Survey of techniques for domain specific Named Entity Recognition

Manik Bhandari
Indian Institute of Science
manikb@iisc.ac.in

Abstract

This document is a brief summary of relevant work in the field of domain- specific Named Entity Recognition (NER).

1 Introduction

1.1 Define the problem

1.2 Major Challenges

1. Lack of corpus.
2. Lack of supervised NER tags - you have to manually define the set of entities that you are interested in so this set will always be small.
3. Disambiguation - same surface form can mean two things in different context.

2 Summary of Papers

2.1 AutoNER

AutoNER (Shang et al., 2018b) is the recent SOTA in domain-specific NER. It has two contributions: Fuzzy LSTM CRF and Tie-or-Break scheme.

Tie-or-Break Scheme In a sentence, predict whether two words should be tied together to form one phrase or broken apart. Now, between every to 'breaks' you have a potential named entity. Predict its type ('None' being the type that it's not a named entity).

Training data for these phrases comes from another paper of the same group (Shang et al., 2018a) which uses unsupervised methods to extract interesting phrases from a large corpus.

Fuzzy LSTM CRF Have to read about CRF and how they work.

1. Neither BERT nor ELMO use CRF and both report SOTA results on NER.
2. Need to check if CRFs work for domain specific case? If so, why?

They also introduce a training mechanism which models the noise in supervision. Let v_i be the vector for a phrase (constructed using concatenation of beginning and ending word). Pass it through a softmax layer to get

$$p(t_j|v_i) = \frac{e^{t_j^T v_i}}{\sum_k e^{t_k^T v_i}}$$

Usually, you would now take a cross-entropy loss. But since a word can belong to multiple types, define

$$\hat{p}(t_j|v_i) = \frac{\delta(s_i \in t_j) e^{t_j^T v_i}}{\sum_k \delta(s_i \in t_k) e^{t_k^T v_i}}$$

where δ is a boolean function which checks whether the span s_i is ever marked with type t_j in distant supervision.

Now take cross-entropy loss between p and $\hat{p}$.

The only difference between $\hat{p}$ and p is that to get $\hat{p}$ you mask the values of p that are never assigned in distant supervision.

Essentially this punishes the model if it gives any weight to those values and allows the model to learn whatever it wants for the probable types.

[[Maybe there's something wrong here.]] But if the above is true, the model never gets to see the true type in the current context. How will the model learn to disambiguate?

2.2 Character Level Language Modeling

1. Character level features are important for NER. For instance, first letter being capitalized implies a proper noun, names of places in North India typically end in '-pur' (like Jaipur, Raipur) etc.

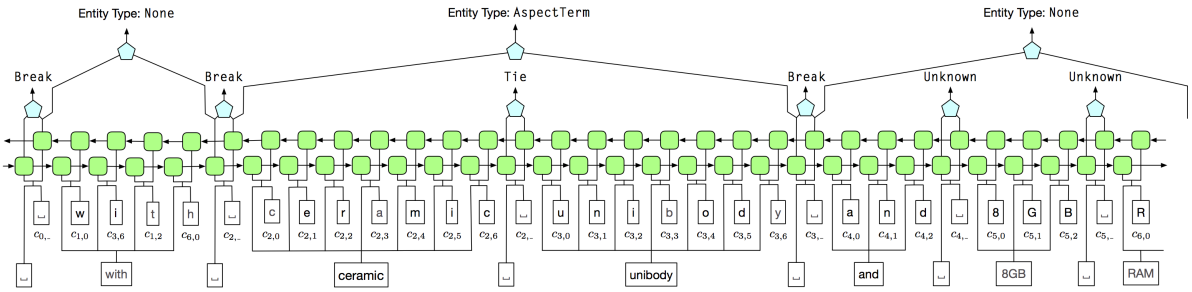


Figure 1: Overview of the Tie-or-Break scheme used in AutoNER.

2. For char-level LM, sequence lengths become too large to be directly fed into a Transformer Network. Naive solution is to break a sequence down into shorter pieces but then you lose context. One way to retain context is to keep a *memory* and use that to remember earlier parts of a sequence (Dai* et al., 2019).

3. I struggled with the code of this paper. Decided to first test the hypothesis on standard BERT and if there is potential, use it for such a character-level language model as well.

3 Proposed Method

Syntactic-BERT solves all three major challenges.

1. Using transfer learning, we avoid the problem of lack of a huge corpus for getting contextualized embeddings.

2. We use additional low-level, syntactic tasks which provide low level supervision and force the model to learn syntactic information. This syntactic information like POS tags can be used by the model to predict Named Entities (Shang et al., 2018b) [[Also cite POS paper]].

3. Using contextualized embeddings should ideally learn to disambiguate the types based on the context.

4 Experiments

5 Discussion

1. Preliminary experiments suggest that training BERT from scratch on a small corpus (20K sentences) is bad. This was expected but I was hoping it would perform slightly better because the vocabulary size of this corpus is only around 10K.

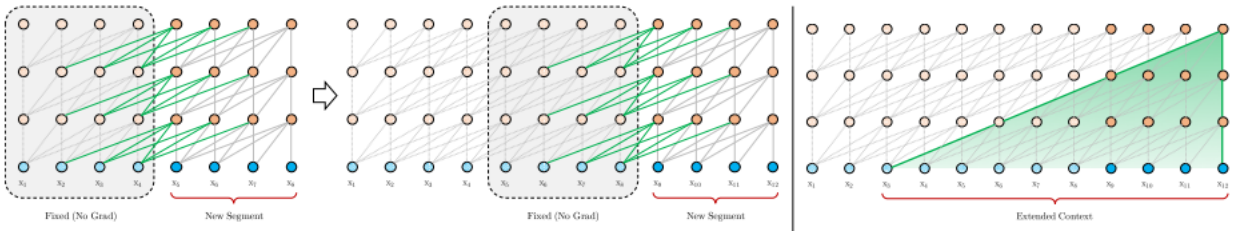
2. Next in performance is Pretrained BERT without any language modeling fine tuning. Best so far (apart form SOTA) is BERT with language model fine tuning to domain specific corpus.

References

Zihang Dai*, Zhilin Yang*, Yiming Yang, William W. Cohen, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Language modeling with longer-term dependency.

Jingbo Shang, Jialu Liu, Meng Jiang, Xiang Ren, Clare R. Voss, and Jiawei Han. 2018a. Automated phrase mining from massive text corpora. *IEEE Transactions on Knowledge and Data Engineering*, 30:1825–1837.

Jingbo Shang, Liyuan Liu, Xiaotao Gu, Xiang Ren, Teng Ren, and Jiawei Han. 2018b. Learning named entity tagger using domain-specific dictionary. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2054–2064. Association for Computational Linguistics.



(a) Training phase. (b) Evaluation phase.

Figure 2: Illustration of the Transformer-XL model with a segment length 4.

is produced (schematically) as follows,

$$\widetilde{\mathbf{h}}_{\tau+1}^{n-1} = [\mathrm{SG}(\mathbf{h}_{\tau}^{n-1}) \circ \mathbf{h}_{\tau+1}^{n-1}], \qquad \text{(extended context)}$$

$$\mathbf{q}_{\tau+1}^{n}, \mathbf{k}_{\tau+1}^{n}, \mathbf{v}_{\tau+1}^{n} = \mathbf{h}_{\tau+1}^{n-1}\mathbf{W}_{q}^{\top}, \widetilde{\mathbf{h}}_{\tau+1}^{n-1}\mathbf{W}_{k}^{\top}, \widetilde{\mathbf{h}}_{\tau+1}^{n-1}\mathbf{W}_{v}^{\top}, \qquad \text{(query, key, value vectors)}$$

$$\mathbf{h}_{\tau+1}^{n} = \text{Transformer-Layer}\left(\mathbf{q}_{\tau+1}^{n}, \mathbf{k}_{\tau+1}^{n}, \mathbf{v}_{\tau+1}^{n}\right). \qquad \text{(self-attention + feed-forward)}$$

Figure 2: Overview of transformer-xl which uses extended context to solve the problem of sending large sequences through Transformers.