Supplementary Material for “Better Modeling of Incomplete Annotations for Named Entity Recognition”

Zhanming Jie\(^1\), Pengjun Xie\(^2\), Wei Lu\(^1\), Ruixue Ding\(^2\) and Linlin Li\(^2\)
\(^1\)StatNLP Research Group, Singapore University of Technology and Design
\(^2\)DAMO Academy, Alibaba Group
zhanming.jie@mymail.sutd.edu.sg, luwei@sutd.edu.sg
\{chengchen.xpj, ada.drx, linyan.lll\}@alibaba-inc.com

Abstract

We present the industry dataset information and experimental details of the main paper (Jie et al., 2019) in this supplementary material.

1 Industry Dataset

To justify the robustness of our approach, we also conduct experiments on two datasets from industry: Taobao\(^1\) and Youku\(^2\).

Taobao is an e-commerce site with various types of products. We crawled the product titles from the website and annotate the titles with 9 entity types. Table 1 shows the detailed entity information of the Taobao dataset. The leftmost column represents the categorized entity types (i.e., PATTERN, PRODUCT, BRAND and MISC) we used in the experiments.

Youku is a video-streaming website where a number of videos from various domains are presented. We crawled the video titles from the website and again annotate them with 9 entity types. Table 2 shows the detailed entity information of the Youku dataset. Specifically, we group the entities into three types (i.e., FIGURE, PROGRAM and MISC) for our experiments.

As mentioned in the main paper, we further found that there are more entity labels per sentence on these two industry datasets compared to the standard datasets (i.e., CoNLL-2003 and CoNLL-2002). For example, there are 51% of entity labels per sentence (on average) in Taobao dataset whereas there are only 23% of entity labels per sentence (on average) in CoNLL-2003 dataset. Because of the high ratio of entity labels in Taobao and Youku datasets, the missing label CRF (M-CRF) can perform a lot less worse compared to the M-CRF on CoNLL datasets.

<table>
<thead>
<tr>
<th>Grouped Type</th>
<th>Entity Type</th>
<th>#Entity</th>
</tr>
</thead>
<tbody>
<tr>
<td>PATTERN</td>
<td>Model Type</td>
<td>2,173</td>
</tr>
<tr>
<td>PRODUCT</td>
<td>Product Description</td>
<td>5,506</td>
</tr>
<tr>
<td></td>
<td>Core Product</td>
<td>21,958</td>
</tr>
<tr>
<td>BRAND</td>
<td>Brand Description</td>
<td>331</td>
</tr>
<tr>
<td></td>
<td>Core Brand</td>
<td>3,430</td>
</tr>
<tr>
<td>MISC</td>
<td>Location</td>
<td>1,893</td>
</tr>
<tr>
<td></td>
<td>Person</td>
<td>367</td>
</tr>
<tr>
<td></td>
<td>Literature</td>
<td>814</td>
</tr>
<tr>
<td></td>
<td>Product Specification</td>
<td>2,732</td>
</tr>
</tbody>
</table>

Table 1: The entity information for the Taobao dataset.

<table>
<thead>
<tr>
<th>Grouped Type</th>
<th>Entity Type</th>
<th>#Entity</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIGURE</td>
<td>Figure</td>
<td>4,402</td>
</tr>
<tr>
<td>PROGRAM</td>
<td>Variety Show</td>
<td>1,349</td>
</tr>
<tr>
<td></td>
<td>Movie</td>
<td>1,303</td>
</tr>
<tr>
<td></td>
<td>Animation</td>
<td>3,133</td>
</tr>
<tr>
<td></td>
<td>TV Drama</td>
<td>3,087</td>
</tr>
<tr>
<td>MISC</td>
<td>Character</td>
<td>446</td>
</tr>
<tr>
<td></td>
<td>Number</td>
<td>1,022</td>
</tr>
<tr>
<td></td>
<td>Location</td>
<td>523</td>
</tr>
<tr>
<td></td>
<td>Song</td>
<td>640</td>
</tr>
</tbody>
</table>

Table 2: The entity information for the Youku dataset.

2 Experimental Details

This section introduces the settings of the baseline models and our approaches. Throughout the experiments, we apply the bidirectional long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) networks as our neural architecture (Lample et al., 2016) for conditional random fields (CRF) (Lafferty et al., 2001). Specifically, the hidden size of LSTM is set to 100, hidden size of character-level LSTM is set to 50, dropout is set to 0.5. During training, we use the Adam optimizer (Kingma and Ba, 2015) with a batch size of 1 and clipping value of 5. Our model is trained

\(^1\)https://www.taobao.com
\(^2\)https://www.youku.com
Algorithm 1 Partial Perceptron

Input: Training data: set of \((x, y_p) \in \mathcal{D}\)

Output: Model parameters \(w\)

1: \(w = 0\)
2: for \(i = 1 \ldots T\) do // T iterations
3:     for \((x, y_p) \in \mathcal{D}\) do
4:         \(z = \arg\max_{z} w \cdot f(x, z)\)
5:         if \(z \neq y_p\) then
6:             // check positions with gold labels
7:                 update(w)
8:         end if
9:     end for
10: end for

with 100 epochs. We select the above hyperparameters based on the best performance on development set.

2.1 Baseline Systems

Partial Perceptron We augment the partial perceptron model (Carlson et al., 2009) with BiLSTM as the neural architecture. In this model, we only consider the scores involved the tokens with available (i.e., annotated) labels during the training process. Algorithm 1 shows the procedure of training a partial perceptron.

Transductive Perceptron This model (Fernandes and Brefeld, 2011) augment an additional Hamming loss function during the update procedure in the structured perceptron. Essentially, they applied the max-margin training strategy in the structured perceptron. First, we obtain a pseudo ground-truth label sequence through a constrained Viterbi decoding:\(^3\):

\[
y_{pseudo} = \arg\max_{y \in \mathcal{C}(y_p)} w^T f(x, y) \tag{1}
\]

In other words, we first use the current model parameters to obtain a label sequence as pseudo gold sequence for structured perceptron training. Secondly, they obtain the prediction by max-margin decoding procedure:

\[
y = \arg\max_{y} [\text{loss}(y_{pseudo}, y) + w^T f(x, y)] \tag{2}
\]

where the loss function is a Hamming loss. Lastly, we perform a perceptron update:

\[
w' = w + f(x, y_{pseudo}) - f(x, \hat{y}) \tag{3}
\]

The Hamming loss in the transductive perceptron is defined as follows:

\[
\text{loss}(y_{pseudo}, y) = \sum_{t=1}^{|y|} \lambda(t) \tag{4}
\]

where \(\lambda(t) = \lambda_L\) when \(t\) is the time step that involves available labels, and \(\lambda(t) = \lambda_U\) when \(t\) is the time step that involves unavailable labels. During our experiments, we set \(\lambda_L = 1\) and \(\lambda_U = 0.1\).

2.2 Implementation Details

We perform iterative training for our hard and soft approaches. As mentioned in the main paper, we perform iterative training in a \(k\)-fold cross-validation manner. Empirically, we set the maximum iteration number to 10, which is enough for our approaches to converge and have a stable \(q\) distribution.

References


