A Baseline Fine-Grained Entity Extraction System for TAC-KBP2019

Ying Lin, Xiaoman Pan, Manling Li, Heng Ji
Computer Science Department
University of Illinois at Urbana-Champaign
hengji@illinois.edu

Abstract
For fine-grained entity extraction, we propose a fine-grained entity typing model with a novel attention mechanism and a hybrid type classifier. We advance existing methods in two aspects: feature extraction and type prediction. To capture richer contextual information, we adopt contextualized word representations instead of fixed word embeddings used in previous work. In addition, we propose a two-step mention-aware attention mechanism to enable the model to focus on important words in mentions and contexts. We also develop a hybrid classification method beyond binary relevance to exploit type interdependency with latent type representation. Instead of independently predicting each type, we predict a low-dimensional vector that encodes latent type features and reconstruct the type vector from this latent representation.

1 Introduction
To assist the coordination of TAC-KBP2019, UIUC team has developed a simple system for fine-grained entity extraction to serve as a baseline, for comparing other more sophisticated methods and also testing the integration of docker containers into NIST platform.

2 Named Mention Extraction
2.1 Coarse-grained Named Mention Extraction
We implement an LSTM-CNN model with ELMo contextualized word representations to extraction named mentions. The basic model consists of an embedding layer, a character-level network, a bi-directional long-short term memory (LSTM) layer, a linear layer, and a conditional random fields (CRF) layer. In this architecture, each sentence is represented as a sequence of vectors \( X = \{ x_1, ..., x_L \} \), where \( x_i \) represents features of the \( i \)-th word. We use two types of features in our model: 1. Word embedding that encodes the semantic information of words. 2. Character-level representation that captures subword information. We utilize character features as word embeddings take words as atomic units and ignore useful subword clues, and pre-trained word embeddings are not available for unknown words and a large number of rare words.

The LSTM layer then processes the sentence in a sequential manner and encodes both contextual and non-contextual features of each word \( x_i \) into a hidden state \( h_i \). After that, we decode the hidden state into a score vector \( y_i \) with a linear layer. The value of each component of \( y_i \) represents the predicted score of a label. However, as the label of each token is predicted separately, the model may produce a path of inconsistent tags such as [B-GPE, I-GPE, S-GPE]. Therefore, we add a CRF layer on top of the model to capture tag dependencies and predict a global optimal tag path for each sentence. Given an sentence \( X \) and scores predicted by the linear layer \( Y = \{ y_1, ..., y_L \} \), the score of a sequence of tags is calculated as:

\[
s(X, \hat{z}) = \sum_{i=1}^{L+1} A_{\hat{z}_{i-1}, \hat{z}_i} + \sum_{i=1}^{L} y_i, \hat{z}_i,
\]

where each entry \( A_{\hat{z}_{i-1}, \hat{z}_i} \) is the score of jumping from tag \( \hat{z}_{i-1} \) to tag \( \hat{z}_i \), and \( y_i, \hat{z}_i \) is the \( \hat{z}_i \) dimension of \( y_i \) that corresponds to tag \( \hat{z}_i \). We append two special tags \#start\# (\( \hat{z}_0 \)) and \#end\# (\( \hat{z}_{L+1} \)) to denote the beginning or end of a sentence. Finally, we maximize the sentence-level log-likelihood of the gold tag path \( z \) given the input sentence by

\[
\log p(z|X) = \log \left( \frac{e^{s(X,z)}}{\sum_{\hat{z} \in Z} e^{s(X,\hat{z})}} \right) = s(X,z) - \log \sum_{\hat{z} \in Z} e^{s(X,\hat{z})},
\]

where \( Z \) denotes the set of all possible paths.
For English, we improve the model by incorporating ELMo contextualized word representations. We use a pre-trained ELMo encoder to generate the contextualized word embedding $c_i$ for each token and concatenate it with $h_i$.

We train separate models for named, nominal, and pronominal mentions and merge their outputs into the final mention extraction result.

We also explore a reliability-aware dynamic feature composition mechanism to obtain better representations for rare and unseen words. We design a set of frequency-based reliability signals to indicate the quality of each word embedding. These signals control mixing gates at different levels in the model. For example, if a word is rare, the model will rely less on its pre-trained word embedding, which is usually not well trained, but assign higher weights to its character and contextual features.

### 2.2 Fine-grained Name Mention Extraction

Fine-grained entity typing is performed on the mention extraction result. We develop an attentive classification model (Lin and Ji, 2019) that takes a mention with its context sentence and predicts the most possible fine-grained type. Unlike previous neural models that generally use fixed word embeddings and task-specific networks to encode the sentence, we employ contextualized word representations (Peters et al., 2018) that can capture word semantics in different contexts.

After that, we use a novel two-step attention mechanism to extract crucial information from the mention and its context as follows

$$m = \sum_i a_i^m r_i,$$

$$c = \sum_i a_i^c r_i,$$

where $r_i \in \mathbb{R}^{d_r}$ is the vector of the $i$-th word, $d_r$ is the dimension of $r$, and attention scores $a_i^m$ and $a_i^c$ are calculated as

$$a_i^m = \text{Softmax}(v^m^T \tanh(W^m r_i)),$$

$$a_i^c = \text{Softmax}(v^c^T \tanh(W^c (r_i) \oplus m \oplus p_i)),$$

$$p_i = \left(1 - \mu \left(\min(|i - a|, |i - b|) - 1\right)\right)^{+},$$

where parameters $W^m \in \mathbb{R}^{d_a \times d_r}$, $v^m \in \mathbb{R}^{d_a}$, $W^c \in \mathbb{R}^{d_a \times (2d_r + 1)}$, and $v^c \in \mathbb{R}^{d_a}$ are learned during training, $a$ and $b$ are indices of the first and last words of the mention, $d_a$ is set to $d_r$, and $\mu$ is set to 0.1.

Next, we adopt a hybrid type classification model consisting of two classifiers. We first learn a matrix $W^b \in \mathbb{R}^{d_r \times 2d_r}$ to predict type scores by

$$\hat{y}^b = W^b (m \oplus c),$$

where $\hat{y}^b_i$ is the score for the $i$-th type.

We also learn to predict the latent type representation from the feature vector using

$$l = V^l (m \oplus c),$$

where $V^l \in \mathbb{R}^{2d_r \times d_l}$. We then recover a type vector from this latent representation using

$$\hat{y} = U \Sigma l,$$

where $U$ and $\Sigma$ are obtained via Singular Value Decomposition (SVD) as

$$Y \approx \hat{Y} = U \Sigma L^T,$$

where $U \in \mathbb{R}^{d_l \times d_l}$, $\Sigma \in \mathbb{R}^{d_l \times d_l}$, $L \in \mathbb{R}^{N \times d_l}$, and $d_l \ll d_l$. Finally, we combine scores from both classifier

$$\hat{y} = \sigma(W^b (m \oplus c) + \gamma W^l l),$$

where $\gamma$ is set to 0.1. The training objective is to minimize the cross-entropy loss function as

$$J(\theta) = -\frac{1}{N} \sum_i y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i).$$

Furthermore, we get the YAGO fine-grained types by linking entities to the Freebase (LDC2015E42), and mapped them to AIDA entity types. Besides, for GPE and LOC entities, we link them to GeoNames \(^1\) and decide their fine-grained types using GeoNames attributes feature_class and feature_code. We compute a weighted score for these typing results and normalize the score as typing confidence.

\(^1\)http://geonames.org/
References
