TriggerNER: Learning with Entity Triggers as Explanations for Named Entity Recognition

Bill Yuchen Lin[†*] **Dong-Ho Lee**[†*] **Ming Shen**[†] **Ryan Moreno**[†]
Xiao Huang[†] **Prashant Shiralkar**[‡] **Xiang Ren**[†]
{yuchen.lin,dongho.lee,shenming,morenor}@usc.edu
huan183@usc.edu, shiralp@amazon.com, xiangren@usc.edu
[†]University of Southern California [‡] Amazon

Abstract

Training neural models for named entity recognition (NER) in a new domain often requires additional human annotations (e.g., tens of thousands of labeled instances) that are usually expensive and time-consuming to collect. Thus, a crucial research question is how to obtain supervision in a cost-effective way. In this paper, we introduce "entity triggers," an effective proxy of human explanations for facilitating label-efficient learning of NER models. An entity trigger is defined as a group of words in a sentence that helps to explain why humans would recognize an entity in the sentence.

We crowd-sourced 14k entity triggers for two well-studied NER datasets[1]. Our proposed model, *Trigger Matching Network*, jointly learns trigger representations and soft matching module with self-attention such that can generalize to unseen sentences easily for tagging. Our framework is significantly more cost-effective than the traditional neural NER frameworks. Experiments show that using only 20% of the trigger-annotated sentences results in a comparable performance as using 70% of conventional annotated sentences.

1 Introduction

Named entity recognition (NER) is a fundamental information extraction task that focuses on extracting entities from a given text and classifying them using pre-defined categories (e.g., persons, locations, organizations) (Nadeau and Sekine, 2007). Recent advances in NER have primarily focused on training neural network models with an abundance of human annotations, yielding state-of-the-art results (Lample et al., 2016). However, collecting human annotations for NER is expensive and time-consuming, especially in social media messages (Lin et al., 2017a) and technical domains such as biomedical publications, financial documents, legal reports, etc. As we seek to advance NER into more domains with less human effort, how to learn neural models for NER in a cost-effective way becomes a crucial research problem.

The standard protocol for obtaining an annotated NER dataset involves an annotator selecting token spans in a sentence as mentions of entities, and labeling them with an entity type. However, such annotation process provides limited supervision *per example*. Consequently, one would need large amount of annotations in order to train high-performing models for a broad range of entity types, which can clearly be cost-prohibitive. The key question is then *how can we learn an effective NER model in presence of limited quantities of labeled data*?

We, as humans, recognize an entity within a sentence based on certain words or phrases that act as cues. For instance, we could infer that *'Kasdfrcxzv'* is likely to be a location entity in the sentence "*Tom traveled a lot last year in Kasdfrcxzv.*" We recognize this entity because of the cue phrase "*travel ... in*," which suggests there should be a location entity following the word 'in'. We call such phrases "entity triggers." Similar to the way these triggers guide our recognition process, we hypothesize that they can also help the model to learn to generalize efficiently.

Specifically, we define an ***"entity trigger"*** (or trigger for simplicity) as a group of words that can help explain the recognition process of a particular entity in the same sentence. For example, in Figure 1, *"had ... lunch at"*[2] and *"where the food"* are two distinct *triggers* associated with the RESTAURANT entity *"Rumble Fish."* An entity

[*]The first two authors contributed equally.

[1]We release the entity triggers and code at http://github.com/INK-USC/TriggerNER

[2]Note that a trigger can be a discontinuous phrase.

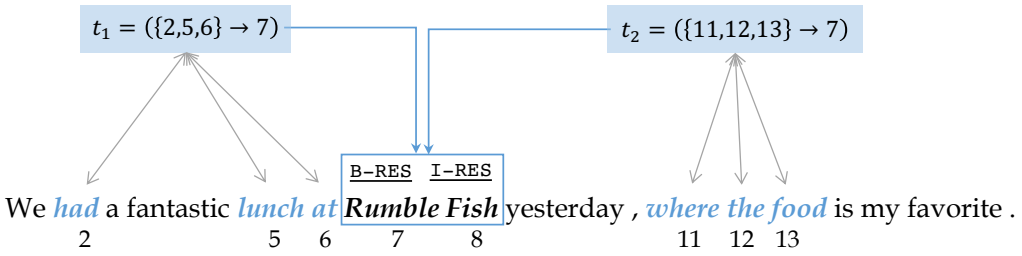

Figure 1: We show two individual **entity triggers**: t_1 ("*had ... lunch at*") and t_2 ("*where the food*"). Both are associated to the same entity mention *"Rumble Fish"* (starting from 7th token) typed as restaurant (RES).

trigger should be a necessary and sufficient cue for humans to recognize its associated entity even if we mask the entity with a random word. Thus, unnecessary words such as "*fantastic*" should not be considered part of the entity trigger.

In this paper, we argue that a combination of entity triggers and standard entity annotations can enhance the generalization power of NER models. This approach is more powerful because unlabeled sentences, such as *"Bill enjoyed a great dinner with Alice at Zcxlbz."*, can be matched with the existing trigger *"had ... lunch at"* via their semantic relatedness. This makes it easier for a model to recognize *"Zcxlbz"* as a RESTAURANT entity. In contrast, if we only have the entity annotation itself (i.e., *"Rumble Fish"*) as supervision, the model will require many similar examples in order to learn this simple pattern. Annotation of triggers, in addition to entities, does not incur significantly additional effort because the triggers are typically short, and more importantly, the annotator has already comprehended the sentence, identifying their entities as required in the traditional annotation. On the benchmark datasets we consider, the average length of a trigger in our crowdsourced dataset is only 1.5-2 words. Thus, we hypothesize that using triggers as additional supervision is a more cost-effective way to train models.

We crowd-sourced annotations of 14,708 triggers on two well-studied NER datasets to study their usefulness for the NER task. We propose a novel framework named Trigger Matching Network that learns trigger representations indicative of entity types during the training phase, and identifies triggers in an unlabeled sentence at inference time to guide a traditional entity tagger for delivering better overall NER performance. Our TMN framework consists of three components: 1) a trigger encoder to learn meaningful trigger representations for an entity type, 2) a semantic trigger matching module for identifying triggers in a new sentence, and 3) an entity tagger that leverages trigger representations for entity recognition (as present in existing NER frameworks). Different from conventional training, our learning process consists of two stages, in which the first stage comprises jointly training a trigger classifier and the semantic trigger matcher, followed by a second stage that leverages the trigger representation and the encoding of the given sentence using an attention mechanism to learn a sequence tagger.

Our contributions in this paper are as follows:

- We introduce the concept of "entity triggers," a novel form of explanatory annotation for named entity recognition problems. We crowd-source and publicly release 14k annotated entity triggers on two popular datasets: *CoNLL03* (generic domain), *BC5CDR* (biomedical domain).
- We propose a novel learning framework, named *Trigger Matching Network*, which encodes entity triggers and softly grounds them on unlabeled sentences to increase the effectiveness of the base entity tagger (Section 3).
- Experimental results (Section 4) show that the proposed trigger-based framework is significantly more cost-effective. The TMN uses 20% of the trigger-annotated sentences from the original CoNLL03 dataset, while achieving a comparable performance to the conventional model using 70% of the annotated sentences.

2 Problem Formulation

We consider the problem of *how to cost-effectively learn a model for NER using entity triggers*. In this section, we introduce basic concepts and their notations, present the conventional data annotation process for NER, and provide a formal task definition for learning using entity triggers.

In the conventional setup for supervised learning for NER, we let $\mathbf{x} = [x^{(1)}, x^{(2)}, \cdots, x^{(n)}]$ de-

note a sentence in the labeled training corpus $\mathcal{D}_L$. Each labeled sentence has a NER-tag sequence $\mathbf{y} = [y^{(1)}, y^{(2)}, \cdots, y^{(n)}]$, where $y^{(i)} \in \mathcal{Y}$ and $\mathcal{Y}$ can be {O, B-PER, I-PER, B-LOC, I-LOC, $\cdots$}. The possible tags come from a BIO or BIOES tagging schema for segmenting and typing entity tokens. Thus, we have $\mathcal{D}_L = \{(\mathbf{x_i}, \mathbf{y_i})\}$, and an unlabeled corpus $\mathcal{D}_U = \{\mathbf{x_i}\}$.

We propose to annotate entity triggers in sentences. We use $T(\mathbf{x}, \mathbf{y})$ to represent the set of annotated entity triggers, where each trigger $t_i \in T(\mathbf{x}, \mathbf{y})$ is associated with an entity index e and a set of word indices $\{w_i\}$. Note that we use the index of the first word of an entity as its entity index. That is, $t = (\{w_1, w_2, \cdots\} \rightarrow e)$, where e and w_i are integers in the range of $[1, |\mathbf{x}|]$. For instance, in the example shown in Figure 1, the trigger *"had ... lunch at"* can be represented as a trigger $t_1 = (\{2, 5, 6\} \rightarrow 7)$, because this trigger specifies the entity starting at index 7, *"Rumble"*, and it contains a set of words with indices: *"had"* (2), *"lunch"* (5), and *"at"* (6). Similarly, we can represent the second trigger *"where the food"* as $t_2 = (\{11, 12, 13\} \rightarrow 7)$. Thus, we have $T(\mathbf{x}, \mathbf{y}) = \{t_1, t_2\}$ for this sentence.

Adding triggers creates a new form of data $\mathcal{D}_T = \{(\mathbf{x_i}, \mathbf{y_i}, T(\mathbf{x_i}, \mathbf{y_i})\}$. Our goal is to learn a model for NER from a trigger-labeled dataset $\mathcal{D}_T$, such that we can achieve comparable learning performance to a model with a much larger $\mathcal{D}_L$.

3 Trigger Matching Networks

We now present our framework for a more cost-effective learning method for NER using triggers. We assume that we have collected entity triggers (the trigger collection process is discussed in Section 4.1). At a high-level, we aim to learn trigger representations for entity types that allow the entity tagger to generalize for sentences beyond the training phase. Our intuition is that triggers acting as cues for the same named-entity type should have similar trigger representations, and thus triggers can be identified in an unlabeled sentence at inference time by soft-matching between the sentence representation and trigger representations seen during training. We perform such soft-matching using a self-attention mechanism.

We propose a straightforward yet effective framework, named *Trigger Matching Networks* (TMN), consisting of a trigger encoder (`TrigEncoder`), a semantic-based trigger matching module (`TrigMatcher`), and a base sequence tagger (`SeqTagger`). We have two learning stages for the framework: the first stage (Section 3.1) jointly learns the `TrigEncoder` and `TrigMatcher`, and the second stage (Section 3.2) uses the trigger vectors to learn NER tag labels. Figure 2 shows this pipeline. We introduce the inference in Section 3.3.

3.1 Trigger Encoding & Semantic Matching

Learning trigger representations and semantically matching them with sentences are inseparable tasks. Desired trigger vectors capture the semantics in a shared embedding space with token hidden states, such that sentences and triggers can be semantically matched. Recall the example we discussed in Sec. 1, *"enjoyed a great dinner at"* versus *"had ... lunch at."* Learning an attention-based matching module between entity triggers and sentences is necessary so that triggers and sentences can be semantically matched. Therefore, in the first stage, we propose to jointly train the trigger encoder (`TrigEncoder`) and the attention-based trigger matching module (`TrigMatcher`) using a shared embedding space.

Specifically, for a sentence $\mathbf{x}$ with multiple entities $\{e_1, e_2, \cdots\}$, for each entity e_i we assume that there is a set of triggers $T_i = \{t_1^{(i)}, t_2^{(i)}, \cdots\}$ without loss of generality. To enable more efficient batch-based training, we reformat the trigger-based annotated dataset $\mathcal{D}_T$ such that each new sequence contains only one entity and one trigger. We then create a training instance by pairing each entity with one of its triggers, denoted $(\mathbf{x}, e_i, t_j^{(i)})$.

For each reformed training instance $(\mathbf{x}, e, t)$, we first apply a bidirectional LSTM (BLSTM) on the sequence of word vectors[3] of $\mathbf{x}$, obtaining a sequence of hidden states that are the contextualized word representations $\mathbf{h}_i$ for each token x_i in the sentence. We use $\mathbf{H}$ to denote the matrix containing the hidden vectors of all of the tokens, and we use $\mathbf{Z}$ to denote the matrix containing the hidden vectors of all trigger tokens inside the trigger t.

In order to learn an attention-based representation of both triggers and sentences, we follow the self-attention method introduced by (Lin et al.,

[3]Here, by "word vectors" we mean the concatenation of external GloVe (Pennington et al., 2014) word embeddings and char-level word representations from a trainable CNN network (Ma and Hovy, 2016).

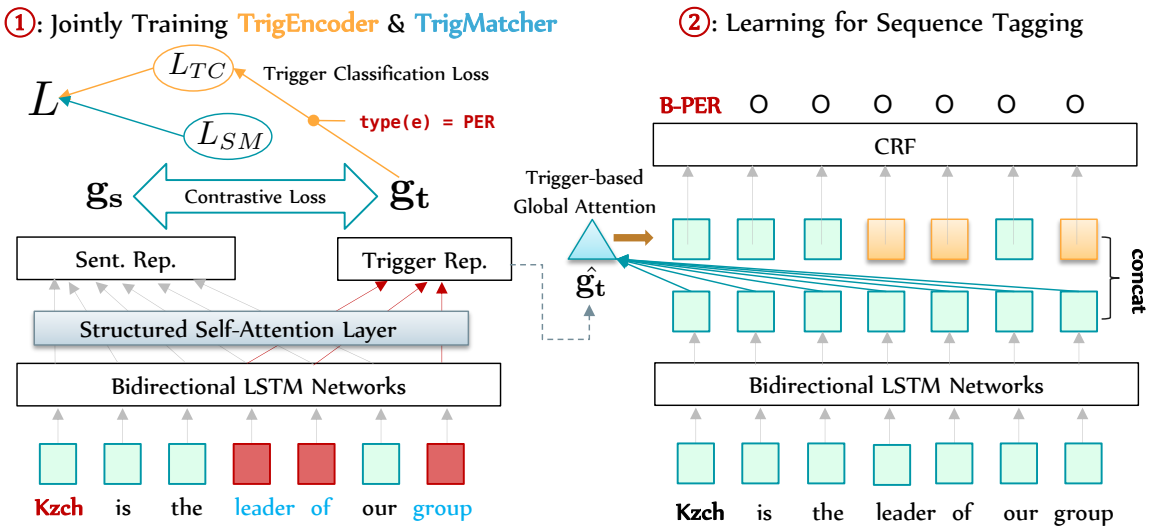

Figure 2: **Two-stage training of the *Trigger Matching Network*.** We first jointly train the `TrigEncoder` (via trigger classification) and the `TrigMatcher` (via contrastive loss). Then, we reuse the training data trigger vectors as attention queries in the `SeqTagger`.

2017b) as follows:

$$\vec{a}_{sent} = \text{SoftMax}\left(W_2 \tanh\left(W_1 \mathbf{H}^T\right)\right)$$
$$\mathbf{g_s} = \vec{a}_{sent}\mathbf{H}$$
$$\vec{a}_{trig} = \text{SoftMax}\left(W_2 \tanh\left(W_1 \mathbf{Z}^T\right)\right)$$
$$\mathbf{g_t} = \vec{a}_{trig}\mathbf{Z}$$

W_1 and W_2 are two trainable parameters for computing self-attention score vectors $\vec{a}_{sent}$ and $\vec{a}_{trig}$. We obtain a vector representing the weighted sum of the token vectors in the entire sentence as the final sentence vector $\mathbf{g_s}$. Similarly, $\mathbf{g_t}$ is the final trigger vector, representing the weighted sum of the token vectors in the trigger.

We want to use the type of the associated entity as supervision to guide the trigger representation. Thus, the trigger vector $\mathbf{g_t}$ is further fed into a multi-class classifier to predict the *type* of the associated entity e (such as `PER`, `LOC`, etc) which we use $\texttt{type}(e)$ to denote. The loss of the trigger classification is as follows:

$$L_{TC} = -\sum \log \text{P}\left(\texttt{type}(e) \mid \mathbf{g_t}; \theta_{TC}\right),$$

where θ_{TC} is a model parameter to learn.

Towards learning to match triggers and sentences based on attention-based representations, we use contrastive loss (Hadsell et al., 2006). The intuition is that similar triggers and sentences should have close representations (i.e., have a small distance between them, d). We create negative examples (i.e., mismatches) for training by randomly mixing the triggers and sentences, because `TrigMatcher` needs to be trained with both positive and negative examples of the form (sentence, trigger, label). For the negative examples, we expect a margin m between their embeddings. The contrastive loss of the soft matching is defined as follows, where $\mathbb{1}_{\text{matched}}$ is 1 if the trigger was originally in this sentence and 0 if they are not:

$$d = \|\mathbf{g_s} - \mathbf{g_t}\|_2$$
$$L_{SM} = (1 - \mathbb{1}_{\text{matched}})\frac{1}{2}(d)^2$$
$$+\mathbb{1}_{\text{matched}}\frac{1}{2}\{\max(0, m - d)\}^2$$

The joint loss of the first stage is thus $L = L_{TC} + \lambda L_{SM}$, where λ is a hyper-parameter to tune.

3.2 Trigger-Enhanced Sequence Tagging

The learning objective in this stage is to output the tag sequence $\mathbf{y}$. Following the most common design of neural NER architecture, BLSTM-CRF (Ma and Hovy, 2016), we incorporate the entity triggers as attention queries to train a trigger-enhanced sequence tagger for NER. Note that the BLSTM used in the the `TrigEncoder` and `TrigMatcher` modules is the same BLSTM we use in the `SeqTagger` to obtain $\mathbf{H}$, the matrix containing the hidden vectors of all of the tokens. Given a sentence $\mathbf{x}$, we use the previously trained `TrigMatcher` to compute the mean of all the trigger vectors $\hat{\mathbf{g}}_\mathbf{t}$ associated with this sentence.

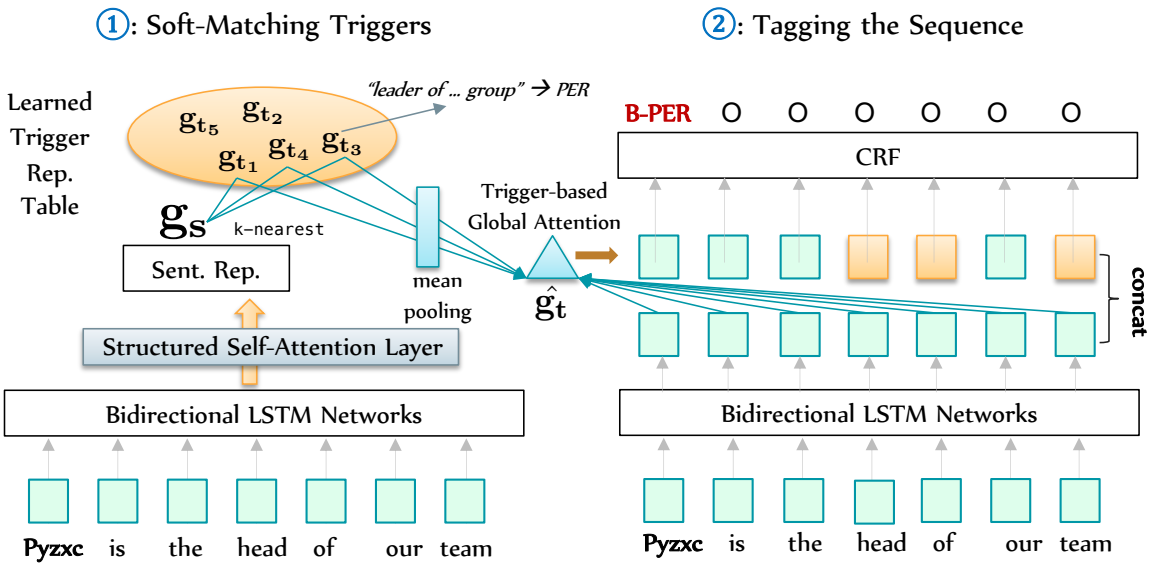

Figure 3: **The inference process of the TMN framework.** It uses the `TrigMatcher` to retrieve the k nearest triggers and average their trigger vectors as the attention query for the trained `SeqTagger`. Thus, an unseen cue phrase (e.g., *"head of ... team"*) can be matched with a seen trigger (e.g., *"leader of ... group"*).

Following the conventional attention method (Luong et al., 2015), we incorporate the mean trigger vector as the query, creating a sequence of attention-based token representations, $\mathbf{H}'$.

$$\vec{\alpha} = \text{SoftMax}\left(\boldsymbol{v}^\top \tanh\left(U_1\mathbf{H}^T + U_2\hat{\mathbf{g}_t}^T\right)^\top\right)$$

$$\mathbf{H}' = \vec{\alpha}\,\mathbf{H}$$

U_1, U_2, and v are trainable parameters for computing the trigger-enhanced attention scores for each token. Finally, we concatenate the original token representation $\mathbf{H}$ with the trigger-enhanced one $\mathbf{H}'$ as the input ($[\mathbf{H}; \mathbf{H}']$) to the final CRF tagger. Note that in this stage, our learning objective is the same as conventional NER, which is to correctly predict the tag for each token.

3.3 Inference on Unlabeled Sentences

When inferencing tags on unlabeled sentences, we do not know the sentence's triggers. Instead, we use the `TrigMatcher` to compute the similarities between the self-attended sentence representations and the trigger representations, using the most suitable triggers as additional inputs to the `SeqTagger`. Specifically, we have a trigger dictionary from our training data, $\mathcal{T} = \{t|(\cdot,\cdot,t) \in \mathcal{D}_T\}$. Recall that we have learned a trigger vector for each of them, and we can load these trigger vectors as a look-up table in memory. For each unlabeled sentence $\mathbf{x}$, we first compute its self-attended vector $\mathbf{g_s}$ as we do when training the `TrigMatcher`. Using L2-norm distances to compute the contrastive loss, we efficiently retrieve the most similar triggers in the shared embedding space of the sentence and trigger vectors.

Then, we calculate $\hat{\mathbf{g}_t}$, the mean of the top k nearest semantically matched triggers, as this serves a proxy to triggers mentioned for the entity type in the labeled data. We then use it as the attention query for `SeqTagger`, similarly in Sec. 3.2. Now, we can produce trigger-enhanced sequence predictions on unlabeled data, as shown in Fig. 3.

4 Experiments

In this section, we first discuss how to collect entity triggers, and empirically study the data-efficiency of our proposed framework.

4.1 Annotating Entity Triggers as Explanatory Supervision

We use a general domain dataset CoNLL2003 (Tjong Kim Sang and De Meulder, 2003) and a bio-medical domain dataset BC5CDR (Li et al., 2016). Both datasets are well-studied and popular in evaluating the performance of neural named entity recognition models such as BLSTM-CRF (Ma and Hovy, 2016).

In order to collect the entity triggers from human annotators, we use *Amazon SageMaker Ground Truth*[4] to crowd-source entity triggers.

[4] An advanced version of *Amazon Mechanical Turk*. https://aws.amazon.com/sagemaker/

Dataset	Entity Type	# of Entities	# of Triggers	Avg. # of Triggers per Entity	Avg. Trigger Length
CONLL 2003	PER	1,608	3,445	2.14	1.41
	ORG	958	1,970	2.05	1.46
	MISC	787	2,057	2.61	1.4
	LOC	1,781	3,456	1.94	1.44
	Total	5,134	10,938	2.13	1.43
BC5CDR	DISEASE	906	2,130	2.35	2.00
	CHEMICAL	1,085	1,640	1.51	1.99
	Total	1,991	3,770	1.89	2.00

Table 1: **Statistics of the crowd-sourced entity triggers**.

More recently, Lee et al. (2020) developed an annotation framework, named LEAN-LIFE, which supports our proposed trigger annotating. Specifically, we sample 20% of each training set as our inputs, and then reform them to be the same format as we discussed in Section 2. Annotators are asked to annotate a group of words that would be helpful in typing and/or detecting the occurrence of a particular entity in the sentence. We masked the entity tokens with their types so that human annotators are more focused on the non-entity words in the sentence when considering the triggers. We consolidate multiple triggers for each entity by taking the intersection of the three annotators' results. Statistics of the final curated triggers are summarized in Table 1. We release the 14k triggers to the community for future research in trigger-enhanced NER.

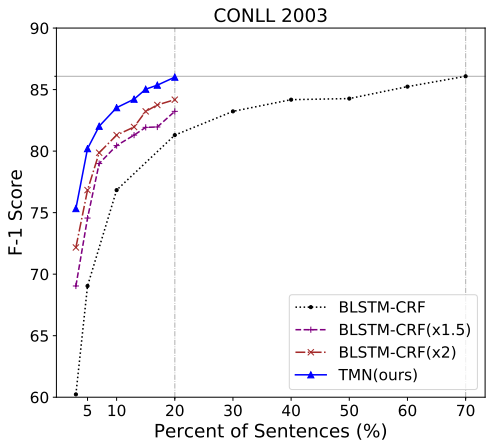

Figure 4: **The cost-effectiveness study.** We stretch the curve of BLSTM-CRF parallel to the x-axis by 1.5/2. Even if we assume annotating entity triggers cost 150/200% the amount of human effort as annotating entities only, TMN is still much more effective.

4.2 Base model

We require a base model to compare with our proposed TMN model in order to validate whether the TMN model effectively uses triggers to improve model performance in a limited label setting. We choose the CNN-BLSTM-CRF (Ma and Hovy, 2016) as our base model for its wide usage in research of neural NER models and applications. Our TMNs are implemented within the same codebase and use the same external word vectors from GloVE (Pennington et al., 2014). The hyper-parameters of the CNNs, BLSTMs, and CRFs are also the same. This ensures a fair comparison between a typical non-trigger NER model and our trigger-enhanced framework.

4.3 Results and analysis

Labeled data efficiency. We first seek to study the cost-effectiveness of using triggers as an additional source of supervision. Accordingly, we explore the performance of our model and the baseline for different fractions of the training data. The results on the two datasets are shown in Table 2. We can see that by using only 20% of the trigger-annotated data, TMN model delivers comparable performance as the baseline model using 50-70% traditional training data. The drastic improvement in the model performance obtained using triggers thus justifies the slightly additional cost incurred in annotating triggers.

Self-training with triggers. We also do a preliminary investigation of adopting self-training (Rosenberg et al., 2005) with triggers. We make inferences on unlabeled data and take the predictions with high confidences as the weak training examples for continually training the model. The confidence is computed following the MNLP metric (Shen et al., 2017), and we take top 20% every epoch. With the self-training method, we further improve the TMN model's F-1 scores by about 0.5∼1.0%.

CONLL 2003										
	BLSTM-CRF				TMN			TMN + SELF-TRAINING		
sent.	Precision	Recall	F1	**trig.**	Precision	Recall	F1	Precision	Recall	F1
5%	70.85	67.32	69.04	3%	76.36	74.33	75.33	80.36	75.18	77.68
10%	76.57	77.09	76.83	5%	81.28	79.16	80.2	81.96	81.18	81.57
20%	82.17	80.35	81.3	7%	82.93	81.13	82.02	82.92	81.94	82.43
30%	83.71	82.76	83.23	10%	84.47	82.61	83.53	84.47	82.61	83.53
40%	85.31	83.1	84.18	13%	84.76	83.69	84.22	84.64	84.01	84.33
50%	85.07	83.49	84.27	15%	85.61	84.45	85.03	86.53	84.26	85.38
60%	85.58	84.54	85.24	17%	85.25	85.46	85.36	86.42	84.63	85.52
70%	86.87	85.3	**86.08**	**20%**	86.04	85.98	**86.01**	87.09	85.91	**86.5**
BC5CDR										
	BLSTM-CRF				TMN			TMN + SELF-TRAINING		
sent.	Precision	Recall	F1	**trig.**	Precision	Recall	F1	Precision	Recall	F1
5%	63.37	43.23	51.39	3%	66.47	57.11	61.44	65.23	59.18	62.06
10%	68.83	60.37	64.32	5%	69.17	73.31	66.11	68.02	66.76	67.38
20%	79.09	62.66	69.92	7%	64.81	69.82	67.22	69.87	66.03	67.9
30%	80.13	65.3	71.87	10%	71.89	69.57	70.71	69.75	72.75	71.22
40%	82.05	65.5	72.71	13%	73.36	70.44	71.87	75.11	69.31	72.1
50%	82.56	66.58	**73.71**	15%	70.91	72.89	71.89	71.23	73.31	72.26
60%	81.73	70.74	75.84	17%	75.67	70.6	73.05	77.47	70.47	73.97
70%	81.16	75.29	76.12	**20%**	77.47	70.47	**73.97**	75.23	73.83	**74.52**

Table 2: **Labor-efficiency study on BLSTM-CRF and TMN.** "sent." means the percentage of the sentences (labeled only with entity tags) we use for BLSTM-CRF, while "trig." denotes the percentage of the sentences (labeled with both entity tags and trigger tags) we use for TMN.

Annotation time vs. performance. Although it is hard to accurately study the time cost on the crowd-sourcing platform we use[5], based on our offline simulation we argue that annotating both triggers and entities are about 1.5 times ("BLSTM-CRF (x1.5)") longer than only annotating entities. our offline simulation. In Figure 4, The x-axis for BLSTM-CRF means the number of sentences annotated with only entities, while for TMN means the number of sentences tagged with both entities and triggers. In order to reflect human annotators spending 1.5 to 2 times as long annotating triggers and entities as they spend annotating only entities, we stretch the x-axis for BLSTM-CRF. For example, the line labeled ("BLSTM-CRF (x2)") associates the actual F1 score for the model trained on 40% of the sentences with the x-axis value of 20%. We can clearly see that the proposed TMN outperforms the BLSTM-CRF model by a large margin. Even if we consider the extreme case that tagging triggers requires twice the human effort ("BLSTM-CRF (x2)"), the TMN is still significantly more labor-efficient in terms of F1 scores.

[5]Annotators may suspend jobs and resume them without interaction with the crowd-sourcing platform.

Interpretability. Figure 5 shows two examples illustrating that the trigger attention scores help the TMN model recognize entities. The training data has 'per day' as a trigger phrase for chemical-type entities, and this trigger matches the phrase 'once daily' in an unseen sentence during the inference phase of `TrigMatcher`. Similarly, in CoNLL03 the training data trigger phrase 'said it' matches with the phrase 'was quoted as saying' in an unlabeled sentence. These results not only support our argument that trigger-enhanced models such as TMN can effectively learn, but they also demonstrate that trigger-enhanced models can provide reasonable interpretation, something that lacks in other neural NER models.

5 Related Work

Towards low-resource learning for NER, recent works have mainly focused on dictionary-based distantly supervision (Shang et al., 2018; Yang et al., 2018; Liu et al., 2019). These approaches create an external large dictionary of entities, and then regard hard-matched sentences as additional, noisy-labeled data for learning a NER model. Although these approaches largely reduce human ef-

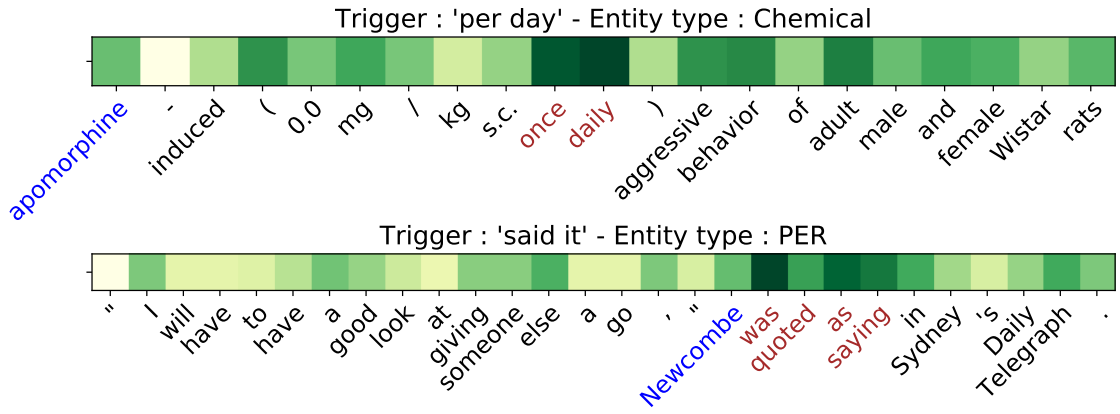

Figure 5: **Two case studies of trigger attention during inference.** The darker cells have higher attention weights.

forts in annotating, the quality of matched sentences is highly dependent on the coverage of the dictionary and the quality of the corpus. The learned models tend to have a bias towards entities with similar surface forms as the ones in dictionary. Without further tuning under better supervision, these models have low recall (Cao et al., 2019). *Linking rules* (Safranchik et al., 2020) focuses on the votes on whether adjacent elements in the sequence belong to the same class. Unlike these works aiming to get rid of training data or human annotations, our work focuses on how to more cost-effectively utilize human efforts.

Another line of research which also aims to use human efforts more cost-effectively is active learning (Shen et al., 2017; Lin et al., 2019). This approach focuses on instance sampling and the human annotation UI, asking workers to annotate the most useful instances first. However, a recent study (Lowell et al., 2019) argues that actively annotated data barely helps when training new models. Transfer learning approaches (Lin and Lu, 2018) and aggregating multi-source supervision (Lan et al., 2020) are also studied for using less expensive supervision for NER, while these methods usually lack clear rationales to advise annotation process unlike the trigger annotations.

Inspired by recent advances in learning sentence classification tasks (e.g., relation extraction and sentiment classification) with explanations or human-written rules (Li et al., 2018; Hancock et al., 2018; Wang* et al., 2020; Zhou et al., 2020), we propose the concept of an "entity trigger" for the task of named entity recognition. These prior works primarily focused on sentence classification, in which the rules (parsed from natural language explanations) are usually continuous token sequences and there is a single label for each input sentence. The unique challenge in NER is that we have to deal with rules which are discontinuous token sequences and there may be multiple rules applied at the same time for an input instance. We address this problem in TMN by jointly learning trigger representations and creating a soft matching module that works in the inference time.

We argue that either dictionary-based distant supervision or active learning can be used in the context of trigger-enhanced NER learning via our framework. For example, one could create a dictionary using a high-quality corpus and then apply active learning by asking human annotators to annotate the triggers chosen by an active sampling algorithm designed for TMN. We believe our work sheds light on future research for more cost-effectively using human to learn NER models.

6 Conclusion

In this paper, we introduce the concept of "entity trigger" as a complementary annotation. Individual entity annotations provide limited explicit supervision. Entity-trigger annotations add in complementary supervision signals and thus helps the model to learn and generalize more efficiently. We also crowdsourced triggers on two mainstream datasets and will release them to the community. We also propose a novel framework TMN which jointly learns trigger representations and soft matching module with self-attention such that can generalize to unseen sentences easily for tagging named entities. Future directions with TriggerNER includes: 1) developing models for

automatically extracting novel triggers, 2) transferring existing entity triggers to low-resource languages, and 3) improving trigger modeling with better structured inductive bias (e.g., OpenIE).

Acknowledgements

This research is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via Contract No. 2019-19051600007, NSF SMA 18-29268, and Snap research gift. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. We would like to thank all the collaborators in USC INK research lab for their constructive feedback on the work.

References

Yixin Cao, Zikun Hu, Tat-seng Chua, Zhiyuan Liu, and Heng Ji. 2019. Low-resource name tagging learned with weakly labeled data. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 261–270, Hong Kong, China. Association for Computational Linguistics.

R. Hadsell, S. Chopra, and Y. LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742.

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.

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.

Ouyu Lan, Xiao Huang, Bill Yuchen Lin, He Jiang, Liyuan Liu, and Xiang Ren. 2020. Learning to contextually aggregate multi-source supervision for sequence labeling. In *Proceedings of Association for Computational Linguistics*.

Dong-Ho Lee, Rahul Khanna, Bill Yuchen Lin, Jamin Chen, 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 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. 2016. Biocreative v cdr task corpus: a resource for chemical disease relation extraction. *Database : the journal of biological databases and curation*, 2016.

Shen Li, Hengru Xu, and Zhengdong Lu. 2018. Generalize symbolic knowledge with neural rule engine. *ArXiv*, abs/1808.10326.

Bill Y. Lin, Frank Xu, Zhiyi Luo, and Kenny Zhu. 2017a. Multi-channel BiLSTM-CRF model for emerging named entity recognition in social media. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 160–165, Copenhagen, Denmark. Association for Computational Linguistics.

Bill Yuchen Lin, Dong-Ho Lee, Frank F. Xu, Ouyu Lan, and Xiang Ren. 2019. AlpacaTag: An active learning-based crowd annotation framework for sequence tagging. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 58–63, Florence, Italy. Association for Computational Linguistics.

Bill Yuchen Lin and Wei Lu. 2018. Neural adaptation layers for cross-domain named entity recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2012–2022, Brussels, Belgium. Association for Computational Linguistics.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017b. A structured self-attentive sentence embedding. In *International Conference on Learning Representations*.

Tianyu Liu, Jin-Ge Yao, and Chin-Yew Lin. 2019. 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.

David Lowell, Zachary C. Lipton, and Byron C. Wallace. 2019. Practical obstacles to deploying active learning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 21–30, Hong Kong, China. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

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.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

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.

Chuck Rosenberg, Martial Hebert, and Henry Schneiderman. 2005. Semi-supervised self-training of object detection models. *2005 Seventh IEEE Workshops on Applications of Computer Vision (WACV/MOTION'05) - Volume 1*, 1:29–36.

Esteban Safranchik, Shiying Luo, and Stephen H. Bach. 2020. Weakly supervised sequence tagging from noisy rules. In *AAAI Conference on Artificial Intelligence (AAAI)*.

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.

Yanyao Shen, Hyokun Yun, Zachary Lipton, Yakov Kronrod, and Animashree Anandkumar. 2017. Deep active learning for named entity recognition. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 252–256, Vancouver, Canada. Association for Computational Linguistics.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

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 *International Conference on Learning Representations*.

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.

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. *The Web Conference*.