Complex Named Entity Recognition via Deep Multi-task Learning from Scratch

Guangyu Chen[1], Tao Liu[1], Deyuan Zhang[2](✉), Bo Yu[3], and Baoxun Wang[3]

[1] School of Information, Renmin University of China, Beijing, China
{hcs,tliu}@ruc.edu.cn
[2] School of Computer, Shenyang Aerospace University, Shenyang, China
dyzhang@sau.edu.cn
[3] Tricorn (Beijing) Technology Co., Ltd, Beijing, China
{yubo,wangbaoxun}@trio.ai

Abstract. Named Entity Recognition (NER) is the preliminary task in many basic NLP technologies and deep neural networks has shown their promising opportunities in NER task. However, the NER tasks covered in previous work are relatively simple, focusing on classic entity categories (Persons, Locations, Organizations) and failing to meet the requirements of newly-emerging application scenarios, where there exist more informal entity categories or even hierarchical category structures. In this paper, we propose a multi-task learning based subtask learning strategy to combat the complexity of modern NER tasks. We conduct experiments on a complex Chinese NER task, and the experimental results demonstrate the effectiveness of our approach.

Keywords: Complex named entity recognition · Multi-task learning
Deep learning

1 Introduction

Nowadays, many basic NLP technologies have been utilized in the newly-emerging application scenarios, among which the Named Entity Recognition (NER) models are believed to be of paramount importance for locating the essential information slots and predicting user intentions in the task-oriented AI products, especially the ones with speech interface such as conversational agents[1], smart speakers[2].

This work is supported by visiting scholar program of China Scholarship Council and National Natural Science Foundation of China (Grant No. 61472428 and No. U1711262). The work was done when the first author was an intern in Tricorn (Beijing) Technology Co., Ltd.

Bo Yu is currently working in Baidu, Inc.

[1] https://dueros.baidu.com/.

[2] https://developer.amazon.com/alexa.

Compared to traditional NER tasks focusing on the classic entity categories involving names of persons, locations, organizations, etc. [4,11,21], the NER modules for the task-oriented scenarios of new AI products are facing the more complex situation with more informal entity categories or even hierarchical category structures (Fig. 1 gives an example). The difference is basically brought by the requirements of practical task-oriented systems which take spoken language as the interactive interface, since it is much more difficult to parse spoken language sentences to detect slots or predict intentions. More importantly, in the task-oriented NER scenarios, the complex entity categories significantly increase the difficulty of human annotation. Consequently, the amount of high-quality annotated datasets generally can not be guaranteed, which blocks the NER models from achieving satisfying performance with no doubt.

For the task-oriented NER models which take spoken languages as input, the limitation of the amount of human-annotated data is a severe problem that should be handled first. From the perspective of the principle of NER, this task objectively keeps the correlation with the other NLP tasks such as word segmentation, part-of-speech (POS) tagging, etc. More importantly, the NER model is very possible to benefit from the performance improvements on such tasks, since they are logistically the basis of the NER task. Consequently, it is fairly reasonable to conduct the multi-task learning procedure upon a basic shared learnable component, which can be updated in accordance with each training iteration of each task. This architecture becomes more practicable due to the natural characteristics of deep learning models, since the parameter sharing and fine-tuning mechanisms are suitable for building trainable shared layers providing implicit representations of linguistic knowledge.

In this paper, we propose a learning strategy to combat the complex NER task by firstly dividing the NER task into fine-grained subtasks (according to domain affiliation) then integrating these subtasks into the multi-task learning process and finally training them from scratch. A key aspect of our idea is that these fine-grained subtasks will get better performance without the disturbance coming from other domains.

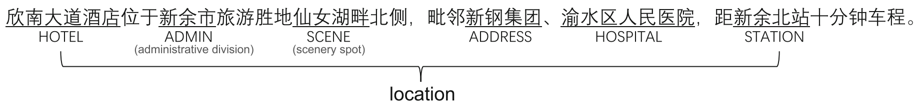
欣南大道酒店位于新余市旅游胜地仙女湖畔北侧，毗邻新钢集团、渝水区人民医院，距新余北站十分钟车程。
HOTEL
ADMIN
(administrative division)
SCENE
(scenery spot)
ADDRESS
HOSPITAL
STATION
location

Fig. 1. An example of the complex NER task facing by new AI products. All the six entity names in this figure belong to the same domain of 'Location' in traditional NER tasks. It demonstrates the complex NER is a challenging learning problem. On the one hand, the model not only needs to classify which domain (Person, Location, Organization) the named entity belongs to, but also needs to dive down and get the correct fine-grained category. On the other hand, the boundaries between these categories may be blurry. For example, "Lijiang" is an administrative division, but in a specific context it may refer to the "Old Town of Lijiang" which is a scenery spot.

The remaining part of this paper is organized as follows. Section 2 surveys the related studies. Our proposed methodology is presented in Sect. 3. The experimental results are given and analyzed in Sect. 4. Finally, we conclude our work in Sect. 5.

2 Related Work

NER is a challenging learning problem considering the amount of supervised training data available and the few constraints on the kinds of words that can be names. As a result, orthographic features and language-specific knowledge resources, such as gazetteers, are widely used to improve the NER tasks' performance [6,16,22]. However, this practice ruins the possibility of training the NER task from scratch, since the language-specific resources and features are costly to obtain when facing new language or new domains.

Multi-task learning (MTL) has led to success in many applications of language processing [5] by sharing representations between related tasks [1,3]. Compared with single-task learning, the architecture commonly used in MTL has shared bottom layers and several individual top layers for each specific task. By jointly (or iteratively) training on related tasks, the representation capacity of the shared layers are enhanced. On MTL for sequence labeling tasks, Ando [1] proposed a multi-task joint training framework that shares structural parameters among multiple tasks, and improved the performance on various tasks including NER. Collobert [6] presented a task independent convolutional network and employed multi-task joint training to improve the performance of chunking. The BiLSTM-CRFs neural architecture proposed by Lample [15] achieved state-of-the-art results [23]. These previous works exclusively focused on the traditional named entity recognition, however, the named entity categories are much more complicated in practical applications nowadays. We argue that it would be beneficial to take apart the original complex NER task into fine-grained subtasks. In the training process, each subtask is trained independently, while in the testing process, these subtasks will be executed simultaneously and the results of these subtasks will be integrated to produce the final result. This perspective also makes it easy to add these subtasks into MTL's iterative training procedure with an end-to-end manner. To the best of our knowledge, the work that is closet to ours is [19], which focuses on domain adaptation with a simple NER classification category, but their work does not explore the possibility of applying it to the more complicated NER task.

3 Multi-task Learning Architecture for NER

In this section, we first provide a brief description of LSTMs used in our architecture, and then present a baseline model for single sequence tagging tasks based on Bidirectional LSTM (Bi-LSTM) recurrent units. Finally we elaborate on the iteration training procedure of MTL and the details about NER subtasks training strategy.

3.1 Basic Bi-LSTM Structure

Recurrent neural networks are a sort of neural networks taking the input data in the form of vector sequence $(x_1, x_2, \ldots, x_n)$ and generating the sequence output $(h_1, h_2, \ldots, h_t)$ which represents the context information encoded in every step of the input vectors. It has been shown the traditional RNNs tend to be biased toward the recent tokens, which caused the failure of learning long-term dependencies [2]. Long Short-term Memory networks (LSTMs) [12] are put forward to combat the above problem by applying additional gates to control the proportion of input given to the memory cell, and the proportion of the previous state to forget. There are several architectures of LSTM units. We used the following implementations [9]:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{1}$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{2}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{3}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{4}$$

$$h_t = o_t \odot \tanh(c_t) \tag{5}$$

where $\odot$ denotes element-wise product and σ is the sigmoid logistic function (defined as $\sigma = 1/(1 + e^{-x})$).

For an input sequence $(x_1, x_2, \ldots, x_n)$ containing n characters, the output $\overrightarrow{h_t}$ encodes the left context information of character t in the input sequence, which means the network models the input sentence only in forward direction. It is usually helpful by adding another LSTM reading reversely and concatenating both outputs as the final output $h_t = [\overrightarrow{h_t}, \overleftarrow{h_t}]$. This is referred as the bidirectional LSTM [10]. It has been demonstrated that this bidirectional architecture will improve the performance in many sequence labeling tasks, thus it has become the common building component in such tasks and is extensively used in many sequence tagging tasks such as Chinese word segmentation (SEG), POS tagging and NER.

3.2 Single Task Training

We start with a basic neural network (NN) model for training single sequence tagging task in this subsection, then move on to the structure for iterative training in Sect. 3.3.

The NN model for single sequence tagging tasks can be seen in Fig. 2. For an input character sequence $X = \{w_1, w_2, \ldots, w_n\}$, it can be transformed to a sequence of character vectors with length n by performing embedding lookup operation on each character w_i. Then, this vector sequence will be fed to the Bi-LSTM layer. Finally, a fully connected layer and a softmax layer will be used to produce the final tagging result. This structure can be easily applied to many sequence tagging tasks in an end-to-end manner and have the capacity of being integrated into the MTL environment.

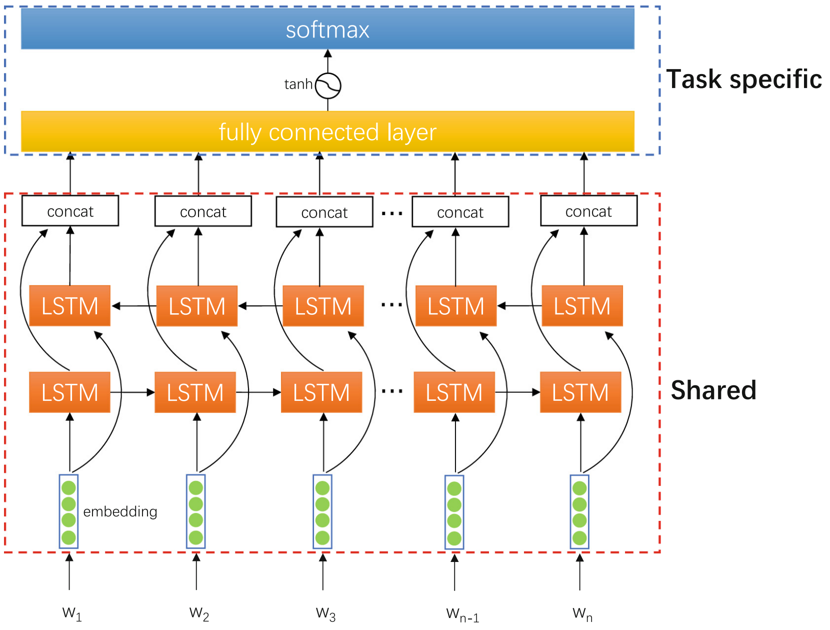

Fig. 2. The architecture of our Bi-LSTM recurrent network for single sequence tagging task. Character embeddings successively pass though a bidirectional LSTM layer, a fully connected layer and a softmax layer to generate the predicted outputs. This architecture can be employed directly in the MTL environment by simply dividing it into shared parts and task specific parts which marked by the red box and blue box above. Figure 3 gives the further detail in MTL aspects.

We employ mini-batch Adam optimizer [13] to train our neural network in an end-to-end manner with back propagation. In order to train the model with batched inputs, input sentences will be tailed to a fixed length before the embedding lookup operation, thus the lengths of input sentences for the Bi-LSTM layer are equal. When performing evaluation for the NER task, the Viterbi algorithm [8] is used to decode the most probable tag sequence.

3.3 Multi-Task Training Scheme

Multi-task training is desirably leveraged to boost model performance, since different sequence tagging tasks (such as SEG, POS and NER) in the same language share language-specific characteristics and should learn similar underlying representation. To apply the basic NN model mentioned above to the multi-task mechanism, we divide the parameters (W_t) of single task t into two sets (task specific set and shared set):

$$W_t = W_{tshare} \cup W_{tspec} \tag{6}$$

according to different roles of the parameters. We share all the parameters below the fully connected layer including character embeddings to learn language-specific characteristics for all the tasks, and the rest are regarded as task specific parameters. These two sets are marked out in Figs. 2 and 3 respectively.

There are two patterns commonly used in MTL, one of which is training different tasks jointly (optimize more than one loss function at a time), for example, training the SEG task and the NER task simultaneously by maximizing a weighted joint objective [18]:

$$Loss_{joint} = \lambda Loss_{SEG} + Loss_{NER} \tag{7}$$

where λ trades off between better segmentation or better NER. However, this approach needs additional modifications to the model architecture and it is hard to combine too many tasks. Thus, we adopt the iterative training mechanism of MTL. For each iteration, all the tasks are trained sequentially. For each task t, we first load previous trained parameters for initialization. The shared parameter set is initialized with the latest W_{tshare} if there exists one. The task specific parameter set is initialized with the latest W_{tspec} of the same task if there exists one. Otherwise, we make a random initialization of parameters. Then, we perform gradient descent steps to update model parameters W_t until the model performance tends to be stable, and then switch to training the next task. In this way, the models for all the tasks are gradually optimized through the iterative training process and the performance for each task is improved. We show the effect of this iterative training mechanism in the experiment.

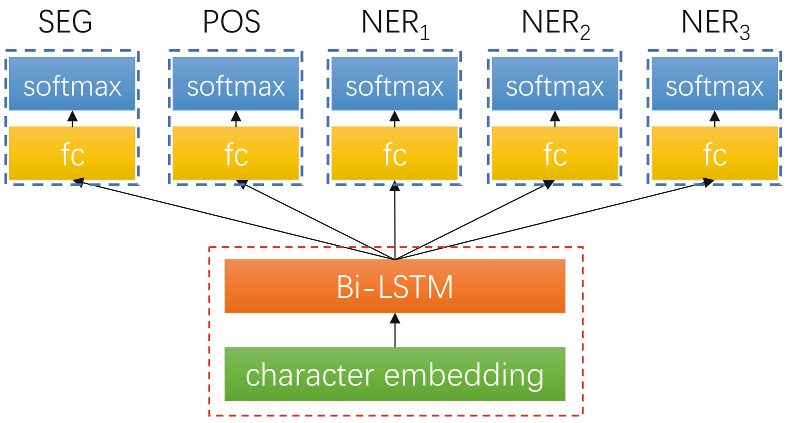

Fig. 3. The architecture used for NER subtask training. In the MTL training process, the parts marked by blue boxes are task-specific and the parts located in the red box is shared by all the tasks. To combat the complexity of the NER task, we divide the NER task into three fine-grained subtasks (NER_{Travel}, $NER_{Shopping}$, $NER_{Entertainment}$) according to the domain affiliation.

3.4 NER Subtasks Training Scheme

To reduce the complexity of the complex NER task, we divided it into three fine-grained subtasks (NER_{Travel}, $NER_{Shopping}$, $NER_{Entertainment}$) by splitting the dataset according to the domain affiliation (Sect. 4 presents the detail of data processing). Since the training data for each NER subtask is isolate, it is possible to treat them as independent sequence tagging tasks and combine them into the iterative training process of MTL. This design captures two intuitions. First, each fined-grained subtask only focuses on a single domain and will get better performance since the labels needed to predict are downsized, without the disturbance of other domains. Second, since SEG and POS tasks may bring fundamental and latent contributions to NER task, it could be helpful to enhance the representation capacity of shared layers by training with SEG and POS tasks. After adding these tasks, the final architecture is shown in Fig. 3.

During evaluation, we employ these NER models on the same test set, collect the results and perform merging operations:

(1) Ideally, these models' predictions will be non-intersected, for example the named entity "Beijing Hotel" belongs to the Travel domain, thus it will only get the named label from NER_{Travel}. The other two subtasks will neglect this entity, thus this merging operation can be performed by simply merging theses models' predictions.
(2) However, due to the complexity of our NER task, there is a possibility that these models may given different tagging results for the same entity which can not be merged, e.g. for named entity "Shaolin Temple", NER_{Travel} marks it as SCENE (scenery spot) while $NER_{Entertainment}$ marks it as FILM. And in more complex situations, these results can be overlapping. In these cases, the tagging result with higher sentence probability (generated by the Viterbi decoding operation) will be chosen as the final result. We find there are rare situations (about 0.5% of total entities) that need this further merging operation from experiments, which demonstrates theses subtasks have the capacity of concentrating on entities of their own domains.

4 Experiments and Analysis

In this section, we first introduce the datasets used in SEG, POS and NER tasks. Then we present the details of tagging schemes and embedding settings. Finally we give the results and the analysis of our experiment.

4.1 Datasets

We consider three tasks: sentence segmentation, part-of-speech tagging, and complex named entity recognition. All these tasks are in Chinese, and each task corresponds to a dataset. Chinese Treebank 9.0[3] datasets are used for SEG and

[3] https://catalog.ldc.upenn.edu/ldc2016t13.

Table 1. The NER tags for each domain and their proportions in total named entities.

Domain	Tag	Entity number	Percentage (%)
Travel	CATER	88944	37.1
	HOTEL	9186	3.8
	SCENE	36641	15.3
Shopping	PROD_BRAND	17934	7.5
	PROD_TAG	44138	18.4
Entertainment	TV	19052	7.9
	FILM	16125	6.7
	MUSIC	5032	2.1
	ENT_OTHER	2822	1.2

POS tagging, and each contains 13.2k training sentences. An internal dataset crawled from Chinese forums and Chinese News websites is used for complex NER. It is preprocessed with data cleaning, and labeled with nine entity types[4] covering domains of travel, shopping and entertainment. The NER dataset contains 189.7k sentences, and each sentence only belongs to one domain. Entity distribution of this dataset is shown in Table 1. Note that the datasets for SEG (POS) and NER are drawn from different distributions, yet share many commonalities and still make contributions to each task. To combat the complexity of the complex NER task, we propose the subtask strategy. It consists of the following steps:

(1) Hold out 10% of the NER dataset for testing. The remaining part NER_{remain} is used for further process.
(2) Split NER_{remain} into three subsets according to the domain affiliation, since each sentence belongs to one domain in our NER dataset.
(3) Balance the dataset of each subtask by equally adding sentences that drawn from the other two datasets with the out-of-domain labels transformed to "O". The motivation is to prevent biased model for each subtask if the training data is only drawn from one domain.
(4) For each subset, split the data using an 8:1 for training and validation.

Table 2 shows the tags and volumes of datasets for all subtasks after processing with the subtask strategy.

4.2 Tagging Schemes

The goal of NER is to assign a named entity label to each character of a sentence. In our NER task, we used a "BMESO" tagging format, where B-label and E-label represent the beginning and ending of a named entity respectively, M-label

[4] CATER,HOTEL,SCENE,PROD_TAG,PROD_BRAND,FILM,MUSIC,TV, ENT_OTHER.

Table 2. Tags contained in each subtask and the sentence volume of each dataset. Note that these three tasks share the same test set and the domain-unrelated tags will be transformed to tag "O" (for Others) during testing.

Subtask	Tag	Train set	Validation set	Test set
NER_{Travel}	CATER HOTEL SCENE	151751	18968	18967
$NER_{Shopping}$	PROD_BRAND PROD_TAG	77071	9633	18967
$NER_{Entertainment}$	ENT_OTHER FILM MUSIC TV	61456	7681	18967

represents the remaining section of an entity except both ends, and the S-label represents a singleton entity. For the character that do not belongs to any entity type, we use "O" as its label. Sentences can also be tagged in a more concise format as "BMO" if we throw away the information about singleton entity and entity ending. However, works in [7,20] showed that using a more expressive tagging scheme like "BMESO" improves the model performance marginally. We employ this expressive format in our experiments.

4.3 Preprocessing and Pretrained Embeddings

In the iterative training process, to make it possible for sharing the character embedding layer between different tasks, we use the same vocabulary of size 10571 covering most of the Chinese characters and punctuations. During training, the numbers and English words in the corpus will be replaced with "_DIGIT" and "_ALPHABET", and for the characters out of this vocabulary, we use "_UNK" to represent them. To initialize the lookup table, we use pretrained character embeddings which are trained by the word2vec [17] tool. We observe improvements using pretrained embeddings compared with randomly initialized ones and finally set the embedding dimension to 256 for improving the model's learning capability.

4.4 Results and Analysis

We consider three baselines, the first baseline is a linear chain Conditional Random Field (CRF) [14] commonly used in sequence labeling task. The second baseline is the single NN model mentioned in Sect. 3.2. The last baseline integrates the single NN model into the MTL pattern mentioned in Sect. 3.3. Compared with the proposed fine-grained model, the difference is that it takes the complex NER task as a whole. Table 3 presents the test results for complex NER in terms of precision, recall and F1 score. Table 4 gives the details of F1 results in each domain. Table 5 shows the F1 scores achieved by the proposed fine-grained model in each training iteration.

As shown in Table 3, the Single Model (Method 2) gets a lower precision but achieves a higher recall, which results in a marginal increase (+0.72) in F1 compared to the CRF model. For the MTL based models (Method 3 and 4), both

Table 3. Statistics of NER results on test dataset. The methods 3 and 4 use the iterative training scheme of MTL. All these models haven't used external labeled data such as gazetteers and knowledge bases.

Method	Precision (%)	Recall (%)	F1 (%)
1 CRF	82.78	74.75	78.56
2 Single Model	80.59	78.00	79.28
3 Single Model with MTL	82.04	78.78	80.37
4 Fine-grained Model with MTL	**83.56**	**81.57**	**82.55**

Table 4. F1 results in domains of *Travel*, *Shopping* and *Entertainment*. The fine-grained model (Method 4) beats all the others in all domains, and the improvements are more than 2% compared with the integrated model (Method 3). It demonstrates that this fine-grained training scheme eliminates the interference from the other domains which helps the model learn better.

Method	F1 (%)		
	NER_{Travel}	$NER_{Shopping}$	$NER_{Entertainment}$
1 CRF	81.31	79.29	68.36
2 Single Model	82.53	78.71	69.52
3 Single Model with MTL	83.53	79.77	71.17
4 Fine-grained Model with MTL	**85.81**	**81.82**	**73.40**

Table 5. The evaluation F1 achieved by the Fine-grained Model with MTL. The last row displays the max improvement gained in iteration processes, which shows the MTL scheme helps improve models' performance.

Iteration	F1 (%)				
	SEG	POS	NER_{Travel}	$NER_{Shopping}$	$NER_{Entertainment}$
1	95.44	89.86	85.87	81.89	72.12
2	95.51	90.07	85.79	82.10	72.84
3	95.66	**90.10**	86.17	82.17	73.42
4	**95.74**	90.01	**86.18**	**82.94**	**73.52**
Improvement	0.3	0.24	0.31	1.05	1.4

of them improve performances over the Single Model. Knowing that the SEG and POS tasks help the model better learn effective representations, the Single Model with MTL (Method 3) gives us an increase of +1.81 and the Fine-grained Model gives us the biggest improvement of +3.99, in terms of F1 score. In the MTL settings, the Fine-grained Model outperforms the Single Model with MTL, which demonstrates that the subtask training strategy does benefit the performance of complex NER. This can be further confirmed by the experimental results in

Table 4, where the proposed fine-grained model beats all the other baselines in all domains.

Considering the results in Table 5, we find that SEG, POS and NER_{Travel} get small improvements compared with $NER_{Shopping}$ and $NER_{Entertainment}$. The reason is as follows. The tasks of SEG and POS are much easier than NER, thus their top limits are easy to achieve and the room for further improvement is also limited. For NER_{Travel}, the data volume of this task is the largest in the three subtasks (nearly twice the data volume of the others), which means it can be trained more sufficiently thus leaving less room for improvement. Among these subtasks, $NER_{Entertainment}$ gets the worst F1 score, which is mainly because the film entities and TV entities have similar contexts, and it is hard to discriminate them without the help of knowledge bases. However, $NER_{Entertainment}$ achieves the highest F1 improvement in the MTL process, which verifies the efficiency of MTL framework.

5 Conclusions

In this paper, we have proposed a learning strategy to combat the complex NER task by first dividing the NER task into fine-grained subtasks (according to domain affiliation), then integrate these subtasks into a multi-task learning process and finally train from scratch. The experimental results show that these fine-grained subtasks will get better results without the disturbance from other domains.

In this work, we mainly focus on the complex NER tasks which only cover three domains. In the future, we will test this strategy on NER tasks with more domains. Furthermore, it will be interesting to apply our approach to other languages.

References

1. Ando, R.K., Zhang, T.: A framework for learning predictive structures from multiple tasks and unlabeled data. J. Mach. Learn. Res. **6**, 1817–1853 (2005). http://dl.acm.org/citation.cfm?id=1046920.1194905
2. Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE Trans. Neural Netw. **5**(2), 157–166 (1994). https://doi.org/10.1109/72.279181
3. Caruana, R.: Multitask learning. Mach. Learn. **28**(1), 41–75 (1997)
4. Chieu, H.L., Ng, H.T.: Named entity recognition: a maximum entropy approach using global information. In: Proceedings of the 19th International Conference on Computational Linguistics, COLING 2002, vol. 1, pp. 1–7. Association for Computational Linguistics, Stroudsburg (2002)
5. Collobert, R., Weston, J.: A unified architecture for natural language processing: deep neural networks with multitask learning. In: Proceedings of the 25th International Conference on Machine Learning, ICML 2008, pp. 160–167. ACM, New York (2008). https://doi.org/10.1145/1390156.1390177, https://doi.acm.org/10.1145/1390156.1390177

6. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. J. Mach. Learn. Res. **12**, 2493–2537 (2011). http://dl.acm.org/citation.cfm?id=1953048.2078186
7. Dai, H.J., Lai, P.T., Chang, Y.C., Tsai, R.T.H.: Enhancing of chemical compound and drug name recognition using representative tag scheme and fine-grained tokenization. J. Cheminform. **7**(Suppl 1), S14–S14 (2015). https://doi.org/10.1186/1758-2946-7-S1-S14, http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4331690/. 1758-2946-7-S1-S14[PII]
8. Forney, G.D.: The viterbi algorithm. Proc. IEEE **61**(3), 268–278 (1973). https://doi.org/10.1109/PROC.1973.9030
9. Gers, F.A., Schmidhuber, J., Cummins, F.: Learning to forget: continual prediction with LSTM. Neural Comput. **12**, 2451–2471 (1999)
10. Graves, A., Schmidhuber, J.: Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Networks **18**(5), 602–610 (2005). https://doi.org/10.1016/j.neunet.2005.06.042, http://www.sciencedirect.com/science/article/pii/S0893608005001206. iJCNN 2005
11. Grishman, R., Sundheim, B.: Design of the MUC-6 evaluation. In: Proceedings of the 6th Conference on Message Understanding, MUC6 1995, pp. 1–11. Association for Computational Linguistics, Stroudsburg (1995)
12. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735
13. Kingma, D., Ba, J.: Adam: A Method for Stochastic Optimization (2014)
14. Lafferty, J.D., McCallum, A., Pereira, F.C.N.: Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: Proceedings of the Eighteenth International Conference on Machine Learning, ICML 2001, pp. 282–289. Morgan Kaufmann Publishers Inc., San Francisco (2001). http://dl.acm.org/citation.cfm?id=645530.655813
15. Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C.: Neural architectures for named entity recognition. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 260–270. Association for Computational Linguistics (2016)
16. Lin, D., Wu, X.: Phrase clustering for discriminative learning. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2, ACL 2009, pp. 1030–1038. Association for Computational Linguistics, Stroudsburg (2009). http://dl.acm.org/citation.cfm?id=1690219.1690290
17. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. CoRR abs/1301.3781 (2013). http://arxiv.org/abs/1301.3781
18. Peng, N., Dredze, M.: Learning word segmentation representations to improve named entity recognition for chinese social media. CoRR abs/1603.00786 (2016). http://arxiv.org/abs/1603.00786
19. Peng, N., Dredze, M.: Multi-task domain adaptation for sequence tagging. In: Proceedings of the 2nd Workshop on Representation Learning for NLP, pp. 91–100. Association for Computational Linguistics (2017). http://aclweb.org/anthology/W17-2612
20. Ratinov, L., Roth, D.: Design challenges and misconceptions in named entity recognition. In: Proceedings of the Thirteenth Conference on Computational Natural Language Learning, CoNLL 2009, pp. 147–155. Association for Computational Linguistics, Stroudsburg (2009). http://dl.acm.org/citation.cfm?id=1596374.1596399

21. Tjong Kim Sang, E.F., De Meulder, F.: 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, CONLL 2003, vol. 4, pp. 142–147. Association for Computational Linguistics, Stroudsburg (2003)
22. Turian, J., Ratinov, L., Bengio, Y.: Word representations: a simple and general method for semi-supervised learning. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, ACL 2010, pp. 384–394. Association for Computational Linguistics, Stroudsburg (2010). http://dl.acm.org/citation.cfm?id=1858681.1858721
23. Yang, Z., Salakhutdinov, R., Cohen, W.W.: Multi-task cross-lingual sequence tagging from scratch. CoRR abs/1603.06270 (2016). http://arxiv.org/abs/1603.06270