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**[†]
[†]University of Southern California [‡] Amazon
{yuchen.lin,dongho.lee,shenming,morenor}@usc.edu,
{huan183,xiangren}@usc.edu, shiralp@amazon.com

Institution

"Tom traveled a lot last year in Kasdfrcxzv."

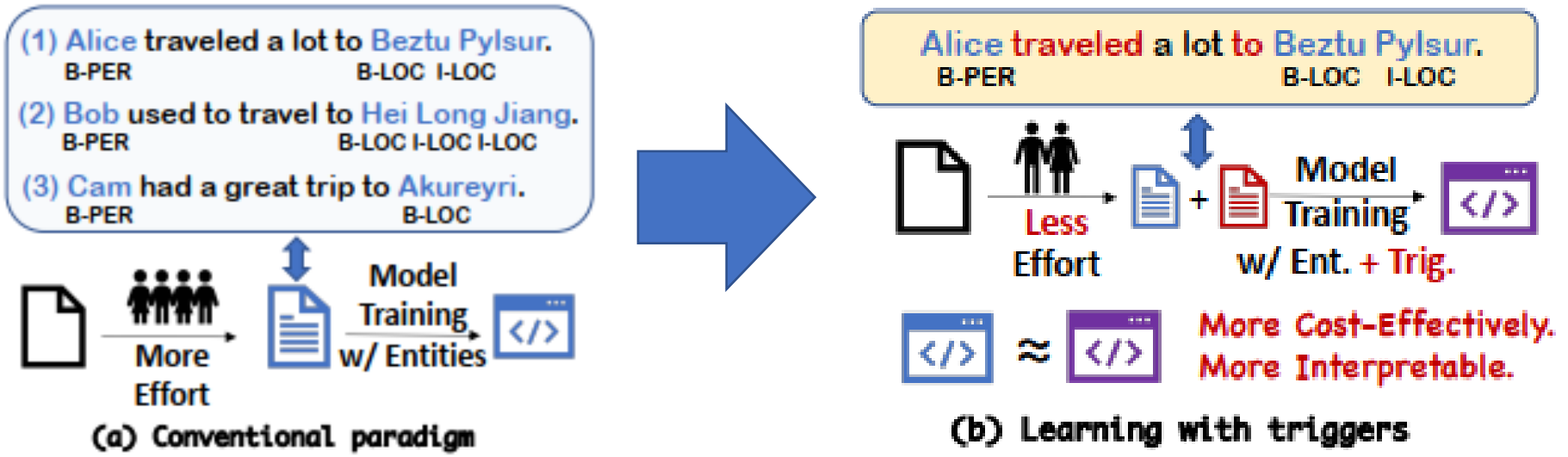
"travel…in" → ***"Entity Trigger"*** →

a group of words that can help explain the recognition process of a particular entity in the same sentence

- ***"Bill enjoyed a great dinner with Alice at Zcxlbz."***
- ***We had a fantastic lunch at Rumble Fish yesterday.***

Summary

- Introduce the concept of **“entity triggers**,” a novel form of explanatory annotation for named entity recognition problems.
- Propose a novel learning framework, named **Trigger Matching Network(TMM)**, which encodes entity triggers and softly grounds them on unlabeled sentences to increase the effectiveness of the base entity tagger
- Proposed trigger-based framework is significantly more cost-effective. →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


(1) Alice traveled a lot to Beztu Pylsur.
B-PER
B-LOC I-LOC
(2) Bob used to travel to Hei Long Jiang.
B-PER
B-LOC I-LOC I-LOC
(3) Cam had a great trip to Akureyri.
B-PER
B-LOC
More Effort
Model Training w/ Entities
(a) Conventional paradigm
Alice traveled a lot to Beztu Pylsur.
B-PER
B-LOC I-LOC
Less Effort
Model Training w/ Ent. + Trig.
More Cost-Effectively.
More Interpretable.
(b) Learning with triggers

Problem Formulation

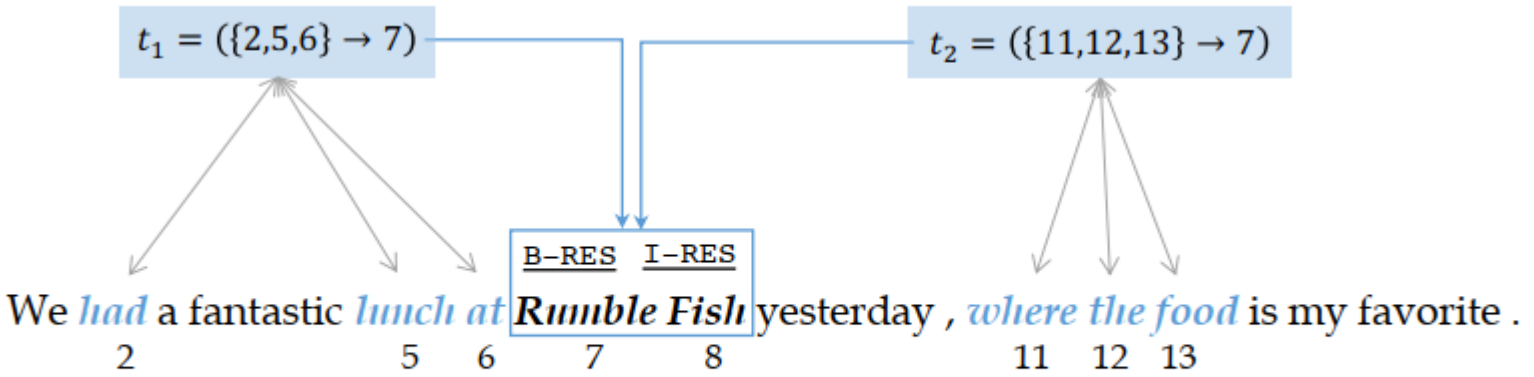


Figure 2: We show two individual **entity triggers**: t_1 ("*had ... lunch at*") and t_2 ("*where the food*"). Both are associated to the same entity mention "*Rumble Fish*" (starting from 7th token) typed as restaurant (RES).

$\mathbf{x} = [x(1); x(2); \cdots; x(n)]$
$\mathbf{y} = [y(1); y(2); \cdots; y(n)]$
$T(\mathbf{x}; \mathbf{y})$: represent the set of annotated entity triggers {t1,t2…tn}, ti is associated with an entity index *e* and a set of word indices *{wi}*.
Adding triggers creates a new form of data
$DT = \{(\mathbf{x_i}; \mathbf{y_i}; T(\mathbf{x_i}; \mathbf{y_i})\}$. Our goal is to learn a model for NER from a trigger-labeled dataset *DT*

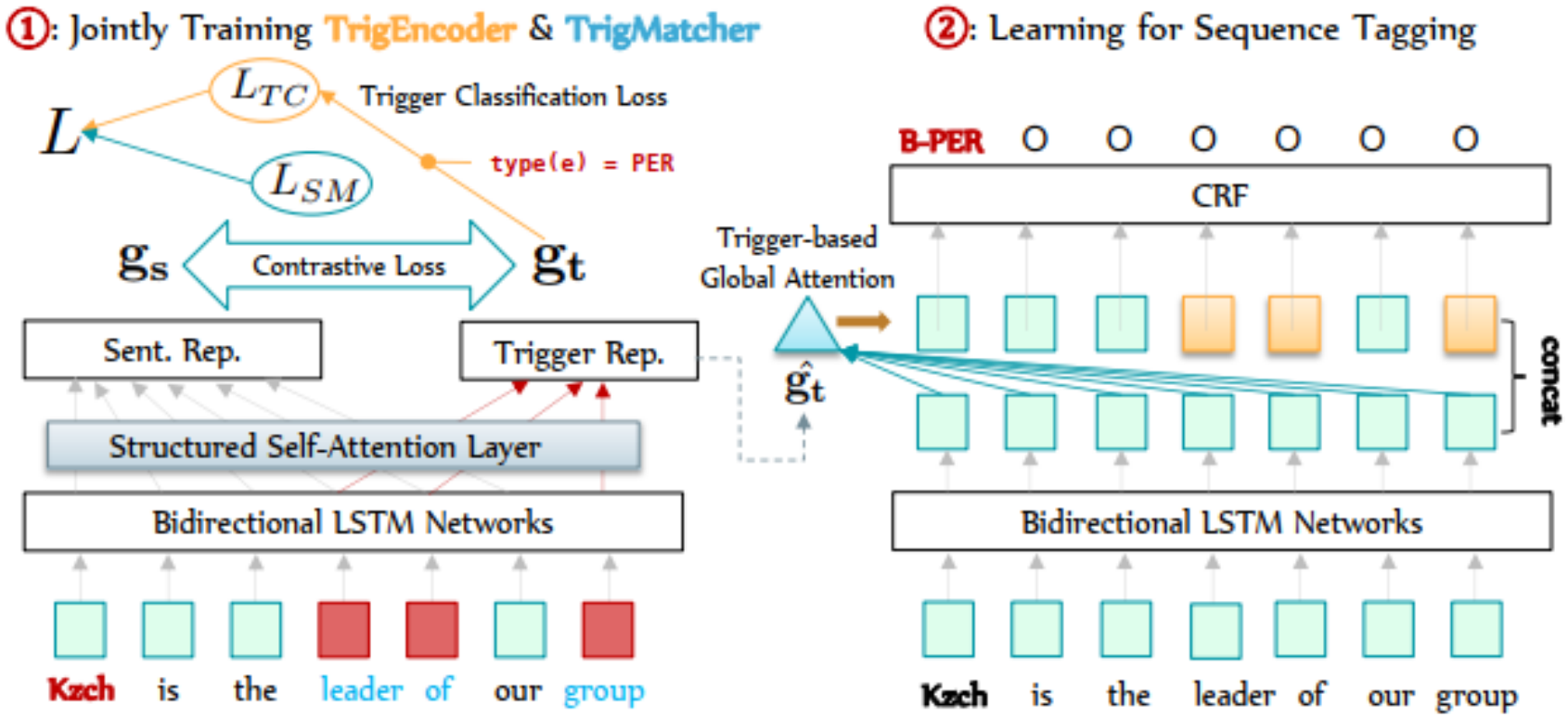


Figure 3: **Two-stage training of the *Trigger Matching Network*.** We first jointly train the `TrigEncoder` (via trigger classification) and the `TrigMatcher` (via contrastive loss). Then, we reuse the training data trigger vectors as attention queries in the `SeqTagger`.

3.1 Trigger Encoding & Semantic Matching

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., 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:

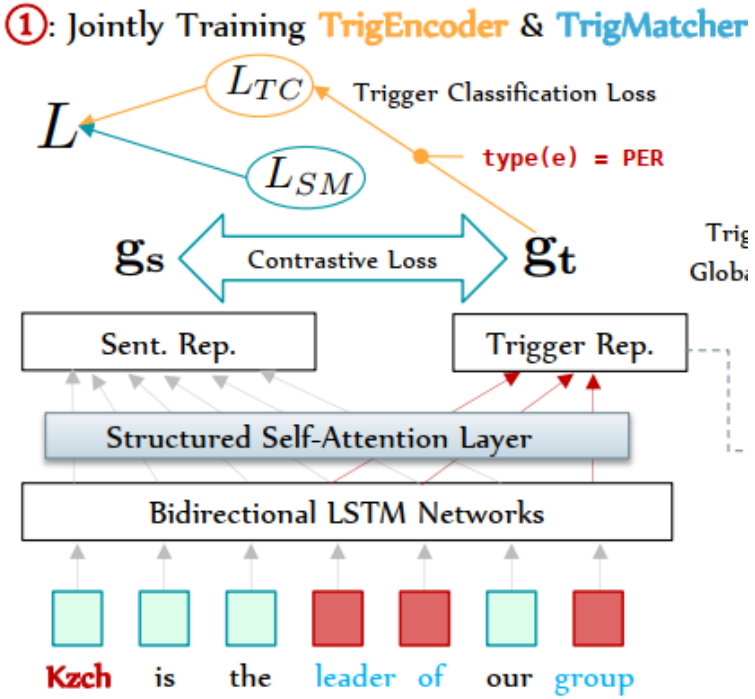
$$L_{TC} = -\sum \log \mathrm{P}\left(\texttt{type}(e) \mid \mathbf{g_t}; \theta_{TC}\right),$$

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

The intuition is that similar triggers and sentences should have close representations (i.e., have a small distance between them, d). We create neg-

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

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



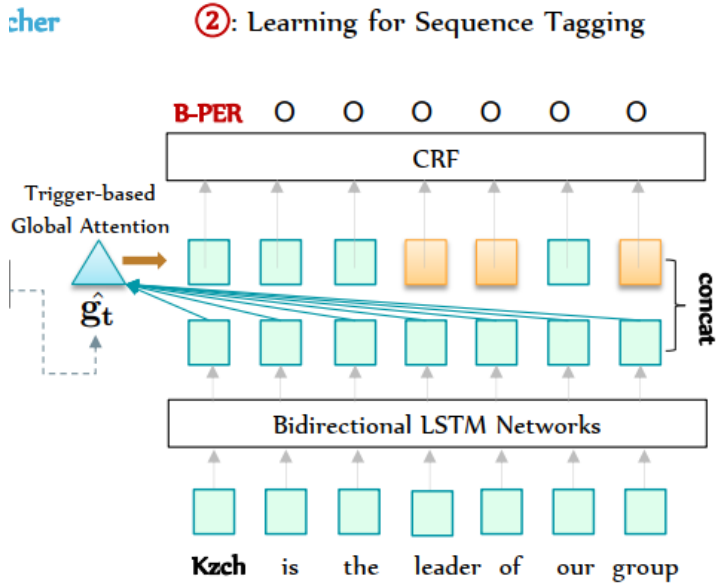
3.2 Trigger-Enhanced Sequence Tagging

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. 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, $\mathbf{H}'$.

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

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

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.



Inference on Unlabeled Sentences

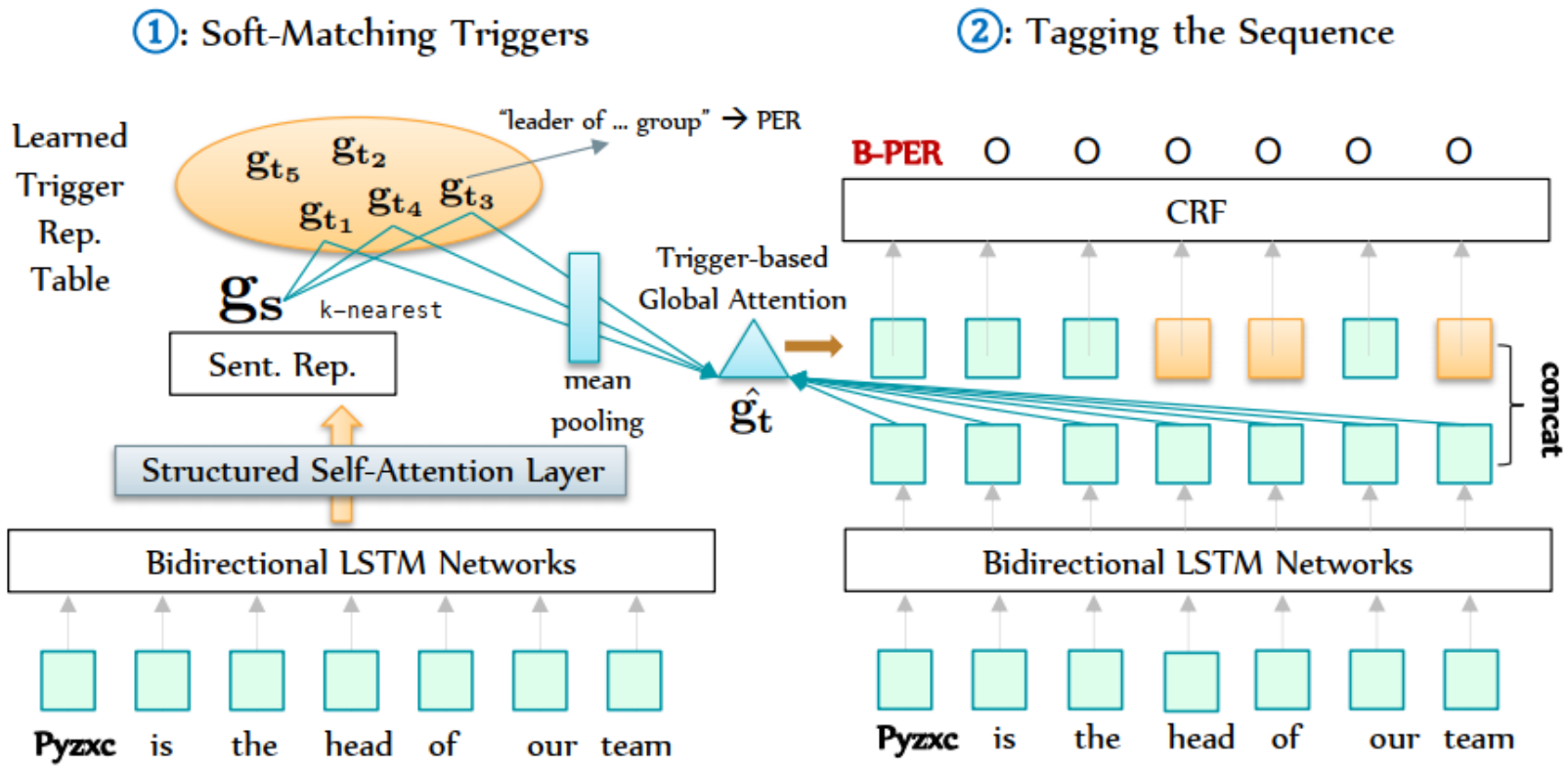


Figure 4: **The inference process of the TMN framework.** It uses the `TrigMatcher` to retrieve the k nearest triggers and average their trigger vectors as the attention query for the trained `SeqTagger`. Thus, an unseen cue phrase (e.g., *"head of ... team"*) can be matched with a seen trigger (e.g., *"leader of ... group"*).

$\mathcal{D}_T\}$. 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 $\hat{\mathbf{g}}_\mathbf{t}$, 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. 4.

Experiment

- CoNLL03 (generic domain)
- BC5CDR (biomedical domain)
- Sample 20% of each training set as inputs, and then reform them to be the same format(each entity with one of its triggers)

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.**

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.