Wide & Deep Learning for improving Named Entity Recognition via Text-Aware Named Entity Normalization

Ying Han[1], Wei Chen[1], Xiaoliang Xiong[*2],Qiang Li[3], Zhen Qiu[3], Tengjiao Wang[1]
[1]Key Lab of High Confidence Software Technologies (MOE), School of EECS, Peking University, Beijing, China
[2]School of EECS, Peking University, Beijing, China
[3]State Grid Information and Telecommunication Group, Beijing, China
{ying.han, pekingchenwei, xxl, tjwang}@pku.edu.cn, {qiuzhen,liqiang}@sgitg,sgcc.com.cn

Abstract

Multiple representations of the same entity often appear together in the results of named entity recognition(NER), such as "the United States" and "the U.S.", named entity normalization(NEN) was introduced to normalize synonymy of names to the concepts they refer to. Traditionally, NER and NEN are conducted as a pipelined way, which ignores the relevance between NER and NEN, so that the effective information of NEN cannot be fed back to NER to help it. Inspired by wide & deep learning in recommender systems, we proposed a model named TANER which combined the advantages of deep neural networks to learn implicit high-order contextual features for NER, and the advantages of linear models to memory explicit low-order textual features for NEN, thus TANER conducted NER and NEN simultaneously, and improved NER with the help of NEN. In addition, unlike other NEN methods that rely on knowledge bases or dictionaries, TANER is able to normalize newly defined entities in the text. Experiments on true dataset showed that TANER significantly improves the recall rate, and achieved promising results compared to the state-of-the-art approaches.

KEYWORDS wide & deep, named entity recognition, named entity normalization

Introduction

Named entity recognition(NER), which recognizes entities with specific meaning such as names of people, places and organizations in the text, is an essential task of information extraction, but due to the rich expression of natural language, the same entity concept may have variant mentions, for example, "the United States" and "the U.S.". These synonyms often appear in the results of NER, which lead to information redundancy and makes the result of NER difficult to use directly.

In the field of natural language processing(NLP), the traditional solution to the above problem is to append the task of named entity normalization(NEN) after the task of NER, normalizing synonymy of names to the concepts they refer to with the help of knowledge bases. The process of this two-step pipelined way was shown in Figure 1(a). It has two main drawbacks: (1) The pipelined way makes the result of NEN cannot feedback to NER. In fact, NEN can benefit information extraction tasks(Khalid, Jijkoun, and De Rijke 2008) (2) NEN relies on knowledge bases cannot adapt to the situation where the entity does not exist in knowledge bases. However, in fact, new entities often emerge in the news text.

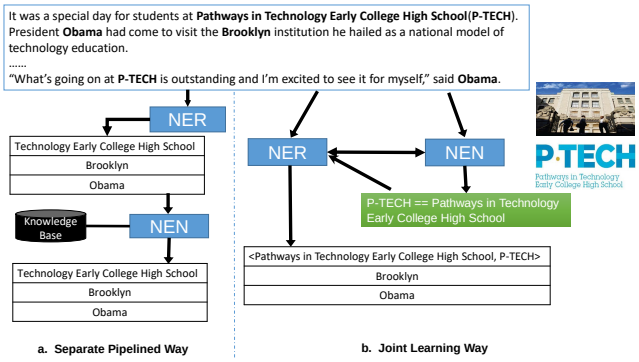

Figure 1: separate way compared to joint learning way

For example, as a piece of news text shown in Figure 1, Obama visited a Brooklyn institution named "Pathways in Technology Early College High School", also referred as "P-TECH". The traditional separate way conduct NER and NEN in a pipeline, first NER then NEN. The state-of-the-art NER tool-Stanford NER (Finkel, Grenager, and Manning 2005) gave the NER result in Figure 1(a), which failed to recognize the entity "Pathways in Technology Early College High School" and "P-TECH", so the error propagated to the following task NEN causing it to miss the entity. Even if NER fortunately recognized "Pathways in Technology Early College High School" and "P-TECH", giving correct results to NEN, NEN based on existing knowledge bases can't determine whether "Pathways in Technology Early College High School" and "P-TECH" refer to the same entity when "P-TECH" first appeared in the news that reported its establishment, because it did not exist in knowledge bases.

To address this issue, we proposed a joint learning model named TANER that conducted NER and NEN simultaneously, so that the result of NEN can be feedback to NER. Figure 1(b) shows the logic of our proposed TANER. The NEN module of TANER learned from the text that "Pathways in Technology Early College High School" and "P-

*corresponding author

TECH" are a pair of mention that refers to the same concept, and shared this information with the NER module that helps NER to extracted the two entity mentions. Inspired by the classical method (Cheng et al. 2016) wide&deep in recommender systems, TANER combines the generalization of deep neural networks to learn the implicit contextual features for NER and the memorization of the linear model to memory the explicit features for NEN, and improves NER via text-aware NEN. The NEN module of TANER was rule-based so that it enhanced the deep neural networks with general knowledge, and it makes TANER can recognize and normalize new entities from definitions in the text.
Our main contributions could be concluded as follows:

- We proposed TANER, a wide & deep model that combines deep neural network and the linear model for joint learning NER and NEN, enables improving NER via text-aware NEN. TANER compensates for the lack of general knowledge of deep-only model by introducing human knowledge into a rule-based mention extractor for NEN, and used our proposed mention vector for encoding the result of NEN.
- We proposed a novel tagging scheme to naturally combine the NER task and the NEN task. It enabled joint training for the two tasks.
- We conduct experiments to validate TANER on our labeled financial news dataset and public CoNLL 2003 shared task English dataset. The results showed the superiority of wide&deep learning. Compared with the deep-only approaches, TANER significantly improves the recall rate by combining the linear model for memorizing the information from the text-aware NEN.

Related Work

Wide & Deep

Wide & Deep(Cheng et al. 2016) as shown in Figure &eepfig:wide &deeps the most classical model in recommendations. It was firstly used in the Google App recommender system and now is applied in all aspect related to recommendations. It was widely welcomed in industry and academia. The most advanced subsequent models in recommender systems such as PNN(Qu et al. 2016), DeepFM(Guo et al. 2017), DCN(Wang et al. 2017), NFM(He and Chua 2017) etc. are all derived from this model. The core strategy of wide & deep is to combine the strengths of memorization and generalization. Wide linear models can eectively memorize sparse feature interactions using cross-product feature transformations, while deep neural networks can generalize to previously unseen feature interactions through low dimensional embeddings.

Although wide & deep is widely known in recommendation systems, as far as we know, it has not been used in NLP.

NER

McCallum and Li proposed conditional random field (McCallum and Li 2003) to solve sequence tagging problem such as NER. Lample et al. introduced neural architecture(Lample et al. 2016) to NER. The neural architecture

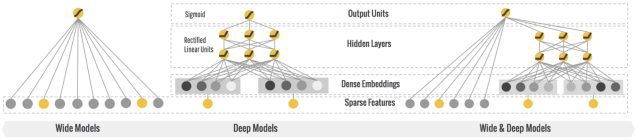

Figure 2: wide & deep model for recommendations

consisted of bidirectional LSTM and conditional random fields. Chiu and Nichols proposed LSTM-CNNs(Chiu and Nichols 2015) to realize character-level embedding. These deep neural network-based methods embed words or characters into high-dimensional vectors and then learn the interaction of these bit-wise high-order features through deep neural networks. These deep-only approaches only pay attention to the interaction of high-order features, improve the performance of NER by improving the design of neural networks, neglect the relevance of NER and other NLP tasks, and do not learn the interaction of low-order features.

Joint learning for NER and NEN

In most cases, NER and NEN are conducted in a pipeline as separate tasks. This separate way is simple, but ignores the high correlation between the NEN and NEN tasks, so that the information of the two cannot be shared. In recent years, some scholars have tried to do joint learning of entity recognition and entity normalization. Liu et al. joint inference entity recognition and normalization based on a factor graph(Liu et al. 2012). They introduced a binary random variable to join NER and NEN, whose value indicates whether the two related words across similar tweets are mentions of the same entity. Durrett and Klein presented a joint model for coreference resolution, named entity recognition and entity linking based on a structured conditional random field(Durrett and Klein 2014). These probabilistic graph-based methods depend on manually features and require a lot of feature engineering work, which are lack of generalization.

TANER model

Overview

Our proposed model named TANER as shown in Figure 3 consisted of three main components, the wide component for extracting mention pairs defined in the text for NEN and encoding them to a low-dimensional vector, the deep component for encoding implicit contextual features for NER based on bidirectional LSTM, and the joint component for merging the results of NER encoding and NEN encoding, then send them to CRF layer for decoding.

The Deep component

The Deep component The Deep component consisted of a word embedding layer and a bidirectional LSTM layer for encoding implicit contextual features for NER. The bidirectional LSTM layer consisted of a forward LSTM layer, a backward LSTM layer and a concatenate layer to concatenate the logits of the forward LSTM layer and the logits

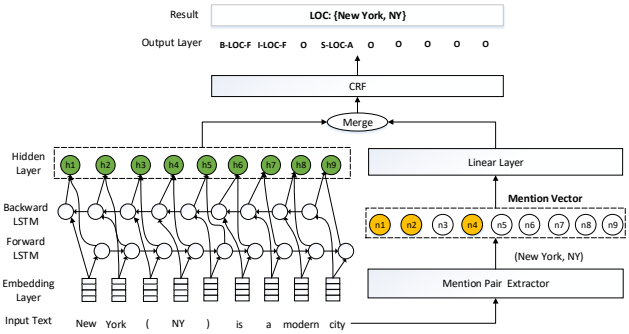

Figure 3: The architecture of TANER

of the backward LSTM layer. The word embedding layer converts each word(a word in English or a character in Chinese) to a vector. For a textual sequence with N words, it represented as $E = [e_1, ...e_t, e_t + 1, ...e_N]$, where e_t is the embedded word vector corresponding to the t_{th} word in the sequence. After word embedding layer, there are two parallel LSTM layers: forward LSTM layer and backward LSTM layer. For each word e_t, the forward layer will encode e_t by considering the contextual words information from e_1 to e_t. In the similar way, the backward LSTM layer will encode e_t based on the contextual words information from e_N to e_t. The hidden output of the deep component was the concentrated result of the forward LSTM layer and the backward LSTM layer, which marked as H_{deep}.

The Wide component

The wide component consists of a mention pair extractor, a mention vector and a linear layer.

Mention Pair Extractor A named entity may have variant mentions in the text. We introduced **mention pair** to describe it. A mention pair is a pair of entity mentions in the text that refer to the same entity concept. As shown in Figure 1, <"Pathways in Technology Early College High School", "P-TECH"> is a mention pair.
In the formal texts, such as news, the non-canonical mention of a named entity is usually introduced by a definition when it first appears. These definitions have obvious patterns. As shown in Figure 1, "Pathways in Technology Early College High School(P-TECH)" is a definition which introduced a mention pair <"Pathways in Technology Early College High School", "P-TECH">.
The mention pair extractor was designed to extract mention pairs defined in the text. It was rule-based. The conditions of rules were shown in Table 1. "F" stands for the full name and "A" stands for the abbreviation. Only patterns that satisfy both syntactic and grammatical conditions are considered valid, then mention pairs extracted by the extractor. We regarded the longer mention as the full name and the shorter mention as the abbreviation.

Mention Vector Mention Vector was introduced as m to embed the result of the mention pair extractor into a low-dimensional vector. For each word in the input textual sequence, m is a fixed-dimensional vector with a length of 7. The meaning of each dimension of the mention vector was shown in Figure 4."BIES"(Begin, Inside, End, Single) was used to represent the position information of a word in an entity mention. "FA"(Full name, Abbreviation) has the same meaning as above. "B-A" means the word was at the beginning of the abbreviation, and "I-F" means the word was inside the full name.
The initialized mental vector is a vector with each value is 0 or 1. The process of mention encoding likes one-hot. For example, the word "Pathways" in "Pathways in Technology Early College High School" was at the beginning of the full name, so the mention vector of this word was represented as [0,0,0,0,1,0,0]. If a word does not belong to any of the mentions in any of the mention pairs, then the mention vector of this word was represented as [0,0,0,0,0,0,0].
For a textual sequence with N words, the mention vector of the whole sequence was represented as M. $M = [m_1, ...m_t, m_t+1, ...m_N]$, where m_t was the mention vector corresponding to the t_{th} word in the sequence.

Table 1: conditions used in mention pair extractor

type	condition
syntactic	F(A) ; A(F) A-F ; F-A A, or F ; A, or F A...stands/short/acronym...F F, A for short
lexical	A is a substring of F F contains A except dot.

B-A	I-A	E-A	S-A	B-F	I-F	E-F

Figure 4: the meaning of each dimension of the mention vector

Linear Layer The mention vector of the whole sequence was delivered to a linear layer, The function of the linear layer is Formula 1, where H_{wide} is the result of the wide component.

$$H_{wide} = W_l \times M + b_l \quad (1)$$

Tagging Scheme

In traditional NER tagging scheme, the format of each label was "Position-EntityType", where the Position was "BIESO"(Begin, Inside, End, Single, Outside) and the types of entities were pre-defined, such as ORG(organization), PER(person), LOC(location). Similarly, the format of each label in NEN tagging scheme could be represented as "Position-NormTag", where the NormTag was "FAS"(full name, abbreviation, an entity with a single mention).
To joint training the two tasks at the same time, we designed a novel NERN tagging scheme, which combined the NER tagging scheme and the NEN tagging scheme. The for-

mat if each label in NERN tagging scheme was "**Position-EntityType-NormTag**". For example, "Pathways in Technology Early College High School" is the full name of an organization entity, so "Pathways" in this mention should be labeled as "B-ORG-F". If a word does not belong to any entity, then it should be labeled as "O".

The Joint Component

The joint component consists of a merge layer, a linear layer and a CRF layer for decodingas shown in Figure 3, this linear layer was omitted in the figure for the sake of brevity. As described in Formula 2, the result of the deep component and the result of the wide component are concatenated by the merge layer, and then did a dimensional transformation via a linear layer, then H_{in} was got. H_{in} was the input of the CRF layer.

$$H_{in} = W_T \times [H_{deep}, H_{wide}] + b_T \tag{2}$$

For a sequence of predictions, Formula 3 defined the score of CRF(Lafferty, McCallum, and Pereira 2001):

$$S(X, y) = \sum_{i=0}^{n} A_{y_i, y_{i+1}} + \sum_{i=1}^{n} P_{i, y_i} \tag{3}$$

where A is a matrix of transition such that A_i, j represents the score of a transition from the i_{th} label to the j_{th} label. y_0 and y_n are the start and end tags of a sequence, that we add to the set of possible tags. P represents the unary score which is given by H_{in}. A softmax over all possible tag sequences yields a probability for the sequence y:

$$p(y|X) = \frac{e^{s(X,y)}}{\sum_{\tilde{y} \in Y_X} e^{s(X,\tilde{y})}} \tag{4}$$

During training, we maximize the log-probability of the correct tag sequence.

$$\log(p(y|X)) = s(X, \tilde{y}) - \log(\sum_{\tilde{y} \in Y_X} e^{s(X,\tilde{y})}) \tag{5}$$

Experiments

datasets

We used two datasets, one is our manually labeled dataset-Chinese financial news dataset, which was crawled from financial websites such as Sina financials, Daily financials, Eastern Fortune Network, etc., then manually labeled for NER and NEN. The entity type we defined includes two categories, "BOND" for bonds and stocks, "ORG" for organizations. Another one is public English dataset CoNLL 2003(Tjong Kim Sang and De Meulder 2003). The details of datasets we used were shown in Table 2.

Baseline

We used CRF(Chen et al. 2006), BiLSTM, BiLTSM-CRF(Lample et al. 2016), BiLSTM-CNNs-CRF(Ma and Hovy 2016) as baselines.

Table 2: Details of datasets

Data	Type	#sequence	#Token	#Mention pair
financial news	train	225,122	6,802,737	748,301
	val	25,014	750,219	78,773
	test	62,534	1,883,837	201,571
CoNLL 2003	train	14,987	204,567	-
	val	3,466	51,578	-
	test	3,684	46,666	-

Table 3: NER results on the financial news dataset

Model	Precision	Recall	F1
TANER	**98.17%**	**95.83%**	**96.99%**
BiLSTM-CRF	97.51%	91.90%	94.62%
BiLSTM-CNNs-CRF	97.18%	92.64%	94.86%
BiLSTM	91.96%	88.15%	90.01%
CRF	97.05%	80.37%	87.93%

Hyperparameters

In the financial news experiment, we trained our networks using the back-propagation algorithm updating trained parameters on every mini batch, using Adam (Kingma and Ba 2014) with a learning rate of 0.001 and a gradient clipping of 5.0.

We used variable embedding with 400 dimensions other than pre-trained word embedding, so the word embedding updates while training.

Batch size is suggested to 128. Our experiments showed a bigger or smaller batch was worse than it.

Bidirectional LSTM cell with 300 dimensions is suitable. We have tried two layer LSTM cell, but the results show a single layer LSTM with bidirection was faster and got higher performance.

Dropout is useful and necessary to make model robust and generalized. Dropout was used twice in our model, one was in word embedding layer, one was between the connection of the merged result of NER and NEN encoding and the linear layer.

In the CoNLL 2003 experiment, we set the learning rate as 0.015, set the batch size as 10 and set the dimensions of word embedding is 120. Other settings are as same as above.

Results

We first conducted experiments on the financial news dataset. The experiments were conducted to verify the validity of our proposed TANER and compare it to other NER-only methods. The results of experiments on the financial news dataset were shown in Table 3.

Then we conducted experiments on the public dataset CoNLL 2003 shared task to verify the robustness of our proposed TANER. The dataset did not have NEN annotation. In this experiments, we used the NER tagging scheme instead of NERN tagging scheme. In order to maintain consistency, we used 'BIESO" to express the position instead of 'BIO'.

Table 4: NER results on CoNLL dataset

Model	Precision	Recall	F1
TANER	**91.14%**	90.61%	90.87%
BiLSTM-CNN	87.26%	90.23%	88.72%
BiLSTM-CNNs-CRF	91.08%	**90.84%**	**90.96%**
BiLSTM	86.12%	86.27%	86.19%
CRF	90.15%	84.73%	87.36%

The results of NER on the CoNLL 2003 dataset were shown in Table 4.

Discussions

The experiments on the financial news dataset showed that Joint learning for NER and NEN is better than NER only. NEN helped improve the performance of NER especially the recall rate, which got 3.93% gain.

The experiments on the CoNLL 2003 shared task dataset showed the robustness of TANER, the NER module of TANER can exist independently, it worked well without the NEN module. This experiment proved that NEN does not disturb NER in TANER.

Inspired by the method of wide & deep learning in recommender systems, we transferred this inspiration to NLP for the first time and proposed TANER based on the wide & deep learning. TANER combined the generalization of deep neural networks and the memorization of linear models to jointly learn NER and NEN, improving NER via text-aware NEN.

The experimental results showed that the idea of wide & deep learning is not only suitable for recommender systems, but also adapt to natural language processing. TANER is especially suitable for formal texts including definitions and new entities, such as news texts. Compared with state-of-the-art methods that rely only on deep neural networks to learn implicit contextual features, TANER combines the advantages of memorization of linear models to memory explicit features from text-aware NEN, and thus TANER can greatly improve the recall rate of NER result. At the same time, even in the corpus which is lack of definitions, TANER also achieved promising performance compared to the state-of-the-art approaches.

Acknowledgments

This work was supported by State Grid Technical Project(No. 52110418002X).

References

Chen, A.; Peng, F.; Shan, R.; and Sun, G. 2006. Chinese named entity recognition with conditional probabilistic models. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, 173–176.

Cheng, H.-T.; Koc, L.; Harmsen, J.; Shaked, T.; Chandra, T.; Aradhye, H.; Anderson, G.; Corrado, G.; Chai, W.; Ispir, M.; et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, 7–10. ACM.

Chiu, J. P., and Nichols, E. 2015. Named entity recognition with bidirectional lstm-cnns. *arXiv preprint arXiv:1511.08308*.

Durrett, G., and Klein, D. 2014. A joint model for entity analysis: Coreference, typing, and linking. *Transactions of the Association for Computational Linguistics* 2:477–490.

Finkel, J. R.; Grenager, T.; and Manning, C. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, 363–370. Association for Computational Linguistics.

Guo, H.; Tang, R.; Ye, Y.; Li, Z.; and He, X. 2017. Deepfm: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247*.

He, X., and Chua, T.-S. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, 355–364. ACM.

Khalid, M. A.; Jijkoun, V.; and De Rijke, M. 2008. The impact of named entity normalization on information retrieval for question answering. In *European Conference on Information Retrieval*, 705–710. Springer.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lafferty, J.; McCallum, A.; and Pereira, F. C. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Lample, G.; Ballesteros, M.; Subramanian, S.; Kawakami, K.; and Dyer, C. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Liu, X.; Zhou, M.; Wei, F.; Fu, Z.; and Zhou, X. 2012. Joint inference of named entity recognition and normalization for tweets. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, 526–535. Association for Computational Linguistics.

Ma, X., and Hovy, E. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf.

McCallum, A., and Li, W. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, 188–191. Association for Computational Linguistics.

Qu, Y.; Cai, H.; Ren, K.; Zhang, W.; Yu, Y.; Wen, Y.; and Wang, J. 2016. Product-based neural networks for user response prediction. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, 1149–1154. IEEE.

Tjong Kim Sang, E. F., and De Meulder, F. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, 142–147. Association for Computational Linguistics.

Wang, R.; Fu, B.; Fu, G.; and Wang, M. 2017. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*, 12. ACM.