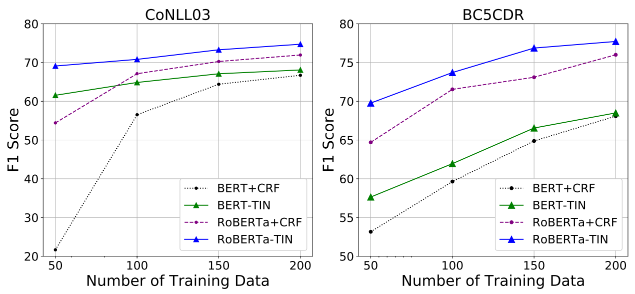
We implement all the baselines using PyTorch [33] and HuggingFace [46]. To initialize the word embeddings, we use 100 dimension pre-trained Glove embeddings, cased BERT-base, and RoBERTa-large for each corresponding model. We set the batch size and learning rate to 10 and 0.01 for BLSTM encoder models (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. We present the details in Appendix A.2.

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 [20], 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 affect our framework. Then, we see the performance of trigger interpolation network by different interpolation weight λ.

5.1 Performance Comparison

In Table 2, we report the performance of the baseline approaches and our model variants on three different datasets. We observe that



(a) CoNLL03 (b) BC5CDR

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

models that receive both entities and triggers as input generally outperform the *entity-only* baselines. TIN model variant 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 assume that the reason for the performance drop might be because of a number of appearances of the MISC entity type in the training data for which the auto-triggers might provide a precision decreasing signal.

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 best baseline for 50 training sentences. For the BC5CDR dataset we observe persistent performance gain. Note that because of the extremely limited training data cases we set the batch size to 4 for training all the models for this experiment.

5.3 Human-in-the-loop Trigger Extraction

Study of 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 (see Table 1). 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>*

Method / Percentage	BC5CDR 20%	40%	60%	80%	100%	JNLPBA 20%	40%	60%	80%	100%	CoNLL03 20%	40%	60%	80%	100%
BLSTM+CRF [19]	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 [24]	**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**

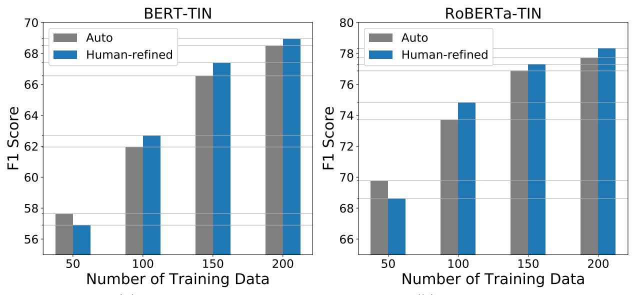
Table 2: Performance comparison (F1-score) of named entity recognition on BC5CDR, JNLPBA, and CoNLL03 datasets by different percentage usage of the train data. For entity+trigger baselines, we use the top 2 candidate phrases from SOC with constituency parsing as triggers. Best models for each encoder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

Sunnongdan.”, while a set of phrase candidates $\mathcal{P}$ from the constituency parse tree can only contain the contiguous token spans. 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 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 [24], we do not rule out the possibility of leveraging human expertise to help.

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). An author of this submission judged relevance of the automatically extracted triggers for entities in 50, 100, 150, and 200 sentences. Figure. 7 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. We leave a full-scale annotation study as future work.

5.4 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 8 shows the model's performance with triggers that have been selected from different sets of phrase candidates. As we can see, constituency



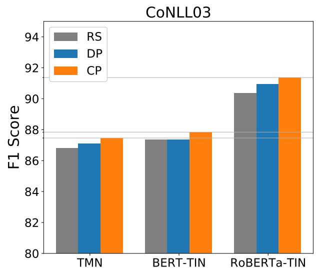
(a) BERT-TIN (b) RoBERTa-TIN

Figure 7: Performance Comparison (F1-score) on BC5CDR by different numbers of train data instances (50, 100, 150, 200) with auto and human-refined auto triggers.

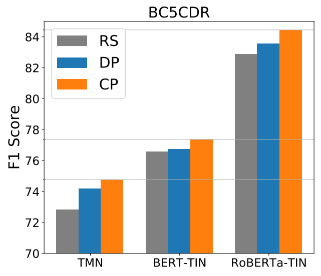
BC5CDR	TMN		BERT-TIN		RoBERTa-TIN	
Percentage / **Model**	human	auto	human	auto	human	auto
5%	**26.96**	24.70	66.20	**66.50**	75.79	**76.92**
10%	**46.24**	43.54	71.25	**71.84**	80.92	**81.63**
15%	**51.29**	50.44	73.88	**74.11**	83.54	**83.87**
20%	**56.28**	54.91	75.97	**76.58**	83.88	**84.17**

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.

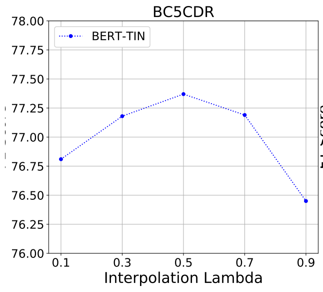


(a) CoNLL03

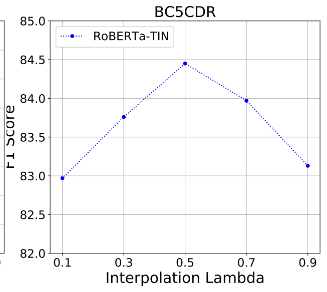


(b) BC5CDR

Figure 8: Performance comparison (F1-score) of entity+trigger baselines on 20% training dataset of CoNLL03 and BC5CDR with different trigger candidate variants.



(a) BERT-TIN



(b) RoBERTa-TIN

Figure 9: Performance comparison (F1-score) of entity+trigger baselines on 20% training dataset of BC5CDR with different interpolation weight λ.

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. 9, 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.

6 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 [43]. 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 [32, 43]. 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.* [43] and Nie *et al.* [32] both showed that key-value memory network (KVMN) [31] 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 [10, 41] as well as limited data scenario [9]. 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 [1, 5], testing model behavior using challenge sets [11, 29, 39], or analyzing an impact of input examples by input perturbations or influence function looking at input examples [18, 38]. 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 [6]. 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 [37] and regularize posterior [13, 14]. 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 [49].

7 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.

ACKNOWLEDGEMENT

This work was supported in part by the Center for Intelligent Information Retrieval and in part by NSF grant #IIS-1617408. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsors.

REFERENCES

[1] Y. Adi, E. Kermany, Y. Belinkov, O. Lavi, and Y. Goldberg. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. In *Proc. of ICLR*, 2017.

[2] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. Dbpedia: A nucleus for a web of open data. In *ISWC/ASWC*, 2007.

[3] Y. Cheng, L. Jiang, W. Macherey, and J. Eisenstein. AdvAug: Robust adversarial augmentation for neural machine translation. In *Proc. of ACL*, 2020.

[4] N. Collier and J.-D. Kim. 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)*, 2004.

[5] A. Conneau, G. Kruszewski, G. Lample, L. Barrault, and M. Baroni. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proc. of ACL*, 2018.

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

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of NAACL-HLT*, 2019.

[8] D. Domingo-Fernández, S. Baksi, B. Schultz, Y. Gadiya, R. Karki, T. Raschka, C. Ebeling, M. Hofmann-Apitius, and A. T. Kodamullil. Covid-19 knowledge graph: a computable, multi-modal, cause-and-effect knowledge model of covid-19 pathophysiology. *bioRxiv*, 2020.

[9] J. Du, E. Grave, B. Gunel, V. Chaudhary, O. Celebi, M. Auli, V. Stoyanov, and A. Conneau. Self-training improves pre-training for natural language understanding. *arXiv preprint arXiv:2010.02194*, 2020.

[10] J. Foley, S. M. Sarwar, and J. Allan. Named entity recognition with extremely limited data. *arXiv preprint arXiv:1806.04411*, 2018.

[11] M. Gardner, Y. Artzi, V. Basmov, J. Berant, B. Bogin, S. Chen, P. Dasigi, D. Dua, Y. Elazar, A. Gottumukkala, N. Gupta, H. Hajishirzi, G. Ilharco, D. Khashabi, K. Lin, J. Liu, N. F. Liu, P. Mulcaire, Q. Ning, S. Singh, N. A. Smith, S. Subramanian, R. Tsarfaty, E. Wallace, A. Zhang, and B. Zhou. Evaluating models' local decision boundaries via contrast sets. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020.

[12] B. Hancock, P. Varma, S. Wang, M. Bringmann, P. Liang, and C. Ré. Training classifiers with natural language explanations. In *Proc. of ACL*, 2018.

[13] Z. Hu, X. Ma, Z. Liu, E. Hovy, and E. Xing. Harnessing deep neural networks with logic rules. In *Proc. of ACL*, 2016.

[14] Z. Hu, Z. Yang, X. Liang, R. Salakhutdinov, and E. P. Xing. 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, 2017.

[15] C. Jiang, Y. Zhao, S. Chu, L. Shen, and K. Tu. Cold-start and interpretability: Turning regular expressions into trainable recurrent neural networks. In *Proc. of EMNLP*, 2020.

[16] X. Jin, Z. Wei, J. Du, X. Xue, and X. Ren. Towards hierarchical importance attribution: Explaining compositional semantics for neural sequence models. In *Proc. of ICLR*, 2020.

[17] V. Joshi, M. Peters, and M. Hopkins. Extending a parser to distant domains using a few dozen partially annotated examples. In *Proc. of ACL*, 2018.

[18] P. W. Koh and P. Liang. 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, 2017.

[19] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer. Neural architectures for named entity recognition. In *Proc. of NAACL-HLT*, 2016.

[20] D.-H. Lee, R. Khanna, B. Y. Lin, S. Lee, Q. Ye, E. Boschee, L. Neves, and X. Ren. LEAN-LIFE: A label-efficient annotation framework towards learning from explanation. In *Proc. of ACL*, 2020.

[21] J. Li, Y. Sun, R. J. Johnson, D. Sciaky, C.-H. Wei, R. Leaman, A. P. Davis, C. J. Mattingly, T. C. Wiegers, and Z. Lu. Biocreative v cdr task corpus: a resource for chemical disease relation extraction. *Database : the journal of biological databases and curation*, 2016.

[22] J. Li, W. Monroe, and D. Jurafsky. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*, 2016.

[23] Z. Lin, M. Feng, C. N. dos Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio. A structured self-attentive sentence embedding. In *Proc. of ICLR*, 2017.

[24] B. Y. Lin, D.-H. Lee, M. Shen, R. Moreno, X. Huang, P. Shiralkar, and X. Ren. TriggerNER: Learning with entity triggers as explanations for named entity recognition. In *Proc. of ACL*, 2020.

[25] L. Liu, J. Shang, X. Ren, F. F. Xu, H. Gui, J. Peng, and J. Han. Empower sequence labeling with task-aware neural language model. In *Proc. of AAAI*, 2018.

[26] T. Liu, J.-G. Yao, and C.-Y. Lin. Towards improving neural named entity recognition with gazetteers. In *Proc. of ACL*, 2019.

[27] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[28] X. Ma and E. Hovy. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proc. of ACL*, 2016.

[29] T. McCoy, E. Pavlick, and T. Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proc. of ACL*, 2019.

[30] Z. Miao, Y. Li, X. Wang, and W. Tan. Snippext: Semi-supervised opinion mining with augmented data. In *Proc. of WWW*, 2020.

[31] A. Miller, A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, and J. Weston. Key-value memory networks for directly reading documents. In *Proc. of EMNLP*, 2016.

[32] Y. Nie, Y. Tian, Y. Song, X. Ao, and X. Wan. Improving named entity recognition with attentive ensemble of syntactic information. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020.

[33] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. 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*, 2019.

[34] M. Peng, X. Xing, Q. Zhang, J. Fu, and X. Huang. Distantly supervised named entity recognition using positive-unlabeled learning. In *Proc. of ACL*, 2019.

[35] J. Pennington, R. Socher, and C. Manning. GloVe: Global vectors for word representation. In *Proc. of EMNLP*, 2014.

[36] M. Peters, W. Ammar, C. Bhagavatula, and R. Power. Semi-supervised sequence tagging with bidirectional language models. In *Proc. of ACL*, 2017.

[37] A. Ratner, S. H. Bach, H. R. Ehrenberg, J. A. Fries, S. Wu, and C. Ré. Snorkel: Rapid training data creation with weak supervision. *VLDB*, 2017.

[38] M. T. Ribeiro, S. Singh, and C. Guestrin. "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*, 2016.

[39] M. T. Ribeiro, T. Wu, C. Guestrin, and S. Singh. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proc. of ACL*, 2020.

[40] E. Safranchik, S. Luo, and S. H. Bach. Weakly supervised sequence tagging from noisy rules. In *AAAI*, 2020.

[41] S. M. Sarwar, J. Foley, and J. Allan. Term relevance feedback for contextual named entity retrieval. In *CHIIR*, 2018.

[42] J. Shang, L. Liu, X. Gu, X. Ren, T. Ren, and J. Han. Learning named entity tagger using domain-specific dictionary. In *Proc. of EMNLP*, 2018.

[43] Y. Tian, W. Shen, Y. Song, F. Xia, M. He, and K. Li. Improving biomedical named entity recognition with syntactic information. *BMC Bioinformatics*, 2020.

[44] E. F. Tjong Kim Sang. 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)*, 2002.

[45] Z. Wang, Y. Qin, W. Zhou, J. Yan, Q. Ye, L. Neves, Z. Liu, and X. Ren. Learning from explanations with neural execution tree. In *Proc. of ICLR*, 2020.

[46] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. Rush. Transformers: State-of-the-art natural language processing. In *Proc. of EMNLP*, 2020.

[47] Y. Yang, W. Chen, Z. Li, Z. He, and M. Zhang. Distantly supervised NER with partial annotation learning and reinforcement learning. In *Proceedings of the 27th International Conference on Computational Linguistics*, 2018.

[48] Q. Ye, X. Huang, E. Boschee, and X. Ren. Teaching machine comprehension with compositional explanations. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020.

[49] O. Zaidan and J. Eisner. Modeling annotators: A generative approach to learning from annotator rationales. In *Proc. of EMNLP*, 2008.

[50] H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. In *Proc. of ICLR*, 2018.

[51] W. Zhou, H. Lin, B. Y. Lin, Z. Wang, J. Du, L. Neves, and X. Ren. NERO: A neural rule grounding framework for label-efficient relation extraction. In *Proc. of WWW*, 2020.

A APPENDIX

A.1 Data Statistics

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 4: Train data statistics.

A.2 Experimental Settings

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 5: Experimental setting details.