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 bidirectional 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 $\boldsymbol{X} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_L\}$, where $\boldsymbol{x}_i$ 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 embddings 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 $\boldsymbol{x}_i$ into a hidden state $\boldsymbol{h}_i$. After that, we decode the hidden state into a score vector $\boldsymbol{y}_i$ with a linear layer. The value of each component of $\boldsymbol{y}_i$ 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 $\boldsymbol{X}$ and scores predicted by the linear layer $\boldsymbol{Y} = \{\boldsymbol{y}_1, ..., \boldsymbol{y}_L\}$, the score of a sequence of tags is calculated as:

$$s(\boldsymbol{X}, \hat{\boldsymbol{z}}) = \sum_{i=1}^{L+1} \boldsymbol{A}_{\hat{z}_{i-1},\hat{z}} + \sum_{i=1}^{L} y_{i,\hat{z}_i},$$

where each entry $\boldsymbol{A}_{\hat{z}_{i-1},\hat{z}_i}$ 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 $\boldsymbol{y}_i$ 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 $\boldsymbol{z}$ given the input sentence by

$$\begin{aligned}\log p(\boldsymbol{z}|\boldsymbol{X}) &= \log \left(\frac{e^{s(\boldsymbol{X},\boldsymbol{z})}}{\sum_{\hat{\boldsymbol{z}} \in Z} e^{s(\boldsymbol{X},\hat{\boldsymbol{z}})}} \right) \\ &= s(\boldsymbol{X}, \boldsymbol{z}) - \log \sum_{\hat{\boldsymbol{z}} \in Z} e^{s(\boldsymbol{X},\hat{\boldsymbol{z}})},\end{aligned}$$

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 $\boldsymbol{c}_i$ for each token and concatenate it with $\boldsymbol{h}_i$.

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

$$\boldsymbol{m} = \sum_{M}^{i} a_i^m \boldsymbol{r}_i,$$

$$\boldsymbol{c} = \sum_{C}^{i} a_i^c \boldsymbol{r}_i,$$

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

$$a_i^m = \text{Softmax}(\boldsymbol{v}^{m\top} \tanh(\boldsymbol{W}^m \boldsymbol{r}_i)),$$

$$a_i^c = \text{Softmax}(\boldsymbol{v}^{c\top} \tanh(\boldsymbol{W}^c (\boldsymbol{r}_i) \oplus \boldsymbol{m} \oplus p_i)),$$

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

where parameters $\boldsymbol{W}^m \in \mathbb{R}^{d_a \times d_r}$, $\boldsymbol{v}^m \in \mathbb{R}^{d_a}$, $\boldsymbol{W}^c \in \mathbb{R}^{d_a \times (2d_r+1)}$, and $\boldsymbol{v}^c \in \mathbb{R}^{d_a}$ 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 μ is set to 0.1.

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

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

where $\tilde{y}_i^b$ is the score for the i-th type.

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

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

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

$$\tilde{\boldsymbol{y}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{l},$$

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

$$\boldsymbol{Y} \approx \tilde{\boldsymbol{Y}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{L}^\top,$$

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

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

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

$$J(\theta) = -\frac{1}{N} \sum_{i}^{N} \boldsymbol{y}_i \log \tilde{y}_i + (1 - \boldsymbol{y}_i) \log(1 - \tilde{\boldsymbol{y}}_i).$$

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

Ying Lin and Heng Ji. 2019. An attentive fine-grained entity typing model with latent type representation. In *Proc. 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP2019)*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics.