A Simple Entity-Specific Encoding for Joint Entity and Relation Extraction

Shibo Hao

July 29, 2021

Abstract

Entity and relation extraction aims to identify named entities from plain texts and relations among them. These two task are typically modeled jointly with a neural network in an end-to-end manner. Recent work [11] models the two tasks separately, and propose a pipeline where the entity information is fused to relation extractor at the input layer. Surprisingly, that model outperforms a lot of strong joint models. In this work, we follow their separate modeling style, and propose an entity-specific encoding with multi-head attention mechanism. The model (BERT-JRE) reaches results comparable with input-level fusion in [11] on relation extraction, but are much more time-efficient.

1 Introduction

Named Entity Recognition (NER) and Relation Extraction (RE) are two fundamental tasks for Information Extraction, which takes texts as input, recognizes named entity mentions and extract relations among them respectively. In the traditional approaches, the two tasks are finished by two separately trained models [1], but recent solutions tend to model them jointly in the end-to-end manner [3–5, 9]. It's claimed that joint models alleviate the error propagation issue and benefit from the interrelation. However, recently the pipeline-style model [11] outperforms a lot of joint models, which cast a shadow over the advantage of end-to-end models.

Popular approaches for NER contain sequence tagging and span classification [7]. Sequence tagging allocates each token a tag following BIO or BILOU scheme and then decode to entities, while span classification directly decide whether each candidate span is an entity of some certain types. The latter approach can better handle the overlapping of entities.

The key to the performance of RE models are the representation of entity pairs. One naive method is to encode a sentence with PLM, get entity representations, and concatenate them as entity pair encoding. Many works claim that this naive method doesn't involve enough information for the target relation to be classified. It's better to insert head and tail entity markers into the input sequence, so that the deep transformers [8] in PLM can

generate more entity-specific representations for the target entity pairs. Essentially, it is a trade-off between time and accuracy. Although this model is better to encode entity-specific information, it has to compute a representation from scratch for each pair of entities, which is more than 100x expensive than the naive encoding. In this work, we use multi-head attention to obtain entity-specific context embedding, with naive entity embedding as query and other token embedding as keys and values. This basically won't slow down the model than the trivial encoding method, since the main workload is still the PLM encoding.

2 Model

The input sequence is a sentence $S = \{s_1, s_2, ..., s_n\}$. Gold entities set is $E = \{e_1, e_2, ..., e_{N_e}\}$, where $e_i = (b_i, e_i, t_i), r_i = (e_h, e_t, l_i), b_i, e_i \in \{0, 1, ..., n\}, t_i \in \mathcal{T}$, and gold relation set is $R = \{r_1, r_2, ..., r_{N_r}\}$, where $e_h, e_t \in E, t_i \in \mathcal{L}$, Let $C = \{c_1, c_2, ..., c_m\}$ be all candidate spans to be classified as entities, where $c_i = (b_i, e_i), b_i, e_i \in \{0, 1, ..., n\}$ contains the begin index and end index of a candidate span. To control the size of candidate set C, each candidate is required to be shorter than L.

For NER, the sentence S is first fed into a BERT [2] model to get contextualized embedding E_{BERT} for all the tokens. Then for each candidate span $c_i = (b_i, e_i)$, we take the concatenation of the begin and end token embedding as the span representation $E_{span,c_i} = [E_{BERT,b_i}; E_{BERT,e_i}]$. Then the span embedding is fed into a multi-layer perceptron (MLP) to get logits of each entity type.

For RE, the sentence S is also fed into a BERT encoder to get contextualized embedding E_{BERT} for all the tokens. For each gold entity $e_i = (b_i, e_i, t_i)$, we take the concatenation of the begin and end token embedding as entity representation $E_{entity,e_i} = [E_{BERT,b_i}; E_{BERT,e_i}]$. Then we get an entity pair representation by concatenate head and tail entity embedding and project it into a vector E_{pair,e_i,e_j} with the same dimension of E_{BERT}. The resulting vector is used as query vector to calculate multi-head attention [8] with E_{BERT} as key and value vectors. The resulting embedding is called $E_{context,e_i,e_j}$. Finally, the concatenation $[E_{context,e_i,e_j}; E_{pair,e_i,e_j}]$ is fed into a MLP to get logits for each relation labels.

3 Experiment

3.1 Dataset and Settings

We evaluate our framework on ACE05 dataset. The corpora is collected from a variety of domains, and contains annotations of entities, values, temporal expressions, relations, and events. For the task of joint entity and relation extraction, we only take the texts, entities and relations of ACE05. We pre-process the dataset with scripts[1] used in [9].

Since the evaluation setting of ACE05 is confusing [7], we strictly following previous works [3–5, 9], and report our results assuming that an entity is recognized correctly when the predicted type and span are exactly the same as gold labels. We use both micro-F1 as an evaluation metrics. Separate and joint extraction are reported respectively. Note that the

[1] https://github.com/LorrinWWW/two-are-better-than-one

Label	P	R	F1
ART	0.62	0.47	0.54
ORG-AFF	0.8	0.86	0.83
GEN-AFF	0.67	0.47	0.55
PHYS	0.53	0.57	0.55
PER-SOC	0.64	0.79	0.7
PART-WHOLE	0.73	0.71	0.72
TOTAL	0.67	0.66	0.66

Table 1: Model's performance on ACE-05 Relation Extraction

Label	P	R	F1
FAC	0.79	0.7	0.74
WEA	0.77	0.69	0.72
LOC	0.75	0.7	0.72
VEH	0.84	0.83	0.83
GPE	0.89	0.88	0.89
ORG	0.79	0.78	0.78
PER	0.9	0.9	0.9
TOTAL	0.87	0.86	0.87

Table 2: Model's performance on ACE-05 Named Entity Recognition

choice of PLM has influence on the results. Previous works typically apply larger models, but due to the computational limitation, we take bert-base-uncased[2] as our encoder. The model is implemented with the help of PyTorch [6] and Transformers [10].

3.2 Results and Analysis

The detailed results are in Table 1, Table 2. We also report the results of several models in Table 3. For comparison, we show the results of some previous works along with our models. The baseline removes the multi-head attention layer from our BERT-JRE model. We can see that BERT-JRE improves 2.1% F1 over Baseline, and reaches comparable results with PURE(BERT-base) on the relation extraction task.

References

[1] Nguyen Bach and Sameer Badaskar. A review of relation extraction. 2007.

[2] J. Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

[3] Q. Li and Heng Ji. Incremental joint extraction of entity mentions and relations. In *ACL*, 2014.

[2]https://huggingface.co/bert-base-uncased

Model	Encoder	Entity		Relation	
		Dev	Test	Dev	Test
MD-RNN [9]	ALBERT-xxl	-	89.5	-	67.6
PURE [11]	ALBERT-xxl	-	89.7	-	69.0
PURE [11]	BERT-base	-	88.7	-	66.7
Baseline (ours)	BERT-base	85.5	87.4	62.1	64.5
BERT-JRE (ours)	BERT-base	85.5	87.4	64.2	66.4

Table 3: Model's performance on ACE-05. The results of cited works is as reported according to original papers. Note that we take results of [11] from single-sentence setting.

[4] Yi Luan, David Wadden, Luheng He, Amy Shah, Mari Ostendorf, and Hannaneh Hajishirzi. A general framework for information extraction using dynamic span graphs. In *NAACL*, 2019.

[5] Makoto Miwa and Mohit Bansal. End-to-end relation extraction using lstms on sequences and tree structures. *ArXiv*, abs/1601.00770, 2016.

[6] Adam Paszke, S. Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, N. Gimelshein, L. Antiga, Alban Desmaison, Andreas Köpf, E. Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

[7] Bruno Taillé, Vincent Guigue, Geoffrey Scoutheeten, and Patrick Gallinari. Let's stop incorrect comparisons in end-to-end relation extraction! *arXiv preprint arXiv:2009.10684*, 2020.

[8] Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *ArXiv*, abs/1706.03762, 2017.

[9] Jue Wang and Wei Lu. Two are better than one: Joint entity and relation extraction with table-sequence encoders. *ArXiv*, abs/2010.03851, 2020.

[10] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019.

[11] Zexuan Zhong and Danqi Chen. A frustratingly easy approach for joint entity and relation extraction. *ArXiv*, abs/2010.12812, 2020.