A Hierarchical LSTM Model for Joint Tasks

Qianrong Zhou, Liyun Wen, Xiaojie Wang,
Long Ma, and Yue Wang

School of Computer,
Beijing University of Posts and Telecommunications,
Beijing, China
{zhouqr,wenliyun,xjwang,
miss_longma,wangyuesophie}@bupt.edu.cn

Abstract. Previous work has shown that joint modeling of two Natural Language Processing (NLP) tasks are effective for achieving better performances for both tasks. Lots of task-specific joint models are proposed. This paper proposes a Hierarchical Long Short-Term Memory (HLSTM) model and some its variants for modeling two tasks jointly. The models are flexible for modeling different types of combinations of tasks. It avoids task-specific feature engineering. Besides the enabling of correlation information between tasks, our models take the hierarchical relations between two tasks into consideration, which is not discussed in previous work. Experimental results show that our models outperform strong baselines in three different types of task combination. While both correlation information and hierarchical relations between two tasks are helpful to improve performances for both tasks, the models especially boost performance of tasks on the top of the hierarchical structures.

Keywords: Hierarchical LSTM · Joint modeling

1 Introduction

It is a normal situation in Natural Language Processing (NLP) that two tasks interact with each other. For example, Chinese word segmentation and POS-tagging, POS-tagging and chunking, intent identification and slot filling in goal-driven spoken language dialogue systems, and so on.

Usually, the second task is modeled after the first one is finished, since the first task is thought to be more fundamental or lower than the second one. It is so called pipeline method, i.e. low level tasks are followed by high level tasks. For example, chunking in character-based languages such as Chinese, Japanese and Thai requires word segmentation and POS-tagging as pre-processing steps [1,2,3]. In Spoken Language Understanding (SLU), intent is firstly identified as a classification problem using Support Vector Machines (SVMs) [4], and then sequence labeling methods such as Conditional Random Field (CRF) [5] are

employed for slot filling task. However, pipeline method suffers from error propagation. Lots of methods for jointly modeling the first and the second tasks simultaneously have been proposed to tackle this problem.

Previous work has shown the effectiveness of joint models. Lyu et al. (2016) [6] introduced a transition-based framework for joint segmentation, POS-tagging and chunking, and it achieved better results compared with a pipelined baseline. Zhu et al. (2010) [7] proposed a joint segmentation and POS-tagging system based on undirected graphical models which could make full use of the dependencies between the two stages. Shi et al. (2015) [8] proposed a hybrid model of Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN) which could exploit possible correlations among intent classification and slot filling. Lee et al. (2015) [9] introduced a new tag addition method turning utterance classification task into sequence labeling task, then classification and slot filling could be analyzed by one sequence tagger. Duh (2005) [10] proposed Factorial Hidden Markov Models (FHMMs) with additional cross-sequence dependencies, enabling information sharing between POS-tagging/chunking subtasks. Li (2010) [11] studied four joint learning approaches on various sequence labeling tasks.

Previous joint modules have been proved to be able to manage the correlations between sub-tasks by jointly modeling two tasks. Most of the joint models aim to model two specific tasks, i.e. the joint model is task specific. For example, a joint model for word segmentation and POS-tagging is not suit for intent identification and slot filling. We argue that there are some common things behind different joint tasks. High level tasks receive information from low level tasks, while raise some constraints on low level side by hierarchical structures. Both of the interactions between two levels and the hierarchical structures for the joint tasks have important influences on the performances of both tasks. This paper therefore considers to build a joint model that can deal with different types of combinations of two tasks by balancing the interactions and hierarchical constraints between tasks in different levels. For example, a single model (or its slight variants) can deal with segmentation and POS-tagging, intent identification and slot filling, and others.

We propose a hierarchical LSTM (HLSTM) model and some its variants. They are used to deal with a wide types of joint tasks without significant modifications. The model has two-layer LSTM, each layer deals with one task. The two LSTMs takes both interactive and hierarchical information into consideration. Hierarchical relations are found to be very important in our experiments for most of joint tasks. It is not seriously considered and discussed in most previous work. All parameters in two layers are estimated together to minimize a joint objective function. Experimental results show that the proposed hierarchical model outperforms non-hierarchical methods in diverse tasks.

The rest of the paper is organized as follows. Section 2 introduces our proposed model in detail; Section 3 presents the tasks and experimental results; Finally, Section 4 draws conclusions.

2 Models

LSTM is the basic unit of models. We give a brief introduction to LSTM, and then propose our models one-by-one.

2.1 LSTM

LSTM model [12] is widely used in NLP since it can deal with arbitrary-length sequences of input. It alleviates the problem of gradients exploding or vanishing in Recurrent Neural Networks (RNNs) by introducing a memory cell. Commonly, a memory cell is composed of four components: an input gate, a forget gate, an output gate and a memory cell. The input gate controls how much information will be updated in memory cell, the forget gate controls how much information from previous time step to be remembered, and the output gate controls how much information will be outputted to next memory cell. At time step t, the hidden state vector $\boldsymbol{h}_t$ is calculated as following:

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}^{(i)}\boldsymbol{x}_t + \boldsymbol{U}^{(i)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(i)}), \tag{1}$$

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}^{(f)}\boldsymbol{x}_t + \boldsymbol{U}^{(f)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(f)}), \tag{2}$$

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}^{(o)}\boldsymbol{x}_t + \boldsymbol{U}^{(o)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(o)}), \tag{3}$$

$$\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot tanh(\boldsymbol{W}^{(u)}\boldsymbol{x}_t + \boldsymbol{U}^{(u)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(u)}), \tag{4}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot tanh(\boldsymbol{c}_t), \tag{5}$$

where $\boldsymbol{i}_t$, $\boldsymbol{f}_t$, $\boldsymbol{o}_t$ are the gating vectors representing the input gate, the forget gate and the output gate respectively. $\boldsymbol{x}_t$ is the input at the current time step. σ denotes the sigmoid function and $\odot$ denotes elementwise multiplication. $\boldsymbol{W}^{(i)}$, $\boldsymbol{U}^{(i)}$, $\boldsymbol{W}^{(f)}$, $\boldsymbol{U}^{(f)}$, $\boldsymbol{W}^{(o)}$, $\boldsymbol{U}^{(o)}$, $\boldsymbol{W}^{(u)}$, $\boldsymbol{U}^{(u)}$ are weight matrices associated with different gates. $\boldsymbol{b}^{(i)}$, $\boldsymbol{b}^{(f)}$, $\boldsymbol{b}^{(o)}$, $\boldsymbol{b}^{(u)}$ are the bias items.

Multilayer LSTM architectures are able to build higher level representations of input data. They can be created by stacking multiple LSTM hidden layers on the top of each other [13]. At time step t, the hidden state vector of layer n-1 is the input of layer n:

$$\boldsymbol{h}_t^{(m)} = f(\boldsymbol{W}^{(m-1,m)}\boldsymbol{h}_t^{(m-1)} + \boldsymbol{W}^{(m,m)}\boldsymbol{h}_{t-1}^{(m)} + \boldsymbol{b}^{(m)}), \tag{6}$$

where $\boldsymbol{W}^{(m,m)}$ denotes weight matrices between two layers.

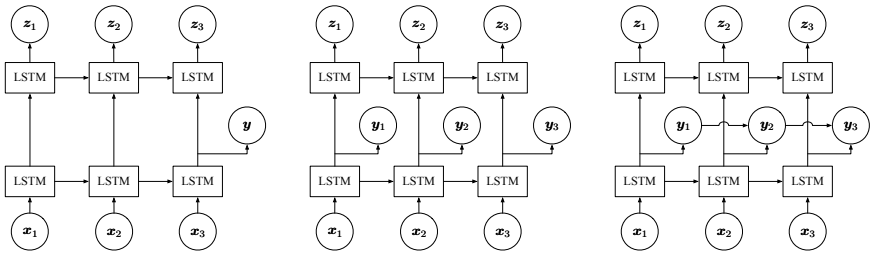
2.2 Hierarchical LSTM and Training

We firstly derive our basic joint model from two-layer LSTM for one basic type of joint tasks, and then propose its variants which fit to different types of joint tasks.

A common type of joint tasks usually combines a sentence-level classification task and a word-level sequence labeling task for a sentence. Examples for

this type of joint tasks include intent identification and slot filling, sentiment classification and sentimental elements extraction, and so on.

Let $w_{(1:n)} = (w_1, w_2, ..., w_n)$ be an input sentence, Y be the label set for sentence, Z be the label set for each word. A joint model assigns a $y \in Y$ to the sentence and $z \in Z$ for each word. A hierarchical model called HLSTM (Hierarchical LSTM) including a sentence-level classification at bottom and a token-level sequence labeling on top is therefore proposed. The structure of HLSTM is shown in Fig. 1(a).



(a) HLSTM (b) HSLTM-L (c) HLSTM-DL

Fig. 1. Our proposed hierarchical models

Suppose the word embedding of w_t is $\boldsymbol{v}_t$ $(1 \leq t \leq n)$ which is used as input of HLSTM. Follow the Eq.(5) we get the last time step of hidden state vector $\boldsymbol{h}_n^{(1)}$ of lower LSTM, and the hidden state vector is fed to a softmax classifier. Then probabilities are estimated for sentence label

$$\boldsymbol{y} = softmax(\boldsymbol{W}^{(1)}\boldsymbol{h}_n^{(1)} + \boldsymbol{b}^{(1)}), \tag{7}$$

where $\boldsymbol{W}^{(1)}$ is a weight matrix for softmax classifier with respect to lower LSTM, and $\boldsymbol{b}^{(1)}$ is a bias term. For each time step t, $\boldsymbol{h}_t^{(1)}$ is still the input of upper LSTM unit. Then we estimate probabilities for sequence output tags

$$\boldsymbol{z}_t = softmax(\boldsymbol{W}^{(2)}\boldsymbol{h}_t^{(2)} + \boldsymbol{b}^{(2)}), \tag{8}$$

where $\boldsymbol{W}^{(2)}$ is a weight matrix for softmax classifier with respect to upper LSTM, $\boldsymbol{b}^{(2)}$ is a bias term.

Two tasks are simultaneously by minimizing a joint loss function. The parameter set of our network is $\theta = \{\boldsymbol{W}^{(i)}, \boldsymbol{U}^{(i)}, \boldsymbol{W}^{(f)}, \boldsymbol{U}^{(f)}, \boldsymbol{W}^{(o)}, \boldsymbol{U}^{(o)}, \boldsymbol{W}^{(u)}, \boldsymbol{U}^{(u)}, \boldsymbol{W}^{(1)}, \boldsymbol{W}^{(2)}\}$ (bias items are omitted). Given a set of training set $\mathcal{D}$, the regularized objective function is the loss function $J(\theta)$ including a l_2-norm term:

$$J(\theta) = \alpha L_S + (1 - a)L_T + \frac{\lambda}{2}||\theta||_2^2, \tag{9}$$

where L_S and L_T are sentence-level loss and token-level loss respectively. Let $\hat{\boldsymbol{y}}^{(i)}$ and $\hat{\boldsymbol{z}}_t^{(i)}$ be correct sentence and token label respectively, $L(\cdot)$ be a cross-entropy error. They are defined as:

$$L_S = \frac{1}{|\mathcal{D}|} \sum_{i}^{|\mathcal{D}|} L(\boldsymbol{y}^{(i)}, \hat{\boldsymbol{y}}^{(i)}), \tag{10}$$

$$L_T = \frac{1}{|\mathcal{D}|} \sum_{i}^{|\mathcal{D}|} \frac{1}{n} \sum_{t=1}^{n} L(\boldsymbol{z}_t^{(i)}, \hat{\boldsymbol{z}}_t^{(i)}) \ . \tag{11}$$

α ($0 < \alpha < 1$) is a hyper-parameter used to tradeoff between two objectives. If $\alpha = 0$, the model only cares about token-level labeling. The network architecture degenerates into standard two-layer LSTM only for token-level labeling task. At the other extreme, if α is set to 1, only sentence-level classification is considered.

In HLSTM (Fig. 1(a)), the sentence-level classification is at the bottom and the token-level sequence labeling is on the top of the model. The inverse situation is another choice, i.e. the sentence-level classification is on the top and the token-level sequence labeling is at the bottom of the model. We call it Inverse HLSTM (I-HLSTM). It has same optimal objective and training algorithm.

Differing from above one, another common type of joint tasks in NLP includes two sequence labeling tasks interacting with each other. For example, POS-tagging and word segmentation, chunking and POS-tagging and so on. We use same hierarchical frame to deal with this type of joint tasks by extending basic HLSTM to HLSTM-L (HLSTM for two Labeling tasks). The structure of HLSTM-L is shown in Fig. 1(b).

HLSTM-L can be formed and trained in same way as those in HLSTM except for substituting class error in HLSTM with sequence label error. Like that in HLSTM, HLSTM-L can also be reversed by exchanging the LSTMs in two layer. We call the Inverse HLSM-L (I-HLSM-L).

In both HLSTM and HLSTM-L, dependencies between sequence labels are not modeled explicitly. For natural language tasks like word segmentation and POS-tagging, dependencies between sequence labels are important.

HLSTM-DL (HLSTM for Dependency Labeling tasks) is therefore proposed by extending HLSTM-L. HLSTM-DL keeps the same frame with HLSTM and HLSTM-L. The structure of HLSTM-DL is shown in Fig. 1(c), where dependencies between labels in lower level LSTM are taken into consideration explicitly.

To model the tag dependency, we follow Chen et al. (2015) [14] by introducing tag transition matrix $\boldsymbol{A}$. Every output tag sequence is given a score by summing tag transition score and tagging score:

$$s(x_{(1:n)}, y_{(1:n)}) = \sum_{t=1}^{n} (\boldsymbol{A}_{y_{t-1} y_t} + (\boldsymbol{y}_t)_{y_t}) \ . \tag{12}$$

Suppose the correct tag sequence of $x^{(i)}$ is $\hat{y}^{(i)}$. Let $Y(x^{(i)})$ be the set of all possible tag sequences. Then the predicted tag sequence $\bar{y}^{(i)}$ can be computed as:

$$\bar{y}^{(i)} = \arg\max_{y \in Y(x^{(i)})} (s(x^{(i)}, y) + \Delta(\hat{y}^{(i)}, y)), \tag{13}$$

where $\Delta(\hat{y}^{(i)}, y)) = \sum_{t=1}^{n} \gamma \mathbf{1}\{\hat{y}_t^{(i)} \neq y_t\}$ is a structured margin loss. γ is a discount parameter, the loss is proportional to the number of words with incorrect tags in the proposed tag sequence.

However, the way we predict most possible tag sequence during testing is a little bit different since the correct tag sequence is unknown, thus Eq.(13) replaced by

$$\bar{y}^{(i)} = \arg\max_{y \in Y(x^{(i)})} s(x^{(i)}, y) \ . \tag{14}$$

Moreover, L_S in Eq.(10) is modified to Eq.(15):

$$L_S = \frac{1}{n} l_S, \tag{15a}$$

$$l_S = \frac{1}{|\mathcal{D}|} \sum_{i}^{|\mathcal{D}|} l_S^{(i)}, \tag{15b}$$

$$l_S^{(i)} = \max(0, s(x^{(i)}, \bar{y}^{(i)}) + \Delta(\hat{y}^{(i)}, \bar{y}^{(i)}) - s(x^{(i)}, \hat{y}^{(i)})) \ . \tag{15c}$$

We can also have I-HLSTM-DL which is the reversed version of HLSTM-DL.

3 Experiments

Three different joint tasks are used for evaluating our models experimentally. They are Intent Classification and Slot Filling in SLU, POS-tagging and Chunking, Chinese Word Segmentation and POS-tagging.

Before the one-by-one introduction of our experiments, it is worthy to mention that hyper-parameters of all our models are tuned only by trying a few different settings in all experiments. We choose the smaller networks to achieve "reasonable" performances rather than picking the best hyper-parameters carefully to achieve their top performances. We employ standard experimental setups for models: for each group of tasks, we use AdaGrad [15] with mini-batches [16] to minimize the objective function. Derivatives are calculated from standard back-propagation [17]. The model achieving the best performance on the development set is used as the final model to be evaluated. The overall performance of joint model is simply evaluated by averaging performances of two tasks. All models are implemented in Theano [18,19].

3.1 Intent Classification and Slot Filling

Joint intent classification and slot filling is typically first type of joint tasks, where the former is a classification task and latter is treated as a sequence labeling problem.

User utterance with only one intent (act) in DSTC2 corpus[1] [20] is used[2]. Basic information about this corpus is listed in Table 1. The number of intent classes is 13, and the number of different slots is 9 including the O label.

Table 1. Intent Classification and Slot Filling corpus

Dataset	**Sentences**	**Tokens**
Training	4,790	19,562
Dev	1,579	6,807
Test	4,485	16,284

HLSTM and its revered version I-HLSTM fit the joint tasks well. The lower layer of HLSTM deals with intent classification and upper layer aims at slot filling task. For I-HLSTM, the lower layer deals with slot filling and upper layer is for intent classification. Lee's model [9], which is a CRF-based joint model for intent classification and slot filling simultaneously, is compared. The open source toolkits is used[3].

Table 2. Test set performance on Intent Classification, Slot Filling

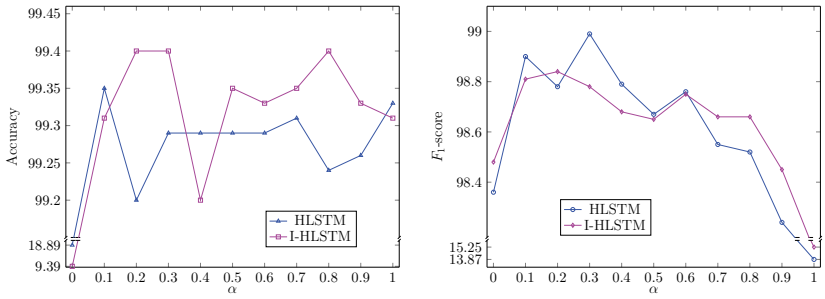
	Intent (Acc.)	Slot (F1)
Lee et al. (2015)	98.84	98.62
I-HLSTM	**99.40**	98.84
HLSTM	99.29	**98.99**

The results are shown in Table 2. From the Table 2, we have two observations. 1) Both HLSTM and I-HLSTM perform better than CRF model on two tasks. It is similar with the conclusion in Mesnil et al. (2013) [21] and Mesnil et al. (2015) [22]. They showed RNN model is better than CRF in slot filling only task. 2) Although the overall performances of two HLSTMs are almost same, HLSTM achieves a slightly higher performance and they show different strengths. HLSTM prefers to slot filling task, while I-HLSTM prefers intent classification. Reminded

[1] `http://camdial.org/~mh521/dstc/`

[2] Sentences with 'request' intent are not included, since there is always no slot values in those sentences.

[3] `http://taku910.github.io/crfpp/`



(a) Intent Classification (b) Slot Filling

Fig. 2. Performances with different values of α

that slot filling is on the top of HLSTM, and intent classification is also on the top of I-HLSTM, we think there is a preference for top-layer LSTM in our two-layer LSTM models. It is a helpful hint on how to make choice between HLSTM and I-HLSTM.

α is used to leverage loss function of two tasks. Fig. 2 shows how performances of model change with α. We can find several points from Fig. 2. 1) It is clear that both HLSTM and I-HLSTM achieve best performance when $\alpha \neq 0$ and $\alpha \neq 1$. It means two tasks can help each other if α is properly given. $\alpha = 0.3$ seems to be a good choice. On the contrary, improper combinations might hurt both of them. 2) Again, we find different strengths of two HLSTMs. Hierarchical structure help tasks on top-layer receive better scores.

3.2 POS-tagging and Chunking

Both POS-tagging and chunking can be regarded as sequence labeling problems. HLSTM-L and its reverse version are therefore used for joint modeling of the two tasks.

Penn Treebank Wall Street Journal corpus is used for joint POS-tagging and chunking. Basic information about this corpus is listed in Table 3. Their training data consists of sections 02-21 and test data consists of section 00. 10% of the training set is split into a development set. The corpus contains 45 different types of POS tags, and 23 types of chunking tags respectively. A transformation-based learning model [23] was used for comparison since it used same dataset.

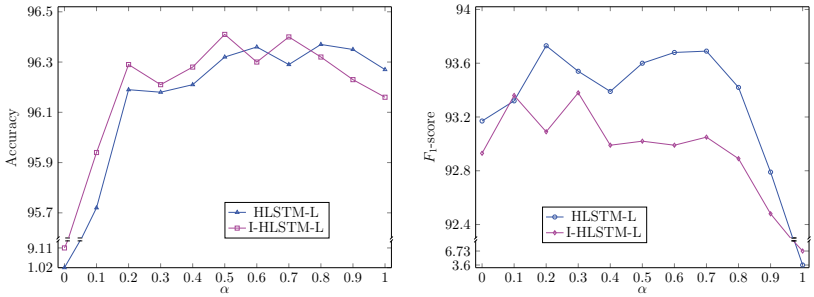
Table 3. POS-tagging and Chunking corpus

Dataset	Sentences	Tokens
Training	39,831	950,011
Test	1,920	46,435

Experimental results are shown in Table 4. As can be seen from Table 4, 1) Both HLSTM-L and I-HLSTM-L are comparable with Florian et al. (2001). They achieve improvements on chunking, but a little lower in POS-tagging task. 2) HLSTM-L who models the hierarchical structure of joint tasks performs better overall. Similar to that in previous model, an appropriate value of α is crucial to the performance of both tasks. The impact of α is shown in Fig. 3.

Table 4. Test set performance on POS-tagging, Chunking

	POS (Acc.)	Chunking (F1)
Florian et al. (2001)	**96.63**	93.12
I-HLSTM-L	96.21	93.38
HLSTM-L	96.36	**93.68**



(a) POS-tagging (b) Chunking

Fig. 3. Performances with different values of α

The results in Fig. 3 illustrate that 1) both POS-tagging and chunking could be benefit by joint training. POS-tagging gets a performance boost and achieves best performance by introducing moderate chunking information, the same situation applies to chunking. $\alpha = 0.6$ seems to be a good choice. 2) Two HLSTM-Ls show different strengths on different tasks. HLSTM-L prefers to chunking, while I-HLSTM-L prefers POS-tagging.

3.3 Chinese Word Segmentation and POS-tagging

Both Chinese word segmentation (CWS) and POS-tagging can be regarded as sequence labeling problems. Instead of using HLSTM-L, the influence of dependency relation is taken into consideration, and HLSTM-DL is employed to model

the joint tasks. Since the result of POS-tagging includes the CWS, it is not possible to put CWS on the top layer with POS-tagging at the bottom, the reversed version will not be included in this experiment. Inspired by Pei et al. (2014) [24], bigram embeddings are used as models' inputs as well.

NLPCC 2015 dataset[4] [25] is used for the joint tasks. Different with the popular used newswire dataset, the NLPCC 2015 dataset collects informal texts from Weibo. The information of the dataset is shown in Table 5. 10% of the training set is split into a development set and keep the remaining 90% as the real training set. Each character is labeled as one of {B, M, E, S} to indicate the segmentation. For POS-tagging labels, each is the cross-product of a segmentation label and a POS tag, e.g. {B-NN, M-NN, E-NN, S-VP, ...}.

Table 5. CWS and POS-tagging corpus

Dataset	Sents	Words	Chars	Word Types	Char Types	OOV Rate
Training	10,000	215,027	347,984	28,208	3,971	-
Test	5,000	106,327	171,652	18,696	3,538	7.25%

A CRF model for joint CWS and POS-tagging, which presented in the NLPCC 2015 shared task, is used as baseline. The templates are unigram feature, bigram feature and trigram feature. Word segmentation can be inferred from the output.

The results are shown in Table 6. HLSTM-DL achieves much improvement compared to HLSTM-L. Even with fewer features, HSLTM-DL gets a much better result on CWS and performs slightly better on POS-tagging compared to CRF model.

The effect of α is shown in Fig. 4. We can see that POS information brings a significant boost to CWS. The same analysis of α also applies to this experiment like previous ones.

Table 6. Test set performance on CWS, POS-tagging

	CWS (F1)	POS (F1)
Qiu (2015)	93.80	87.69
HLSTM-L	92.10	85.44
HLSTM-DL	**95.10**	**87.75**

We compared our proposed hierarchical models with non-hierarchical models on different kinds of combinations of NLP tasks. In most situation, hierarchical

[4] `http://nlp.fudan.edu.cn/nlpcc2015`

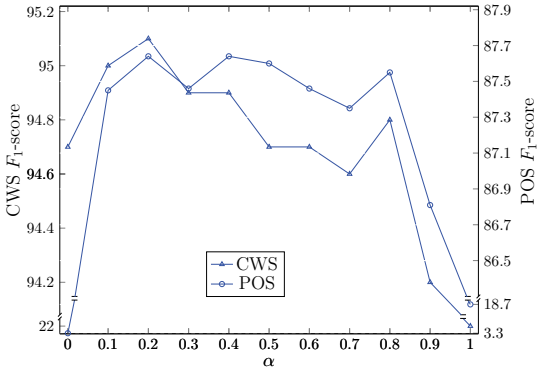


Fig. 4. Performances with different values of α

models achieve better results than strong baselines. Top-layer in hierarchical model is more powerful. Hierarchical model reflecting task hierarchy is likely to achieve higher overall performance. The detailed analysis of experimental results shown that the hyper-parameter α of hierarchical joint model can leverage information in one task to another, thus bring significant performance boosts to both tasks.

4 Conclusion

We have presented a hierarchical LSTM model and several its variants that can handle different kinds of combinations of NLP tasks. Experimental results on three different combinations of NLP tasks show promising results. In most situation, they outperform strong baselines. Hierarchical relations and correlation information between different layers are also discussed experimentally. Tasks get better performances if they are on top-layer. We believe they provide a series of potential solutions for joint learning of two NLP tasks.

There are several problems waiting for future work. While the hierarchical structure has been shown to be a good choice for arranging two NLP tasks, it is still not so clear how the information of two tasks interacts each other, especially how tasks at the bottom transmit supervision to the tasks on the top layer, and vice versa. Another problem is how to character dependency relation between labels in sequence labeling tasks.

Acknowledgments. This paper is partially supported by National Natural Science Foundation of China (No 61273365), discipline building plan in 111 base (No.B08004) and Engineering Research Center of Information Networks of MOE, and the Co-construction Program with the Beijing Municipal Commission of Education.

References

1. Zhou, J., Qu, W., & Zhang, F. (2012, July). Exploiting chunk-level features to improve phrase chunking. In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (pp. 557-567). Association for Computational Linguistics.
2. Chen, W., Zhang, Y., & Isahara, H. (2006, July). An empirical study of Chinese chunking. In Proceedings of the COLING/ACL on Main conference poster sessions (pp. 97-104). Association for Computational Linguistics.
3. Tan, Y., Yao, T., Chen, Q., & Zhu, J. (2005). Applying conditional random fields to chinese shallow parsing. In Computational Linguistics and Intelligent Text Processing (pp. 167-176). Springer Berlin Heidelberg.
4. Fan, R. E., Chang, K. W., Hsieh, C. J., Wang, X. R., & Lin, C. J. (2008). LIBLINEAR: A library for large linear classification. The Journal of Machine Learning Research, 9, 1871-1874.
5. Lafferty, J., McCallum, A., & Pereira, F. C. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data.
6. Lyu, C., Zhang, Y., & Ji, D. (2016, March). Joint Word Segmentation, POS-Tagging and Syntactic Chunking. In Thirtieth AAAI Conference on Artificial Intelligence.
7. Tie-jun, Z. C. H. Z., & De-quan, Z. (2010). Joint Chinese Word Segmentation and POS Tagging System with Undirected Graphical Models [J]. Journal of Electronics & Information Technology, 3, 038.
8. Shi, Y., Yao, K., Chen, H., Pan, Y. C., Hwang, M. Y., & Peng, B. (2015, April). Contextual spoken language understanding using recurrent neural networks. In Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on (pp. 5271-5275). IEEE.
9. Lee, C., Ko, Y., & Seo, J. A Simultaneous Recognition Framework for the Spoken Language Understanding Module of Intelligent Personal Assistant Software on Smart Phones. Volume 2: Short Papers, 818.
10. Duh, K. (2005, June). Jointly labeling multiple sequences: A factorial HMM approach. In Proceedings of the ACL Student Research Workshop (pp. 19-24). Association for Computational Linguistics.
11. Li, Xinxin. (2010). Research on Joint Learning of Sequence Labeling in natural language Processing (Dissertation for the Doctoral Degree in Engineering). Harbin Institue of Technology, Harbin, China.
12. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.
13. Graves, A., Mohamed, A. R., & Hinton, G. (2013, May). Speech recognition with deep recurrent neural networks. In Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on (pp. 6645-6649). IEEE.
14. Chen, X., Qiu, X., Zhu, C., Liu, P., & Huang, X. (2015). Long short-term memory neural networks for chinese word segmentation. In Proceedings of the Conference on Empirical Methods in Natural Language Processing.
15. Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. The Journal of Machine Learning Research, 12, 2121-2159.
16. Cotter, A., Shamir, O., Srebro, N., & Sridharan, K. (2011). Better mini-batch algorithms via accelerated gradient methods. In Advances in neural information processing systems (pp. 1647-1655).

17. Goller, C., & Kuchler, A. (1996, June). Learning task-dependent distributed representations by backpropagation through structure. In Neural Networks, 1996., IEEE International Conference on (Vol. 1, pp. 347-352). IEEE.
18. Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I., Bergeron, A., ... & Bengio, Y. (2012). Theano: new features and speed improvements. arXiv preprint arXiv:1211.5590.
19. Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., ... & Bengio, Y. (2010, June). Theano: a CPU and GPU math expression compiler. In Proceedings of the Python for scientific computing conference (SciPy) (Vol. 4, p. 3).
20. Williams, J., Raux, A., Ramachandran, D., & Black, A. (2013, August). The dialog state tracking challenge. In Proceedings of the SIGDIAL 2013 Conference (pp. 404-413).
21. Mesnil, G., He, X., Deng, L., & Bengio, Y. (2013, August). Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In INTERSPEECH (pp. 3771-3775).
22. Mesnil, G., Dauphin, Y., Yao, K., Bengio, Y., Deng, L., Hakkani-Tur, D., ... & Zweig, G. (2015). Using recurrent neural networks for slot filling in spoken language understanding. Audio, Speech, and Language Processing, IEEE/ACM Transactions on, 23(3), 530-539.
23. Florian, R., & Ngai, G. (2001, July). Multidimensional transformation-based learning. In Proceedings of the 2001 workshop on Computational Natural Language Learning-Volume 7 (p. 1). Association for Computational Linguistics.
24. Pei, W., Ge, T., & Chang, B. (2014). Max-Margin Tensor Neural Network for Chinese Word Segmentation. In ACL (1) (pp. 293-303).
25. Qiu, X., Qian, P., Yin, L., Wu, S., & Huang, X. (2015). Overview of the NLPCC 2015 Shared Task: Chinese Word Segmentation and POS Tagging for Micro-blog Texts. In Natural Language Processing and Chinese Computing (pp. 541-549). Springer International Publishing.