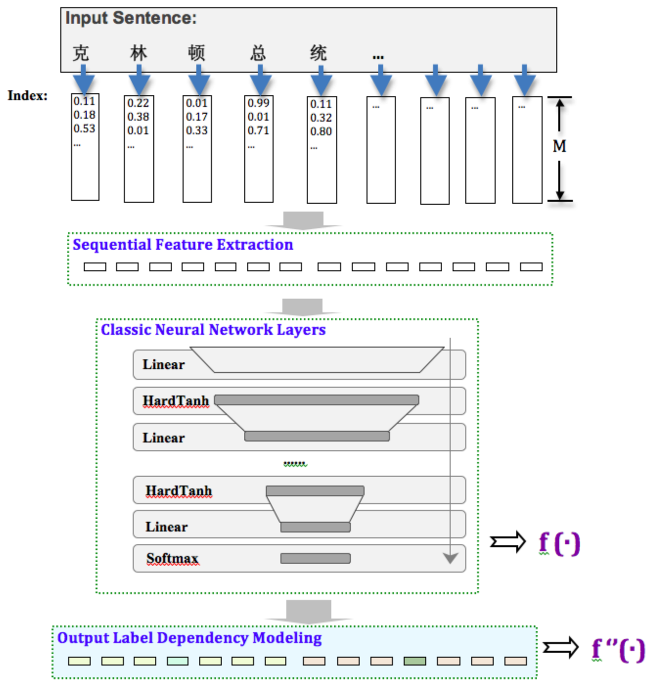
Supplementary for ECIR 2014 Paper "Deep Learning for Character-based Information Extraction on Chinese and Protein Sequence"

Yanjun Qi Sujatha Das G Ronan Collobert Jason Weston

1 System

Figure 1: The basic deep learning system for character-based IE tagging.



2 Experiments

2.1 Data Sets:

Table 1 summarizes some statistics of the datasets we used in experiments. (1) The CTB data we used for WS and POS is from Chinese Treebank 6.0 (LDC2007T36), released in 2007, encompasses 2,036 text files, containing 28,295 sentences, 781,351 words and about $1.3M$ Chinese characters. (2). The CITYU NER data was from SIGHAN3 [6], which includes around $1.8M$ NE-labeled Chinese characters. (3). The CB513 data for SS task consists of 513 unrelated proteins with known 3D structure. Totally the CB513 includes about $84k$ amino acid characters labeled with SS target tags [8].

Table 1: Summary of datasets used in experiment

Dataset/Task	#Chars	#UniqueChars	#Sent
POS(CTB)	1,288,840	4,447	28,295
WS(CTB)	1,287,159	4,696	28,295
NER(CITYU)	1,816,417	4,678	43,734
SS (CB513)	83,707	25	497

The setup for WS and POS tasks used the 10-folds cross validation (CV), following the same configuration in (Zhang&Clark 2008) [9]. We have evenly partitioned CTB into ten groups, and used nine groups for training and the rest for testing. The partition of CTB is provided in Table 2 The train-test split for NER task was from SIGHAN3. The setup for SS task is the same as the standard seven-folds cross validation on CB513 data. The evaluation metrics are averaged per cross-validation test fold in all CV setting.

For the first three tasks related to Chinese text, we construct a list of characters from all three Chinese datasets to form a dictionary of 11951 characters. We convert all non-Chinese characters (such as English words) to lowercase and represented all numeric tokens by "NUMBER". As shown in the performance comparison table, we also try to learn the vector representations for the character bigrams for Chinese IE. The dictionary of character bigrams is constructed similarly as the character unigram dictionary. We just

Table 2: The ten folds uniform partition of CTB (LDC2007T36) for POS and WS tasks.

Fold Index	Start	End
1	0001	0203
2	0204	0525
3	0526	0767
4	0768	1054
5	1055	2126
6	2127	2329
7	2330	2532
8	2533	2735
9	2736	2938
10	2939	3145

rank all the possible bigrams in the three datasets and use the top (most frequent) $30,000$ from the list. For the last SS task, it uses a small dictionary containing only the possible amino acid letters.

2.2 Output Tag Label:

Table 3: Summary of Output Labels

Task	#Labels	Example Tags
NER	14	B-LOC, I-LOC, S-ORG
POS	107	S-NR, B-NN, E-VV, B-DT
WS	5	B, I, E,S
CB513	4	H, B, C

Table 3 gives the number (second column) of target labels used for each task. We employ the IOBES coding scheme for WS, POS and NER tasks. This makes WS task has 5 different tag classes, NER has 14 different tagging labels, and similarly, POS tagging has 107 different target classes to predict for. The label of CB513 data utilizes a 3-letter alphabet $\{H, B, C\}$ in the standard way (H = alpha helix, B = residue in isolated beta bridge and C = Coil).

2.3 Hyperparameter:

The deep framework requires the specification of multiple hyperparameters. This includes the size k of the character window, the size h of the hidden layers, and the learning rate λ. We considered $k \in \{3, 5, 7, 9, 11, 13\}$, $h \in \{80, 100, 120, 150, 200\}$, and $\lambda \in \{0.001, 0.01, 0.03, 0.05, 0.1\}$. The parameter for the embedding layer, M, was chosen from $\{15, 30, 50, 70\}$. The hyperparameter selection was based on the cross-validation results on training. The semi-supervised language model was trained using the freely available Chinese wikipedia corpus [1] and the swissprot protein sequence database [2]. For the three tasks related to Chinese IE, we obtained the best results through setting $k = 3$, $h = 150$, the embedding size $M = 50$, the learning rate for $f(\cdot)$ as 0.05 and the learning rate for the output dependency modeling as 0.01. For SS task, the preferred hyperparameter setup is $k = 13$, $M = 15$, $h = 85$.

The software and methods were implemented using the Torch5 [3] machine learning library. Torch is implemented in C and Lua scripting language. In our basic model, for simplicity, we restrict the classical NN part of our deep neural network to one single hidden linear layer and one output linear layer.

2.4 Result Comparison

As pointed out by Table 2 in the main draft, the proposed architecture includes three strategies, using which we assume to improve the ability of tagging character sequences through, (1) learning embedding representation for characters in the deep framework, (2) including the semi-supervised "language model" ("lm") task, and (3) discriminative training of sentence-level label dependency (termed as "vit", since the prediction step uses viterbi algorithm). Accordingly, we systematically test the incremental combinations of these strategies the Table 2 of the main draft . The first configuration "c1") provides the performance of the most basic setup where only the embedding of character unigrams is utilized. The second configure "c1+lm" adds "language model" component on the top of "c1" case, which clearly improves the tagging ability on all tasks. For instance, WS tagging gets improved F1 measure from 94.73% to 95.57% (about 1 percent increasing). The third configuration "c1+vit" describes the result when combining the unigram embedding and discriminative training of tag dependencies (using "vit" term to label this component). The "tag dependencies" strategy dramatically improves the tagging ability on all three tasks. Especially on Chinese NER, the "c1+vit" configuration achieves the

F1 score 85.81%, a big jump from 80.61% when using "c1" embedding alone. Furthermore, the 4_{th} configuration "c1+lm+vit" improves even more when combining the three strategies of character embedding, language modeling and tag dependency training. In the fifth "c1+lm + c2" and the sixth configuration "c1+lm + c2+vit", we added another embedding feature extraction layer for representing the character bigrams (the same operation as unigram embedding, just on each character bigram). Compared to the state-of-the-art results in the last two rows, our sixth model has already beats the best WS result provided in [9] with about 0.7% improvements. For POS task, our final model is the seventh configuration "c1+c2+lm+vit+ws" which combines all the three functional components plus using word-segmentation as features. Note that the word segmentation is added as one extra new discrete features (i.e. again with a new embedding layer). This clearly beats the top system [9] (with the same 10folds CV setup) in the literature. For the NER task, the seventh configuration beats the second top system, and is slightly lower than the best ranked system from SIGHAN3 [6]. For SS, the seventh configure does not have WS features (of course) and also does not use the character bigram features. Instead, it adds a multitasking strategy which has been proposed in [7]. The extra strategy clearly improves our system even more, which makes SS predictions reach the state-of-the-art performance [5].

3 Discussion

We have described a deep learning framework for multiple character-based information extraction tasks. Our work experimentally proves the power of this architecture and its multiple important techniques. Lastly, we just realize that there is a recent paper from Zheng et al. [10] that uses a very similar method as our paper. The method is almost the same as Collobert et al [4]. Different from Collobert et al [4] using words as the basic units, our paper and Zheng et al. [10] use character as the primary basic unit. Zheng et al. [10] focused primarily on Chinese POS and WS. Different from [10], our framework has been further extended to Chinese NER and SS tasks. Our experimental results also improve the state-of-art on POS and WS.

References

[1] Chinese wikipedia, `http://zh.wikipedia.org`.

[2] swissprot protein sequence databas, `http://www.uniprot.org/downloads`.

[3] Torch. `http://torch5.sourceforge.net/`.

[4] Ronan Collobert, Jason Weston, Léon Bottou, Michael, Kavukcuoglu, and Kuksa. Natural language processing (almost) from scratch. *JMLR*, 12:2493–2537, 2011.

[5] P. Kountouris and J. D. Hirst. Prediction of backbone dihedral angles and protein secondary structure using support vector machines. *BMC Bioinf.*, 10(437), 2009.

[6] Gina-Anne Levow. The third international chinese language processing bakeoff: Word segmentation and named entity recognition. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, volume 117. Sydney: July, 2006.

[7] Yanjun Qi, Merja Oja, Jason Weston, and William Stafford Noble. A unified multitask architecture for predicting local protein properties. *PLoS ONE*, 7(3):e32235, 03 2012.

[8] Burkhard Rost and Chris Sander. Combining evolutionary information and neural networks to predict protein secondary structure. *Proteins: Structure, Function, and Bioinformatics*, 19(1):55–72, 1994.

[9] Yue Zhang and Stephen Clark. Joint word segmentation and pos tagging using a single perceptron. In *Proceedings of the 46th Annual Meeting of ACL*, pages 888–896, 2008.

[10] Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. Deep learning for Chinese word segmentation and POS tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 647–657, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.