Multi-Task Transfer Learning for Fine-Grained Named Entity Recognition

Masato Hagiwara[1] **Ryuji Tamaki**[2] **Ikuya Yamada**[2]
[1]Octanove Labs LLC [2]Studio Ousia, Inc.
masato@octanove.com {ikuya, ryuji.tamaki}@ousia.jp

Abstract

This paper describes Studio Ousia's participation to the EDL track of TAC KBP 2019—(ultra) fine-grained named entity recognition (NER). The proposed system first trains a YAGO-based ultra fine-grained NER model in a multi-label, multi-task fashion. The pre-trained model is then fine-tuned to adopt it to the AIDA taxonomy by adding a lightweight conversion layer. The experiments have shown that this transfer learning approach outperforms a simpler direct method which directly trains the YAGO-based NER model.

1 Introduction

Named entity recognition (NER) is a task where named entity mentions in natural language text are detected and classified into appropriate types. NER is often an important first step for downstream NLP tasks such as entity linking, event detection, and question answering. Traditionally, NER systems only dealt with coarse-grained entity types such as person, geo-political entity (GPE), organization, etc, while downstream tasks often benefit from more fine-grained types such as politician, city, sport team, etc. It was not until the 2010s did more fine-grained entity types such as FIGER (Ling and Weld, 2012) start to be used. However, most work (Ren et al., 2016; Shimaoka et al., 2016; Yogatama et al., 2015) has been focused on fine-grained entity typing (vs entity detection), while few systems have proposed to solve entity detection and classification jointly. Detecting and classifying a large number of entity types has been a challenge for many traditional NER systems due to the high computational cost and the lack of large-scale training data.

This is also the case for the EDL track of TAC KBP 2019, where the task is to build a fine-grained NER system for texts annotated with the AIDA taxonomy. Since there is little training data for this task, we turn to the use of transfer learning, where we train a more general, more fine-grained NER model using the YAGO taxonomy, then fine-tune it using a small dataset.

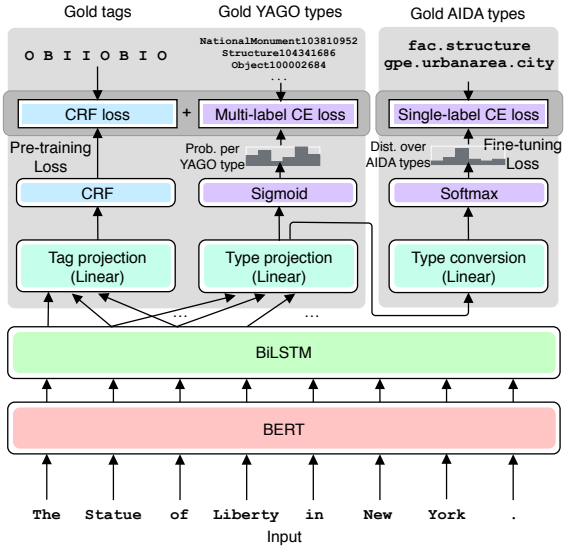

Figure 1: Overview of our fine-grained NER system

We propose an NER system, named OusiaNER, which solves (ultra) fine-grained named entity detection and type classification jointly in a multi-task fashion. The system is capable of detecting both fine-grained (hundreds of types) and ultra fine-grained (thousands of types) entity mentions. The training of the system involves two phases: first, an ultra fine-grained, YAGO-based, multi-label NER tagger is trained from a large silver-standard dataset created from Wikipedia and OntoNotes (Weischedel et al., 2011); second, the pre-trained model is fine-tuned using a small, manually annotated dataset to adopt it to the AIDA taxonomy (Figure 1).

2 Model

2.1 Pre-training

A typical NER system, be it based on traditional machine learning or neural networks, employs a CRF-layer (Huang et al., 2015; Lample et al., 2016) combined with a tag scheme such as IOB2. For a very large taxonomy like YAGO, this is prohibitively expensive, since the complexity of a CFR-layer is proportional to $O(k^2)$ where k is the number of tags. OusiaNER instead opted to separate named entity detection and classification and solve them in a multi-task setting.

Specifically, the neural network architecture consists of two main parts—one for detection and the other for classification. For detection, the embedded input (after BERT and BiLSTM) goes through a linear tag projection layer and a CRF layer to obtain a sequence of the standard IOB2 tags (O, B, and I, without any type information). On the other hand, the same input is passed onto a type projection layer then an element-wise sigmoid layer to obtain the probability per each YAGO type. The type classification layer is trained in a multi-label setting. In an ultra fine-grained taxonomy like YAGO, there is often a large overlap and annotation ambiguity between different types and we hypothesize that a multi-label setting can retain richer information about mentions. The final training objective of the pre-training phase is the weighted sum of the CRF tag loss (log likelihood) and the multi-label type loss (cross entropy).

2.2 Fine-Tuning

At the fine-tuning phase, the output of the trained type projection layer is fed to another two-layer feedforward network to convert YAGO-based logits to AIDA-based ones. The network is fine-tuned using a single-label soft cross entropy loss that takes into account the hierarchical nature of the AIDA taxonomy:

$$L_{\text{soft}} = -\sum_{z_i} y_i^* \log p(z_i) \tag{1}$$

The soft gold label y_i^* is of i-th type is constructed by multiplying the binary gold label by the weight matrix, where the (i, j) element of the weight matrix is the "partial score" the system would get by predicting the label j where the gold label is i:

$$\mathbf{y}^* = \mathbf{y}W \tag{2}$$

Using this soft cross entropy loss instead of its "hard" version boosts the performance of the final model by around 10% (see the experiment section for more details). The parameters of the named entity detection portion of the model were fixed during fine-tuning.

3 Training Data

For pre-training, we used a combination of the following two resources:

- A subset of OntoNotes 5.0 from newswire and Web pages. This dataset only contributes to the detection loss.
- A Wikipedia-based silver-standard dataset where entity mentions are automatically tagged and typed with the YAGO taxonomy. We created this dataset by using Wikipedia2Vec (Yamada et al., 2018) to extract page contents and inter-page links, then mapped the linked articles to YAGO types using the mapping table distributed by the TAC KBP organizers. This dataset only contributes to the classification loss.

The hyperparameters are tuned using a small held-out subset of the manually annotated Wikipedia portion.

For fine-tuning, we first built a small dataset comprised of 1) a small held-out subset of the Wikipedia corpus mentioned above, 2) a small held-out subset of the OntoNotes 5.0 corpus, and 3) AIDA Source Data (LDC2019E04) where missing annotations are manually corrected. The dataset is then split into training and validation portions.

4 Experiments

4.1 Settings

We implemented the entire system using AllenNLP (Gardner et al., 2018). For embedding the input, we used the BERT (Devlin, Jacob et al., 2018) (`bert-base-cased`) model with an two-layer BiLSTM on top of it. The size of the hidden states is 200. A dropout layer ($p = 0.5$) was inserted after BERT as well as the BiLSTM layer. The pre-training loss was optimized using the Adam optimizer with a learning rate of 0.001.

Model	Prec.	Rec.	F1
Direct	0.45	0.42	0.43
Fine-tuned	**0.65**	**0.57**	**0.61**
Fine-tuned w/o hier. loss	0.60	0.50	0.55

Table 1: Performance of NER on the validation set

Run	Prec.	Rec.	F1
1st submission	0.504	0.468	0.485
After feedback	**0.506**	**0.493**	**0.499**

Table 2: Test performance for the different runs

The type conversion layer is a two-layer feed-forward network with a ReLU nonlinearity in between. The size of the hidden layer is again 200. The network is fine-tuned using the BertAdam optimizer with a learning rate of 1.0^{-5} and 2,500 warm-up steps.

4.2 Results

We compared our model ("Fine-tuned" in Table 1) against a variant ("Direct") where the same architecture is used to directly predict AIDA-based named entities (i.e., without fine-tuning) where the YAGO types are automatically converted to AIDA ones using the official mapping table. The fine-tuning method outperformed a simpler direct method. We also included a model that is identical to "Fine-tuned" except it uses the vanilla cross-entropy loss instead of the hierarchical loss in Section 2.2. The result shows that the use of the hierarchical loss leads to an approx. 10% boost of the F1 measure.

After receiving human feedback, we added the additional training instances to the dataset and fine-tuned the final model, without re-training the YAGO-based model. Table 2 shows the performance comparison between different runs.

5 Conclusion

We proposed OusiaNER, a multi-task, transfer learning approach for (ultra) fine-grained NER. Our key contributions are: (1) use of a transfer learning approach due to the lack of the target domain training data, (2) use of a multi-task, multi-task model for joint NE detection and classification, and (3) the hierarchy-aware loss for modeling partial credit.

There could be many possible directions for improvements. One could think of structuring the model in a hierarchical fashion in order to capture the label relationship. Incorporating richer linguistic sources (such as the embeddings of label strings, gazetteers, and/or hand-crafted features) is also future work.

References

Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton, and Toutanova, Kristina. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL 2018*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A Deep Semantic Natural Language Processing Platform. *arXiv.org*.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF Models for Sequence Tagging. *arXiv.org*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. In *Proceedings of HLT-NAACL 2016*.

Xiao Ling and Daniel S Weld. 2012. Fine-Grained Entity Recognition. In *Proceedings of AAAI 2012*.

X Ren, W He, M Qu, L Huang, H Ji, and J Han. 2016. AFET: Automatic Fine-Grained Entity Typing by Hierarchical Partial-Label Embedding. In *Proceedings of EMNLP 2016*.

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2016. Neural Architectures for Fine-grained Entity Type Classification. In *Proceedings of EACL 2017*.

Ralph Weischedel, Eduard Hovy, Mitchell Marcus, Martha Palmer, Robert Belvin, S Pradan, Lance Ramshaw, and Nianwen Xue. 2011. *OntoNotes: A Large Training Corpus for Enhanced Processing*. Springer.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2018. Wikipedia2Vec: An Optimized Tool for Learning Embeddings of Words and Entities from Wikipedia. *arXiv preprint 1812.06280*.

Dani Yogatama, Daniel Gillick, and Nevena Lazic. 2015. Embedding Methods for Fine Grained Entity Type Classification. In *Proceedings of ACL 2015*.